Automation Strategies for Sample Preparation in Life Science Applications

Dissertation

zur

Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

der Fakultät für Informatik und Elektrotechnik

der Universität Rostock

Submitted by:

Xianghua Chu, born on 17th, August 1988 in Shandong Province, China

Rostock, Germany, 2016

urn:nbn:de:gbv:28-diss2018-0191-1

Reviewers:

1. Reviewer:
   Prof. Dr. -Ing. habil. Kerstin Thurow
   Institute of Automation, University of Rostock, Germany
2. Reviewer:
   PD Dr.-Ing. habil. Heidi Fleischer
   Institute of Automation, University of Rostock, Germany
3. Reviewer:
   Prof. Edward Grant, Ph.D.
   Department of Electrical and Computer Engineering, North Carolina State University, USA

Date of Submission: 25th, November, 2016.

Date of Defense: 16th, October, 2017.

Acknowledgement:

This doctoral dissertation would not have been completed without the guidance and support from the following several persons.

First of all, I wish to express my deep gratitude to my supervisors Prof. Dr.-Ing. habil. Kerstin Thurow and Prof. Dr.-Ing. Norbert Stoll for their elaborate guidance, continuous support and precious knowledge. Their suggestions inspired me in all the time of researching and writing of this dissertation.

Besides, I would like to thank all my colleagues of the Institute of Automation and the Center for Life Science Automation (celisca) at University of Rostock, Germany. Without their kinds of support and help to my work and daily life in Germany, this dissertation would not have been finished on time. Especially, I would like to thank PD Dr.-Ing. habil. Heidi Fleischer, Dr. Steffen Junginger, Dr. Thomas Roddelkopf, Dr. Sebastian Neubert, Mr. Lars Woinar, Mr. Heiko Engelhardt and Ms. Sybille Horn for their competent help.

I would also like to acknowledge the support of Dr. Michael Klos and Wolfgang Schuberthan of Yaskawa Europe GmbH.

Last but not least, I would like to thank my father Mr. Qingyu Chu and my mother Mrs. Ruiqing Zhao for their financial support and encouragement.

CONTENTS

List of Figures

List of Tables

List of Abbreviations

LC-MS	Liquid chromatography–mass spectrometry
GC-MS	Gas chromatography–mass spectrometry
GUI	Graphical user interface
API	Applicatin programming interface
HTS	High throughput screening
LSA	Life science automation
DAR	Dual-arm robot
DOF	Degree of freedom
GP	Glass pipette
TI	Tip box
VI	Vial
TM	Thermoshaker
HO	Hotel
MTP	Microplate
PE	Pipette
SY	Syringe
XML	eXtensible Markup Language
PC	Personal computer
SPE	Solid phase extraction
EIS	Electrospray ionization source
UDP	User datagram protocol

Chapter 1 Introduction

The term “Automation” was coined by D.S. Harder in 1936 [1] and was defined as “the execution by a machine agent (usually a computer) of a function that was previously carried out by a human” by R. Parasuraman and V. Riley in 1997 [2]. As automation can save human labor, improve productivity, improve accuracy and precision or perform some hazardous tasks which may put a human at risk, its application has been involved in many fields, including manufacturing, healthcare, security, transportation, agriculture, construction, energy, and many other areas [3].

Automation is also broadly applied in life science field, which includes for example liquid handling automation or high-throughput screening automation [4], [5], [6]. Automation can reduce errors [7], protect human operators while handling hazardous materials [8], ensure uniformity [9] and bring other advantages [4]. In the laboratory automation, robots plays a vital and growing role [10], [11]. Besides the cartesian coordinate robots, which employ x, y and z drives to control the translational motion of the end-effector holding the liquid-dispensing device in the cartesian coordinates [5], robot arms with multiple degrees of freedom, are also widely used in life science automation. As robot arms are flexible to move in the cartesian space, they can be used to enhance the degree of automation by working together with the gantry robots[12], used as the central robot integrating with a series of instruments to improve the productivity [13] and used for other applications [14], [15]. Even an automated screening system, consisting of two single robots, was developed to meet the demands for high-performance and flexibility, supporting early stage drug discovery [16].

Currently, human-like dual-arm robots also draw much attraction in designing an automated manufacturing system [17], [18]. As such robots can move in a similar way to a human due to a structural similarity, they are expected to share the same working environments with humans without artificial barriers in order to reduce costs and also to obtain high flexible automation [19]. Also, they are expected to be quickly reconfigurable to produce a new batch of a completely different product or sub-module (“agile assembly”) [20]. H.M. DO et al. introduced the use of a dual-arm robot for industrial cell production [21].

The automation of processes in life sciences is still a growing field [22]. Biological processes reached a high degree of automation due to the mild conditions and high standardization of these processes and the introduction of the microplate as a standard format [10], [23]. Chemical and analytical processes are still subject for automation since they require different formats [24] and stronger conditions such as temperature, pressure or solvents [25], [26]. Automation strategies for such processes include the use and adaptation of biological screening systems with central robots as system integrators. Liquid handling systems, shakers and heaters are used and adapted to perform the necessary reaction steps. A developed system has been used for performing sample preparation for elemental analysis including the determination of mercury or calcium in organic materials [27]. This approach requires a lot of adaptation of classical laboratory hardware as well as the development of new devices for transferring the manual steps to an automated system. A different approach will be the use of a dual-arm robot. The robot can be used to perform all necessary steps in the processes in human-like way. Thus, less adaption of existing and development of new devices are required. Also, the validation of the processes is easier since there are no changes in the processes while transferring them from manual to an automated procedure. The direct integration of analytical systems such as liquid chromatography–mass spectrometry (LC-MS) instrument is a challenging task as analytical systems are usually closed systems without general interfaces for automated sample delivery from other automation systems or robots. Thus the adoption of complementary liquid-handling robotic tools to prepare samples for analysis by LC-MS has not been as universal or as rapid [28].

After the review of the literatures regarding the state-of-art of robot applications in life science automation and the applications of dual-arm robots (Chapter 2), Chapter 3 describes the processes of sample preparation, which has to be automated. In Chapter 4, the aims of this dissertation is discussed. In Chapter 5, the overview about the robot platform is shown, including the description of the tools for preparing samples and the layout of the platform. In the case of applications in life science automation, the main challenges come from the complex experiment processes, which may require thousands of robot motion steps to fulfill it, and the narrow space environment restricting robot movement. The design of robot motions determines the flexibility and even feasibility of the dual-arm robot platform. The programming strategy for the dual-arm robot is discussed in Chapter 6. The grippers are critical parts of robot platform. The integration of grippers to the robot platform is discussed in Chapter 7. The codes of commercial software are likely to be kept secret, only providing the programming interfaces for system integration. If there is no document detailing the way of using the program interfaces or no available program interfaces, it will be very confined to perform software integrations or even impossible. However, if the software integration is achieved through its graphical user interface (GUI), there will be no such problems. In our application, for integrating the LC-MS software and the GC-MS software with the SAMI software, some parts of the integration have been achieved by using the APIs of the LC-MS software and the GC-MS software, while the other parts have been realized through “operating” the GUIs. The software integrations are described in Chapter 8. Chapter 9 is about the robot program interface (called R-interface) for integrating the robot system with the SAMI system. For one side, the R-interface will interactive with SAMI by receiving and sending XML format data. SAMI will send commands to R-interface. And R-interface will execute them and feedback the results to SAMI. For the other side, R-interface will send call-files to the robot controller in order to drive the robot. A call-file is used to integrate the motion elements. In other word, the R-interface receives commands from SAMI and then controls the robot to fulfill the commands. An M-database is generated by R-interface to record the information of motion elements. When executing the SAMI commands, the related motion elements will be searched from the M-database. Together with parameters, the names of these related motion elements will be integrated into a call-file. The interactivities between R-interface and the robot system are based the HSE protocol, including handling robot jobs, monitoring the status of robot and doing other things. The integration with SAMI software is discussed in Chapter 10. The testing results, including analysis results of real samples by LC-MS and GC-MS, has been shown in Chapter 11. Chapter 12 gives the conclusions and outlook.

Chapter 2 Literature Review

2.1 Introduction

Research activities in life science have been rising in recent years, and there is a high requirement of automated systems, with robots playing a vital role in many of laboratory procedures, such as chemical analysis, drug development and DNA fingerprinting [10], [29]. There are many advantages of using robots, including the reduction of manual errors, providing highly accurate dispensing and ensuring uniformity [30]. By using the robot to conduct the mundane and repetitive tasks, the scientists can be freed up to focus on more difficult but rewarding challenges [28].

High throughput screening (HTS), having its origin in natural products screening in 1986 in the Pfizer Inc. [31], is an important method used in the field of life sciences, allowing users to conduct large numbers of tests in short time. HTS systems can greatly cut the initial phase of drug discovery, from several years to a matter of months [10], significantly increase throughput and reduce assay volumes [32]. The microplate is an essential lab ware to HTS, permitting to automate liquid handling. However, microplates are trended to be made in different formats [33], which is potential to cause problems when use them in automated instruments or on robots. In order to avoiding problems caused by microplate formats, the American National Standards Institute published a series of microplate standards, describing the definitions of a microplate [34]. Then the flourishing of liquid-handling robots in the life-science laboratories followed [22].

Liquid handling is a fundamental part of many experiments related to the life science. Cartesian coordinate robots, using x, y and z drives to control the movements of the end-effector (e.g. liquid-dispensing devices) in the cartesian coordinates, are available to handle the liquids [35], [36]. Robot arms with more degrees of freedom (normally no less than six axes), are more flexible to move in the cartesian coordinate system and are suitable to transfer microplates and another lab ware. By using robot arms to transfer plates, cover plates or conduct other tasks, the degree of automation can be increased. Also by integrating robot arms with the liquid handling robots, high throughput can be provided [37]. Besides, the robot arm can be used as the central robot to integrate different instruments or used for special purposes.

However, robot workstations are likely to work as ‘islands’, independent with each other. A key barrier to automation is the integration of hardware and software tools, as they possibly have different interfaces. Mobile robots can be used as a tool to integrate the workstations, which are located far away in different rooms or even in different floors [38].

This section has firstly provided the literature review about applications of robots in LSA. As dual-arm robots (DAR) also draw many attentions, which are still not widely used in LSA, the state-of-art of DARs and their possible advantages to LSA has also been discussed in this chapter.

2.2 Robots in Life Science Automation

This part begins with the introduction of liquid-handling workstations. For obtaining higher degrees of automation, robots with more degrees of freedom are needed. Several multi-DOF (multiple degrees of freedom) robot workstations, used in crystal harvesting, toxicity assessing, cell culture and drug discovery respectively, are discussed in this part. Then, a mobile robot system used in life science automation is introduced. Robot scientist seems to be the developing tendency of robot workstations. So robot scientist systems are also shown in this section. In the end is a summary.

2.2.1 Liquid-handling Workstations

By automating liquid handling, the costs and assaying time can be reduced. Liquid-handling workstations transfer liquid from one source location (e.g. a vial, a reservoir or a microplate) to one destination location (e.g. a vial, a reservoir or a microplate) at a fast speed. Liquid dispensing can utilize various tools, such as pin tools or syringes [5], [39], [40]. Y. Liu at el, had developed a dispensing system for handling highly viscous reagent. Figure 1 is about the schematic of the system [35]. This system uses the syringe and the needle to dispense liquid, with a step motor to drive the motion of the syringe plunger, a CCD camera for measuring the volume of the droplet, and a 3-axis robot for moving the dispenser and the stage (the z axis moves the dispenser up and down, while the x axis and y axis are for moving the stage). As a result, the system's accuracy can be below 5% and the repetitive precision can be below 8% CV when dispensing 40 nL viscous reagent [35]. C. Uematsu at el. had built a liquid handling system for polymerase chain reaction sample preparation, named as RAD (see Figure 2) [36]. This system uses glass capillaries as the pipetting unites. And the XYZ stage is used to position the pipetting unit at any well of the four 96-well microplates or one 384-well microplate. As a result, liquid samples with volumes greater than 0.2 μL can be dispensed [36].

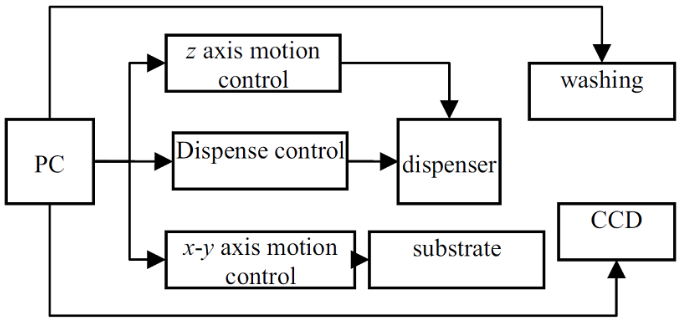


Figure 1. The schematic of dispensing system [35].

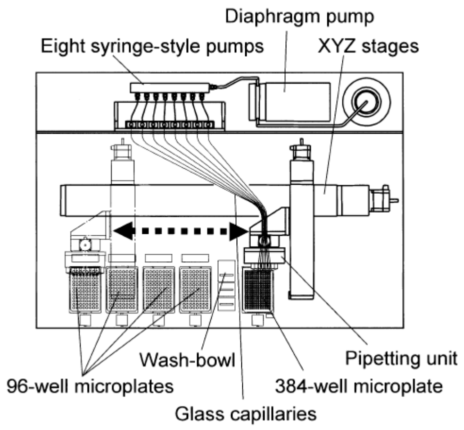


Figure 2. A schematic view of the random access dispenser (RAD) [36].

2.2.2 Automated Crystal Harvesting Workstation

Knowing crystal structures is helpful to develop drugs. In the determination of a protein structure, the harvesting of protein crystals is an essential step. As protein crystals are normally fragile, they are easily to damage when harvesting. Crystals harvesting, which is traditionally done manually by skilled operators, is a bottleneck to high-throughput crystallography. Automated protein crystal harvesting is necessary and will increase throughput, offer the ability to harvest microcrystals and improve flash-cooling procedures [41]. R. Viola at el. had designed a robot workstation, using the Stäubli RX60 as the core robotic component, to address the final frontier in achieving fully automated high-throughput crystallography (see Figure 3) [42]. The robot is reliable to harvesting protein crystals as small as 10 μm from a variety of 96-well plates. The images of each of the plate's wells will be acquired, magnified and displayed to the operator, assisting the operator to identify the wells holding usable crystals [42]. M.Y. Heidari Khajepour et al. had develop the Robotic Equipment for Automated Crystal Harvesting (REACH) on beamline FIP-BM30A at the European Synchrotron Radiation Facility (ESRF) in order to achieve more robust and reliable macromolecular crystal harvesting and flash cooling operations that are compatible with most standard macromolecular crystallography equipment [14].

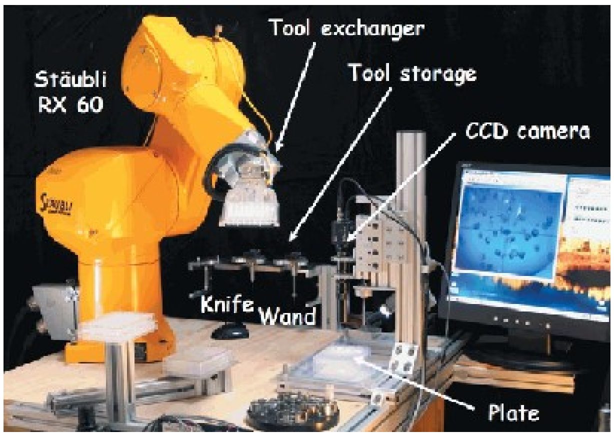


Figure 3. The robotic prototype [42].

The experimental setup can be seen in Figure 4. The G-Rob robot arm, which is accuracy to operate as a goniometer, is equipped with a micro-gripper for crystal handling. The Visualization Bench is used to screen crystallization plates [14]. Besides, a wireless mobile micro-robot, driven by rotating magnetic fields was also developed to assist in automated protein crystal harvesting [43].

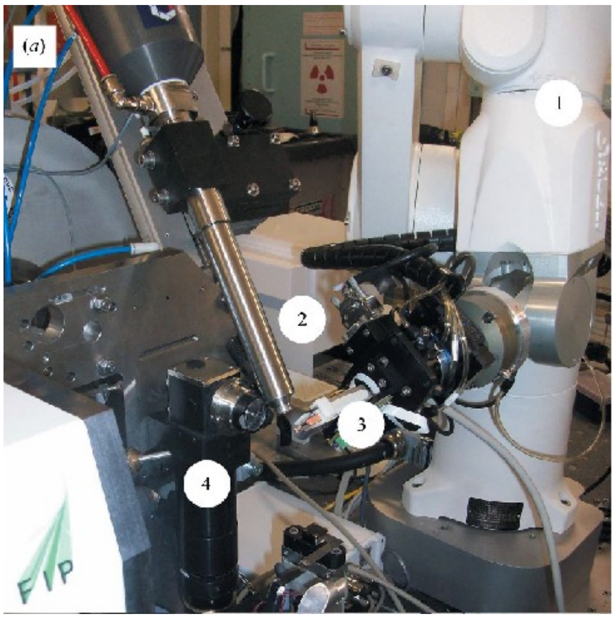


Figure 4. Experimental setup on the FIP-BM30A beamline [14].

2.2.3 The US "Tox21" Robot System

It is important to assess the toxicity of compounds in products, such as food additives or drugs, in order to protect our health. The in vivo toxicology is the traditional method for the chemical hazard evaluation, but it has low testing speed and high animal usage. However, the toxicity testing, with high throughput and a drastic reduction in the use of experimental animals, is expected in the future [44], [45]. The US "Tox21" program was established in 2008, aimed at accelerating the development of mechanism-based in vitro screens to better understand the mechanisms of toxicity and to reduce the use of low-throughput, high-cost traditional toxicity testing relied on animal models [13], [46]. The "Tox21" system consists of a series of peripherals and workstations arranged around a central Stäubli robotic arm (Figure 5). The main components include the Kalypsys Director software, a pin tool device for nanoliter compound transfer, plate storage and environmentally controlled assay incubation units, nanoliter reagent dispensers, a centrifuge, and plate readers [13]. The set of chemicals to be tested has been expanded to nearly 10,000 (10K) compounds and the Tox21 robotic system is capable of screening and profiling the collection of 10K environmental chemicals in triplicate in a week [46].

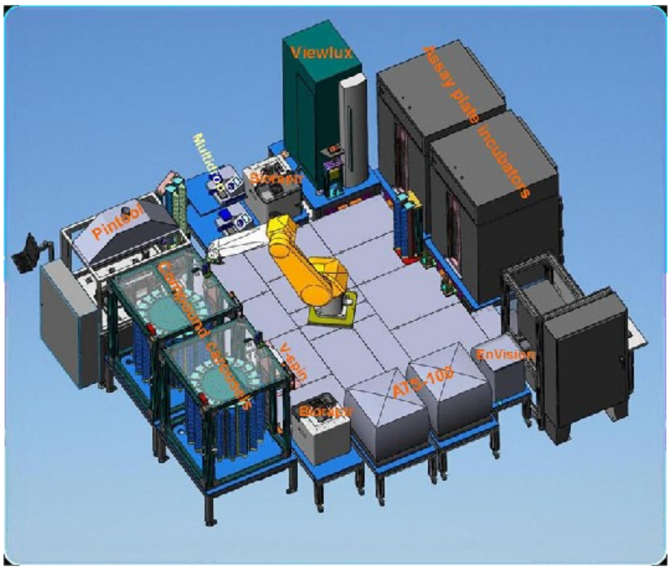


Figure 5. The Tox21 robotic system [13].

2.2.4 Cell Culture Automation

Cell culture is usually carried out under strictly controlled laboratory conditions (including pH, temperature, gases, and pressure) and can produce uniform cultured cell lines, which is useful to manufacture biological products (e.g. viral vaccines) [47]. With the requirements of high throughput and standardization, cell culture automation becomes necessary, which can also provide power to statistical analysis and enable large numbers of variables to be manipulated [48], [49], [50].

The Automation Partnership (TAP) had devised an automated cell culture system branded "Cellmate" to automated cell culture methods. The Cellmate system consists of a 6-degrees of freedom Stäubli RX60 robot arm, in-feed and out-feed conveyors, an attached incubator for roller bottles, and a series of media delivery and retrieval pumps, and automates these operations, including cell seeding, media changes, bottle gassing and cell sheet rinsing. It can process between 10 and 500 roller bottles or tissue culture flasks containing as many as 10 different cell lines a day [51], [15]. Besides, TAP had launched another automated system, marketed as “SelecT”, which can plate 100–300 plates of cells per day [52]. The schematic diagram can be seen in Figure 6. The robot can handle both flasks and pipettes and is capable of media exchange and cell sheet rinsing as well as cell splitting to harvest, pool and seed cells into new flasks [15].

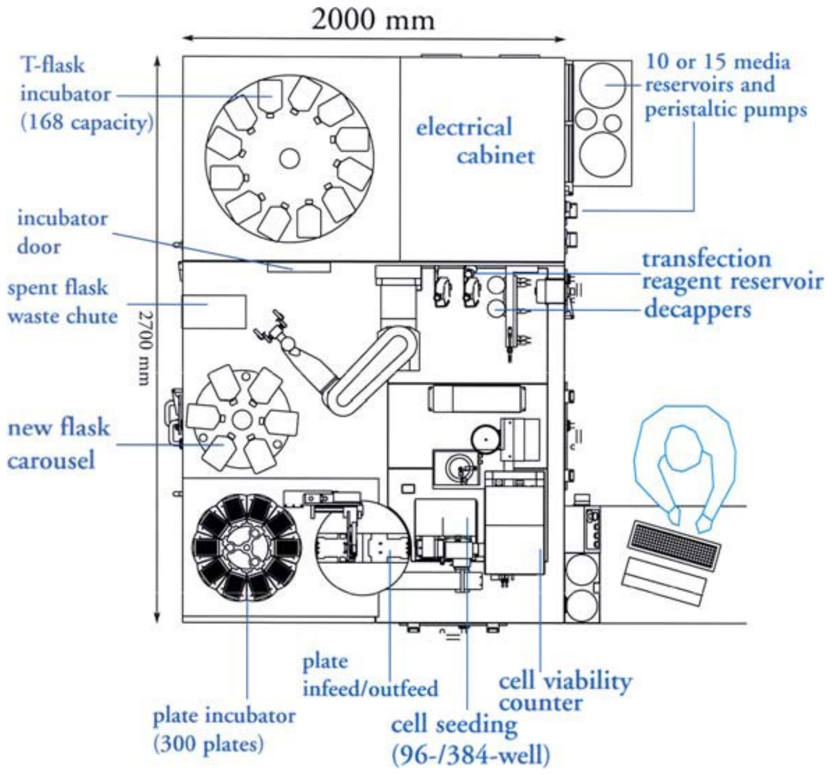

2000 mm
2700 mm
T-flask incubator (168 capacity)
incubator door
spent flask waste chute
new flask carousel
electrical cabinet
10 or 15 media reservoirs and peristaltic pumps
transfection reagent reservoir
decappers
plate incubator (300 plates)
plate infeed/outfeed
cell seeding (96-/384-well)
cell viability counter

Figure 6. Schematic diagram of automated cell culture system (TAP SelecT) [52].

Celisca also have built an automated cell culture system, which is support for high through put screening and 3D cell culture [12], [53]. The top view of the automated cell culture system can be seen in Figure 7. The main components of this system include a Motoman HP 3 JC robot arm (1) for transporting well plates, a Cytomat incubator (2), Cytomat hotels (3) for storing well plates and tips, a Biomek NX (3) and a Biomek FX (5) liquid handlers for different pipetting tasks and a PHERAstar Plus plate reader (6), a FLUOstar galaxy reader (7) as well as a NOVOstar microplate reader (8) for quantifying luminescence, fluorescence or absorbance [12].

2.2.5 A Kind of Dual-Arm Robotic Platform

The automation in drug discovery demands high performance and flexibility. To meet the requirements, the MSD with RTS Life Science have developed a dual-robot arm screening system at the Terlings Park site, which includes two Stäubli robot arms. Figure 8 shows the layout of the robotic platform. Associated with micro plate grippers, the two robot arms were used as the sample transfer mechanism. They can access to different instruments (while one robot arm can access a room temperature storage facility, an eight-tip liquid-handling workstation, and a plate sealer, the other robot arm is used to access a different room temperature storage facility, a 96/384 liquid-handling workstation, a 96-1536 bulk liquid dispenser, and a 96/384 well plate washer) and can also reach some ‘communal’ instruments: a temperature and CO2-controlled incubator, plate waste bins, and two plate readers. Besides, the system can work in two operating modes: combined mode, with the entire system operating in unison, and independent mode, with each robot arm performing two different screens in parallel. [16]

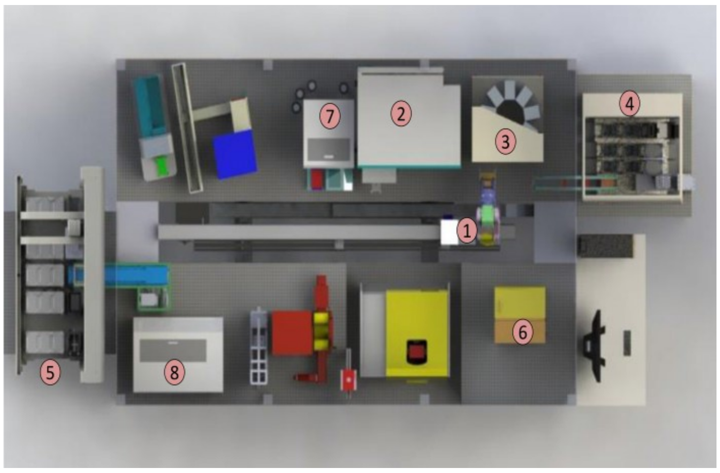


Figure 7. The top view of the automated cell culture system [12].

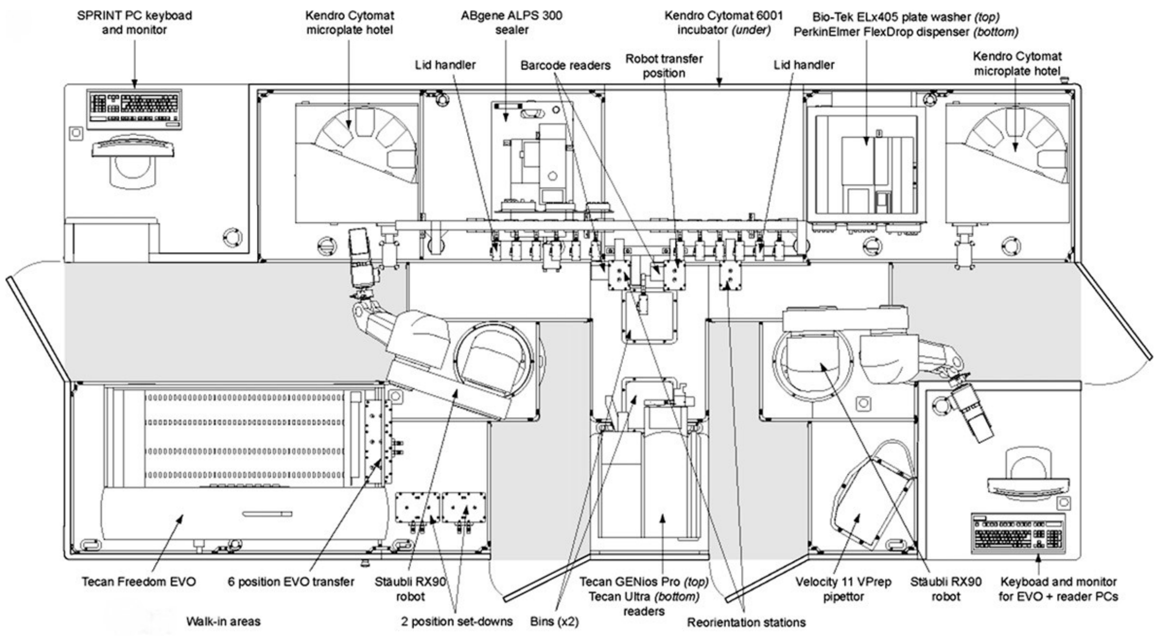


Figure 8. Instrument layout within robot cell [16].

2.2.6 Mobile Robots in Life Science Automation

Automated workstations are likely used as “islands” for doing different or similar tasks independently. Transports will need to be done manually or automatically, when the export from one workstation is the import of the other workstation. Mobile robots are the suitable candidates for the transports between workstations which are located far away or in different rooms [54], [55].

Mobile robots play an important role in life science automation. Celisca has developed a control system for mobile robots in a life science laboratory, supporting the long-distance transportations between the distributed workbenches. The architecture of the system is shown in Figure 9, which comprises a laboratory process management system (PMS), a robot remote center (RRC), and robot onboard/boarding centers (RBCs) controlling all the hardware modules inside the robots [56]. While the PMS takes charge of the life science process control, the RRC manages robot path planning, robot status monitoring and communication with both the PMS and the RBCs [57]. When doing the long-distance transportation, robots need to be recharged frequently. The charging of multi mobile robots, which work in the laboratory at the same time, is managed effectively [58]. Besides, the mobile robots can perform multi-floor transportations in the laboratory environment [59].

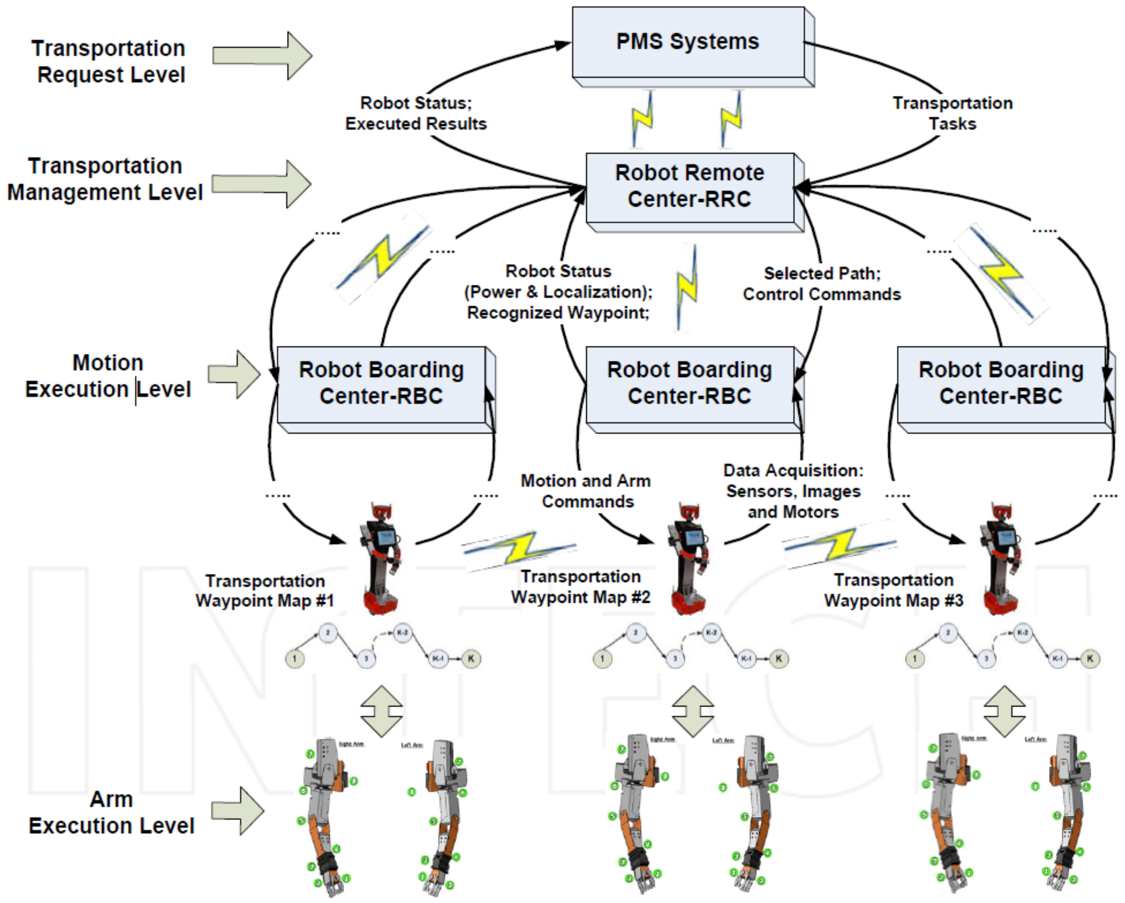

Transportation Request Level
PMS Systems
Robot Status; Executed Results
Transportation Tasks
Transportation Management Level
Robot Remote Center-RRC
Robot Status (Power & Localization); Recognized Waypoint;
Selected Path; Control Commands
Motion Execution Level
Robot Boarding Center-RBC
Robot Boarding Center-RBC
Robot Boarding Center-RBC
Motion and Arm Commands
Data Acquisition: Sensors, Images and Motors
Transportation Waypoint Map #1
Transportation Waypoint Map #2
Transportation Waypoint Map #3
Arm Execution Level

Figure 9. The architecture of the proposed system [56].

2.2.7 Robot Scientist

Benefitting from automation, massive data can be measured in short time. Effectively data analysis becomes the limitation in scientific processes. To solve this problem, a novel system was developed, integrating scientific discovery and robotics, which is named as "Robot Scientist". The "Robot Scientist" not only can automatically originate hypotheses to explain observations, but can also devise experiments and physically run them to test the hypotheses [60], [61]. The hypothesis-generation and experimentation loop of the "Robot Scientist" can be seen in Figure 10.

Focusing on the application domain of yeast functional genomics, R.D. King et al. had designed the robot scientist "Adam" to generate and experimentally test hypotheses about which genes encode locally orphan enzymes. In the hardware aspect, “Adam” can fully automate microbiological experiments, including selecting microbial strains and growth media, and observing the growth of the strain. The software is the core part of “Adam”, which includes many modules: background knowledge, hypothesis generation, and experiment selection, planning experiments, laboratory automation software, analyzing observations and deciding on hypotheses. [62], [63]

For demonstrating the automation in of closed loop learning in drug screening and design, the robot scientist “Eve” had been developed by R.D. King et al. (see Figure 11). “Eve” can screen more than 10000 compounds per day and can be rapidly re-configured to carry out a number of different biological assays. [64], [65]

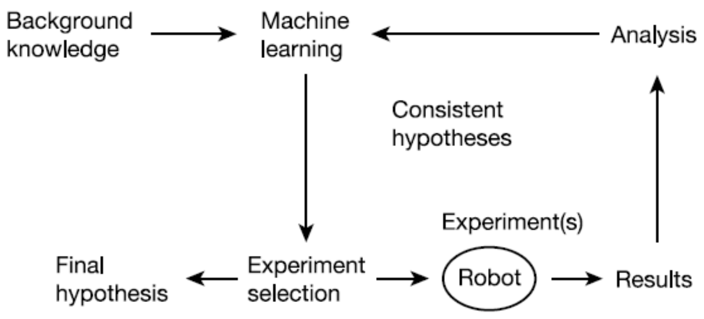


Figure 10. The Robot Scientist hypothesis-generation and experimentation loop [60].

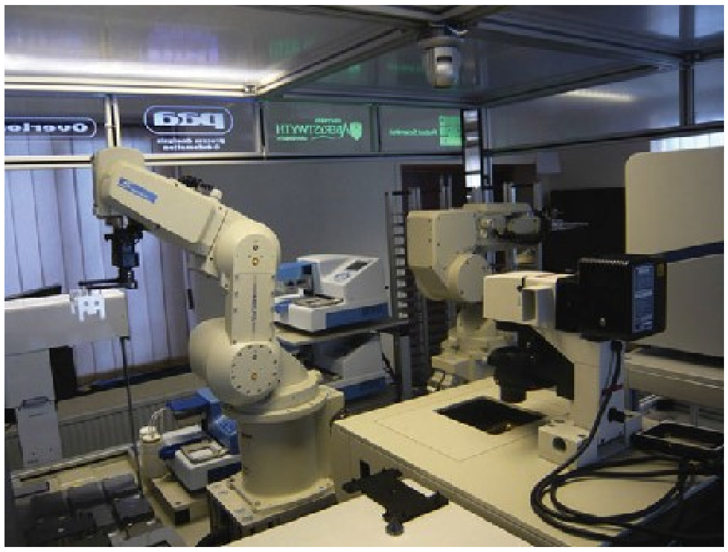


Figure 11. A picture of robot scientist “Eve” [65].

2.2.8 Summary

It can be concluded that automation in life science field will be highly developed with robots playing a critical role. For higher degrees of automation, multiple degrees of freedom robots (robot arms) are required. Even more than one robot arms were mounted in one platform, to meet the demands for high-performance and flexibility. But the cooperation between the robot arms is limited. Human-like dual-arm robots receives many attentions recently due to their flexibility, but are still not widely used in life science automation. The literature review about dual-arm robots is shown in the next section. “Robot scientist”, which is a real complex but powerful system, seems to be the trend of automation development in the field of life sciences.

2.3 Dual-arm Robots and Applications

The dual-arm robot with two robot arms mounted on a torso is somewhat like a human – two arms and one torso. A robot arm consists of several axes (usually six or more) and can move flexibly in the space and easily simulate the motions of the human arm. Not only can do the monotonous, hazardous tasks as a single arm robot does, dual-arm robots are also expected to work together with human workers in the shared working environments [19]. Human-robot cooperation integrates the advantage of humans in intelligence and the advantage of robots in physical agility, and is expected to provide high efficiency, low cost and flexible automation [66]. However, it is challenging to integrating humans and robots in the same workplace. Measures have to be taken to protect human safety, and a robot's appearance or demeanor systematically influences people's perceptions of a robot and their willingness to comply with the robot's instructions [67]. Also, they can be quickly configured to work in the workplace, which is previous built for human works. All these together make dual-arm robots remarkable, drawing many attractions recently.

In this section, several dual-arm robot systems and their applications are introduced. The dual-arm robot systems, which consist of two separate robots, are not involved in this article. In this document, dual-arm robots refer to such robots, which have two arms on one torso and each arm has several axes. At the end is a summary.

2.3.1 Yaskawa Dual-Arm Robot

The Yaskawa SDA (slim dual-arm) series robots are one example of commercial dual-arm robots, which have different versions for different payloads (SDA5 for 5 kg, SDA10 for 10 kg and SDA20 for 20 kg). The SDA10 robot has 15 axes (seven axes for each arm and one rotational axis), a reach of 970 mm and a positional repeatability of ± 0.1 mm, with each arm having a payload of 10 kg. In order to keep the robot arm slim, each axis of the robot arm is compressed, with the motor, the drive, and the encoder and other components incorporated in a small package. Thus, the SDA10 robot can match the flexibility of a human upper body. The two arms can cooperate or perform different tasks independently. The robot can be programed using the INFORM programming language [68], [69].

There are several applications/shows based on Yaskawa dual-arm robots. One interesting application was the robot bartender (Motoman's "RoboBar", see Figure 12(a)), which was based on a standard dual-arm robot and had three available three versions: high production version, entertainment version and non-alcohol version [70]. The dual-arm robot can also be used to form a robotic frozen yogurt kiosk (Figure 12 (b)). The SDA robot could select from our yogurt flavors and ten toppings according the order from the "customer" [71]. Figure 12 (c) shows a robot pancake maker, which automated the whole pancakes making process from mixing the dough to flipping by using the standard kitchen utensils and support customers to make orders by speech [70]. The robotic waiter (Figure 12 (d)) provided a touch-screen display for customers to make orders, and the robot could deliver the food and collect empty dishes [72]. Also, the SDA robot can work as a chair builder to assemble chairs, eliminating human assembly errors and reducing worker injury [71]. Besides, the SDA robot had been used as the core component to form a dual-arm tele-robotic manipulation platform. Compared with typical single arm tele-robotic systems, the dual-arm robot platform can do a broader range of tasks. Based on sensors, the dual-arm tele-robotic platform provided a more natural but less constrained interface, allowing the user to direct robot action through user's gestures, without any mechanical coupling [73].

2.3.2 ABB Dual-Arm Robot

ABB dual-arm robot (named as YuMi or FRIDA) is also a well-known industrial robot, which is considered to be harmless and available to cooperate with humans to perform assembly tasks [74], [75]. ABB dual-arm robot is designed following these demands: (a) minimizing investment cost to enable an acceptable payback time; (b) being fast to program, set-up and reconfigure; (c) being easily relocatable between different stations to address frequent changeovers of production lines; (d) being easily combined with manual assembly in a safe and natural way, without adding safeguards and interlocks that increase engineering and installation effort beyond economic viability [20]. The robot also has 7 axes for each arm. Due to its light weight and small size, the robot can be carried and mounted into workstations easily. The concept of kinetostatic safety field had been introduced to avoid collisions in human-robot interaction [76]. A study about the ABB dual-arm robot and human cooperation for the assembly of a PLC input/output module had been done, with the result showing that the cycle time is shorter than that performed only by robot or only by human [77]. Figure 13 shows the ABB dual-arm robot in the mixed environments with a human.

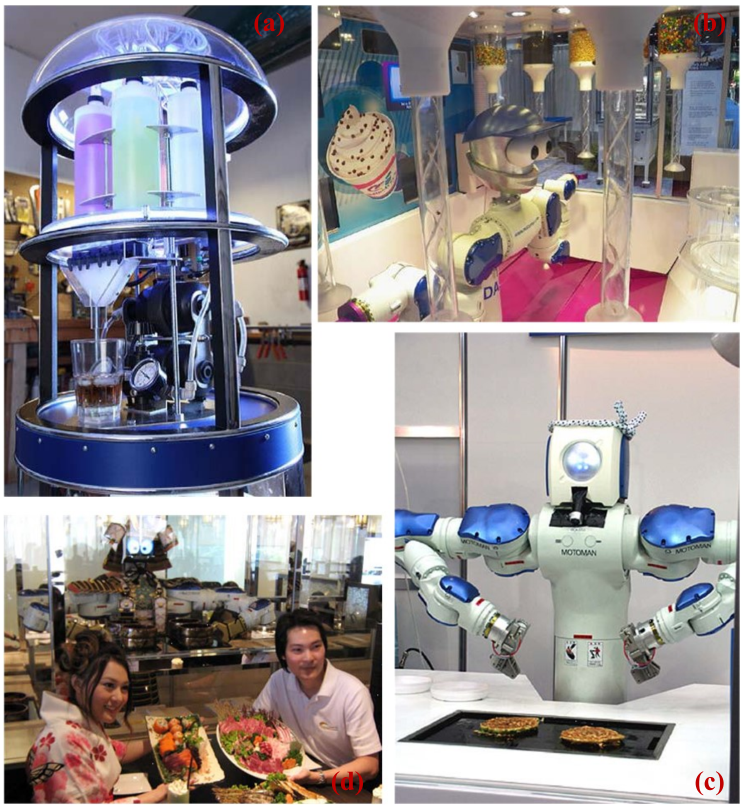


Figure 12. (a) Motoman's RoboBar, (b) Robot operating as a robotic frozen yogurt kiosk, (c) The robot pancake maker, (d) Waiter robot serving meal to customers [70], [71], [72].

2.3.3 ARMAR III Dual-Arm Robot

ARMAR III humanoid robot had been developed for applications in human-centered environments (e.g. kitchen), with most mechatronic modules designed and built at the Institute of Product Development (IPEK) at the University of Karlsruhe (TH) [78]. This humanoid robot had 43 degrees-of freedom (DOF) and consisted of seven subsystems (see Figure 14): head (7 DOFs), left arm (7 DOFs), right arm (7 DOFs), left hand (8 DOFs), right hand (8 DOFs), torso (3 DOFs), and a mobile platform [79]. Due to the multi DOFs, the robot head can also do the humanlike motions. In order to make the robot more acceptable by humans, ARMAR III robot had a similar height to a human – 175mm, and similar motions [80]. The dual-arm grasping and manipulation tasks were achieved using a visual servoing controller [81].

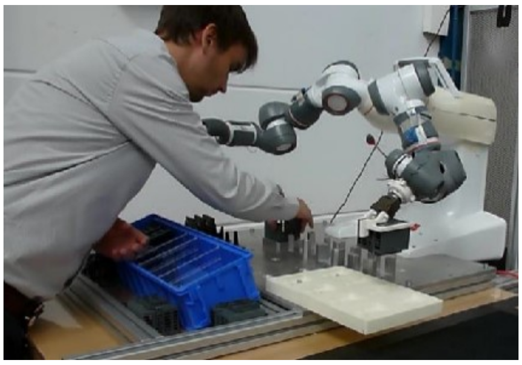


Figure 13. ABB dual-arm robot in mixed environments [77].

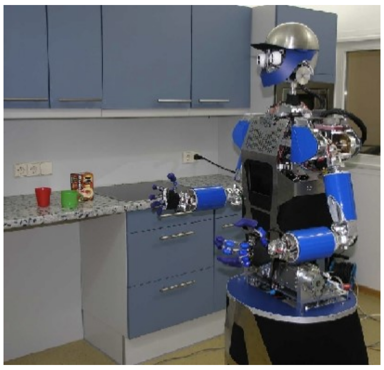


Figure 14. The humanoid robot ARMAR-III [79].

2.3.4 DLR "Justin" Dual-Arm Robot

The "Justin" dual-arm robot developed at the German Aerospace Center (DLR) was used as a research platform for studying dexterous two-handed manipulation (see Figure 15 (a)), with 43 degrees-of freedom [82]. The robot had two flexible arms (7 DOFs for each) [83] and each arm was mounted with a four-finger hand [84]. Later, a mobile platform was designed and integrated with the "Justin" dual-arm robot to study strategies utilizing the increased workspace and redundancy for manipulation tasks (see Figure 15 (b)) [85], [86]. The dynamic reaction and the real-time perception about the robot platform had been studied [87], [88].

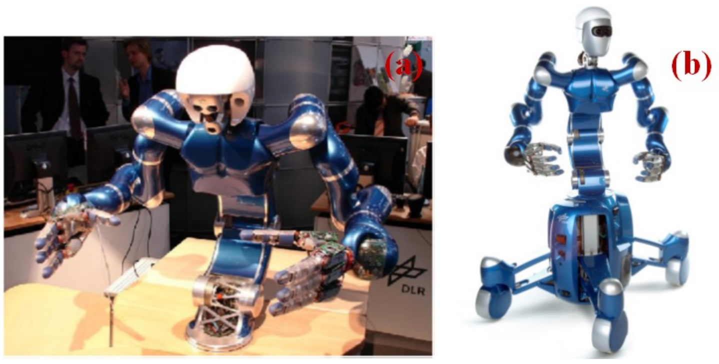


Figure 15. DLR dual-arm robot. (a) upper body [82], (b) robot with a mobile platform [86].

2.3.5 Dual-Arm Robots for Assembly

H.M. Do et al. had designed a dual-arm robot which can be applied to a cell production line for packaging and assembling IT products (e.g. cell phone) [89], [90]. The prototype of the dual-arm robot can be seen in Figure 16. Each arm of the robot had 7 degrees of freedom (DOF) and the waist had 2 DOFs. And a mobile platform is available for the dual-arm robot [91]. The robot had been tested by packaging a cell phone with a successful result [21]. Due to the flexibility of the dual-arm (for conducting complex handling tasks) and the human-size and human-shape (for replace human workers without dramatically changing the structure of the production line), the dual-arm robots were used into an automation system for cellular phone packing processes, excepted to meet the requirements of rapid changes in processes [92]. C. Park et al. had developed a dual-arm robot for the assembly automation of mechanical parts (e.g. gear transmission and constant velocity joints of automobiles), which are not suitable to be done by single arm robots [18]. The robot had two 6-DOF arms and one 2-DOF torso (Figure 17). The performance of the dual-arm robot had been evaluated [93]. In order to make it easy for programming this dual-arm robot, a direct teaching system had been developed [94].

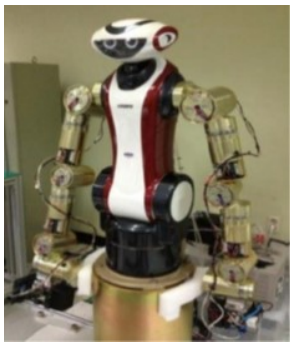


Figure 16. Prototype of the dual-arm robot [21].

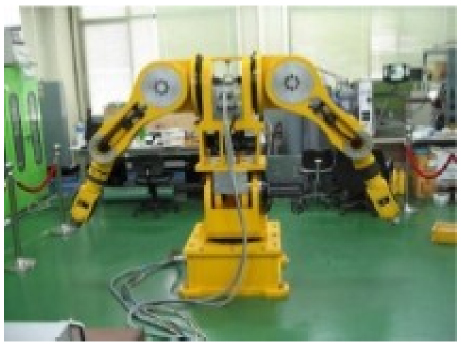


Figure 17. Picture of developed dual arm robot [18].

2.3.6 Dual-Arm Robots for Space Application

Not only needed on ground, robots are also needed to conduct tasks in space. The Robotics Technology Branch at the NASA Johnson Space Center had developed a humanoid robot, known as “Robonaut”, to assist astronauts in space, for performing a variety of tasks (including routine maintenance, setting up and breaking down worksites, assisting crew members while outside of spacecraft, and serving in a rapid response capacity) [95], [96]. As designed to work within existing corridors and use the same tools as astronauts, Robonaut had two dexterous arms and hands [97]. Though Robonaut was the early version, but it demonstrated its ability to work with existing EVA (extravehicular activity) tools and interfaces [98]. By worked together with General Motors, NASA had developed the second generation “Robonaut 2” (or “R2”, see Figure 18), which had a number of significant improvement in electromechanical design, sensing integration, controls strategy, and user interface [99]. “Robonaut 2” was the first humanoid robot sent into space, arrived on the International Space Station in February 2011, and some initial tests had been made[100].

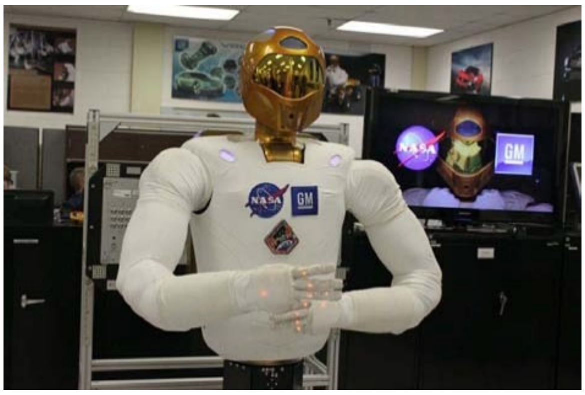


Figure 18. “R2” robot [100].

2.3.7 Dual-arm Robot for Nursing-care and Surgery

Nursing-care assistant robot is strongly demanded in the aging societies. T. Mukai et al. had developed a new prototype of nursing-care assistant robot (named as RIBA, see Figure 19), which had succeeded in transferring a human between a bed and a wheelchair (with 180 kg weight itself, it could lift a patient with a weight of up to 63 kg) [101]. RIBA had two human-type arms for doing various type of lifting motions and had 22 degrees of freedom in total (7 DOFs for each arm, 3 DOFs for the head, 2 DOFs for the waist and a 3-DOF mobile base). As the robot was designed to assist the caregivers, a new human-robot interface had been developed to make the robot easy to operate, named as tactile guidance, which was based on semiconductor pressure sensor arrays and allowed the user to control the robot by touching and leading the robot motions [101]. A whole-body contact manipulation method using the tactile information for human-interactive robots [102] and a dynamic model and an estimating method for estimating the comfortable feeling of humans [103] had been developed. Besides, a dual-arm robot experimental system had been developed by T. Tamei et al., aiming at robotic clothing assistance for the elderly as well as disabled people [104], [105].

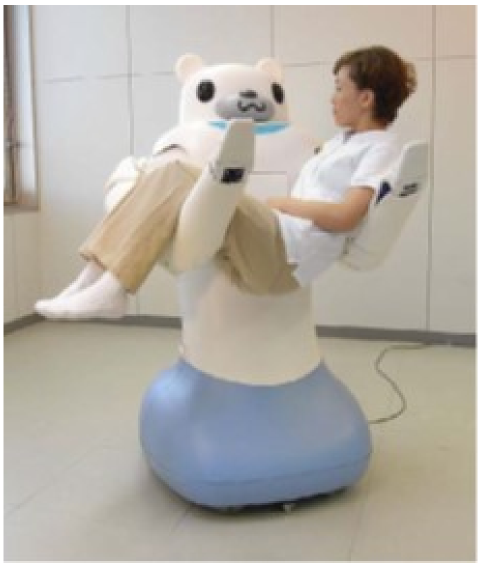


Figure 19. RIBA lifting a human in its arms [101].

Beside for assisting nursing-care, dual-arm robots are also a helpful tool for surgery. In minimally invasive surgery, dual-arm robots are beneficial tools, having the potential to overcome many existing problems (e.g. the loss of direct visual feedback) [106], [107]. Also dual-arm robot will be available to provide tele surgery to astronauts during the long duration space missions [108].

2.3.8 Summary

Recently, dual-arm robots draw many attractions. Firstly, these robots can work in a very similar way to a human due to a structural similarity and they are expected to resemble a human-like behavior at kinematic level, in order to reach the same level of dexterity of humans. So, they are, in principle, able to cooperate with humans in the same way as humans co-operate with humans. And the cooperation between human and robot workers is thought to be the key to low-cost, flexible automation, by taking the advantage of human's intelligence and the advantage of robot's physical power. Secondly, human-like robots are expected to replace human workers without major redesigns of the workplace. Thirdly, human-like dual-arm robots are expected to be quickly reconfigurable to produce a new batch of a completely different product or sub-module. Human-like dual-arm robots open up new application possibilities. Many dual-arm robots are developed or still under developing. Several remarkable dual-arm robot systems, developed for different application fields, have been reviewed in the dissertation. The main features of these dual-arm robots are summarized in Table 1, with their aimed application field or potential applications given.

Table 1. Features of dual-armed robotic systems.

Dual-arm Robot	Mobile base	DOF per arm	End-effector	Application
SDA	no	7	Specialized tool	assembly and food industry
YuMi	no	7	Gripper	cooperating with human for assembly
ARMAR III	yes	7	five-finger hand	applications in human-centered environments
Justin	yes	7	four-finger hand	studying dexterous two-handed manipulation
Robot (by H.M. Do et al.)	yes	7	Specialized tool	cell production line
Robot (by C. Park et al.)	no	6	Specialized tool	assembly of mechanical parts
Robonaut 2	no	7	five-finger hand	assist astronauts in space
RIBA	yes	7	-	transferring patients

2.4 Summary

Research activities in life science have been rising in recent years, and robots play a vital and growing role is life science automated systems. It can be said that the future bio analytical laboratory will be highly automated. In this case, multi-DOF (degree of freedom) robots will be widely used to meet the demands for flexibility, as such robots can offer a wide range of flexibility in motion planning, due to having more DOFs than those strictly necessary to perform a certain task. Even, robot systems, consisting of two separate multi-DOF robots, were designed for the demands for higher performance and more flexibility. Recently, human-like dual-arm robots also draw many attractions. Firstly, these robots can work in a very similar way to a human due to a structural similarity. So, they are, in principle, able to cooperate with humans in the same way as humans co-operate with humans, which is thought to be the key to low-cost, flexible automation. Secondly, human-like robots are expected to replace human workers without major redesigns of the workplace. Thirdly, human-like dual-arm robots are expected to be quickly reconfigurable to produce a new batch of a completely different product or sub-module. Human-like dual-arm robots open up new application possibilities. However, the application of dual-arm robot in life science automation is still limit.

The application of dual-arm robot will bring three advantages. Firstly, tools can be saved, as the robot can use one arm hold the tool, and the other one to process it. Secondly, the processes for preparing samples can be transferred from the manual way to an automatic procedure without major changes, as the robot can use the same tools as the human operator uses. Thirdly, it is flexible to integrate the other instruments in to the platform by the robot.

Main advantage of a dual-arm robot in LSA: in classical automation robot is only used as a system integrator. All sample treatment (pipetting, shaking, opening etc.) has to be done by separate units. These units are expensive. The use of a dual-arm robot allows the using of conventional laboratory equipment and thus reduces costs. Beside that it allows the 1:1 automation of manual processes without changes in the workflow. This is especially of importance for regulated processes in pharmaceutical, medical or food industries.

Chapter 3 General Description of Life Science Processes

Life sciences are closely related to our lives and helpful in providing solutions for solving a series of major problems, such as food shortages, energy crisis and disease hazards. Life sciences belong to experimental sciences and involve a wide range of research objects, e.g. microbiology, bio macromolecule, plants, animals, human beings and the ecosystem. Sample preparation and analysis are important procedures in life sciences, as sample data can be obtained for supporting researches. The general processes involved in this dissertation will be discussed in the following parts of this chapter.

3.1 Sample Preparation

Before analyzing, the raw materials (e.g. solutions or powders) should be transformed to a measurable form (called samples) corresponding to the analytical technologies. The important procedures of sample preparation include:

a) **Dissolution**. Samples to be analyzed by some analytical instruments (e.g. LC-MS) are required to be liquid form. So, powders need to be dissolved with water or other liquids. By dissolving powers with certain weights into liquids with certain volumes, solutions with known concentrations can be obtained. For the determination of chiral compounds, auxiliary solution needs to be prepared by dissolving Nα-(2,4-Dinitro-5-fluorophenyl)-L-valinamide (called L-FDVA for short) and Nα-(5-Fluoro-2,4-dinitrophenyl)-D-leucinamide (called D-FDLA for short) powders. 7.50 mg L-FDVA powder will be dissolved by 10 mL acetone. And the resulting solution will be used to dissolve 7.85 mg D-FDLA powder. As the molecular weights of L-FDVA and D-FDLA are approximately 300 and 314 separately, so in the resulting solution the concentrations of both of them are approximately 2.5 mmol/L. For the determination of cholesterol in the biliary endoprosthesis, the incrustation will be obtained from biliary endoprosthesis samples and milled to powders. 1 mg of resulting sample powder will be weighed and dissolved by 1,000 µL hexane. Thus, the concentration of the sample powder is 1,000 mg/L in the resulting solution. And the concentration of cholesterol in the resulting solution will be analyzed.

b) **Distribution**. In the contrast experiments or for making calibrations, solutions with the same components but different ratios maybe required. So, it needs to distribute liquids to different containers with different volumes. E.g., for calculating the enantiomeric excess of chiral compounds, calibration solutions mixing with L-proline (L) and D-proline (D) solutions will be prepared in five empty glass vials with different volume ratios (L: D): 2,000 µL: 0 µL (100ee%), 1,500 µL: 500 µL (50ee%), 1,000 µL: 1,000 µL (0ee%), 500 µL: 1,500 µL (-50ee%) and 0 µL: 2,000 µL (-100ee%). For calculating concentration of cholesterol in biliary endoprosthesis samples, calibration solutions with the concentration of cholesterol ranging from 500 µg/L to 2,500µg/L will be prepared.

c) **Dilution**. Firstly, it needs to dilute solutions into certain concentrations. The solutions with too high concentrations will not be suitable for analysis. Secondly, some solutions have different properties depending upon its concentration, e.g. sulfuric acid. Concentrated sulfuric acid reacts with metals in a different way with diluted sulfuric acid. The source amino acid solution has a higher concentration of 50 mmol/L, which will be diluted to 1 mmol/L in the process of preparing samples by pipetting 5,880 μL water and 120 μL amino acid into a glass vial. In the final dilution of chiral compound samples, 10 μL resulting sample solution and 490 μL methanol will be pipetted into each well of a new microplate in order to dilute the concentration of the sample. In the final dilution of resulting cholesterol samples, 100 μL filtered sample solution, 50 μL 5α-cholestane solution and 850 μL hexane will be pipetted into a GC vial.

d) **Derivatization**. Derivatization is a chemical technology, which is suitable in analysis of mixtures. In a process of derivatization, a chemical compound will be transformed to a new compound (derivation). Derivatization can be used to promote analysis, especially in the determination of chiral compounds. Though chiral compounds have similar physical and chemical characteristics, but their derivations can be different and thus easily classified. In an experiment, 100 μL auxiliary solution containing L-FDVA and D-FDLA, 50 μL sample solution containing L-proline and D-proline, and 20 μL NH_4HCO_3 solution will be pipetted to each well of a microplate. NH_4HCO_3 solution is used to promote the reaction between the auxiliary solution and the sample solution. Then the derivatization will be carried out under shaking in a thermo shaker.

e) **Extraction**. Chemical extraction is a physical separation process, separating a certain material from a matrix without changing its chemical composition. Ultrasonic can be used to promote extractions. For extracting cholesterol from the sample powder, the sample powder is dissolved by hexane, and then the resulting solution is treated by shaking and ultrasonic.

f) **Filtration**. Filtration is important in chemistry for purifying materials. Before filtering, the mixture is dissolved into a certain solvent, with one component dissolved while the others not. Thus, the component can pass through the filter, and the others will be filtered. For the determination of cholesterol, the extracted solution will be filtered by using syringes. First, load a cannula to a syringe and draw certain extracted solution into the syringe. Then, change the cannula to a filter, and filter the drawn solution into a clean GC vial.

3.2 Sample Analysis

Elements or compounds can be identified according to certain chemical functional or structural properties. Classical analysis methods include qualitative analysis which is to identify the chemical compositions of a sample, and quantitative analysis which is to determine the amount of the components. In the classical methods, materials can be separated by precipitation, extraction or distillation, and identified based on the differences in color, smell or reactivity. The changes in mass or volume can be used to quantify the amount. Nowadays, modern instruments are more widely used in chemical analysis, with the qualitative analysis and quantitative analysis both performed in instruments, e.g. liquid chromatography–mass spectrometer [109].

Chromatography is a technique for separating mixtures. When the mobile phase carries a mixture through the stationary phase, the components of the mixture will be separated, due to their different travelling speeds in the two phases. Thus, chromatography can be used to determine the relative properties of components in a mixture. According to the physical state of mobile phase, it is divided into the gas chromatography with the mobile phase being a gas and the liquid chromatography with the mobile phase being a liquid. Gas chromatography is suitable for materials which are easy to gasify and stable under high temperature. It is not suitable to analyze proteins, as the high temperature used in gas chromatography can cause thermal denaturation.

A common way is using the chromatographic techniques in tandem with mass spectrographic techniques, e.g. gas chromatography-mass spectrometry and liquid chromatography-mass spectrometry. The chromatographic techniques are used to separate compounds. Mass spectrometer is an technology that identifies the chemical composition of a compound or sample based on the mass-to-charge ratio of charged particles [110], which can be used in structural analysis [111]. The mass spectrographic techniques can also be combined with the inductively coupled plasma technique, which has high precision and sensitivity in detecting metals [112]. Elemental analysis is to analyze the elements of a sample, e.g. waste, soil or chemical compounds, for determining what elements are contained and how much of each element is contained.

3.2.1 Elemental Analysis

Metal ions are important to lives, as it was said that "about one-third of all proteins require a metal ion for their structure or function" [113]. Elemental analysis is to analyze a sample to determine its elemental composition. One key technology in elemental analysis is the inductively coupled plasma – mass spectrometry (ICP-MS), which is powerful, mature and with multi-elemental ultra-trace detection capability [114]. Due to its advantages, ICP-MS has practically replaced other techniques, e.g. atomic emission spectroscopy, X-ray fluorescence and instrumental neutron activation analysis [115].

Calcification can cause stent fracture with high ratio of incidence [116], [117]. Thus, it is necessary to do research about the changes of calcium in stents. In our research, ICP-MS instrument is used for the determination of the content of calcium in stents. Stent samples will be prepared and cut open to collect the attachments from their surfaces. Then, certain amount of the attachments will be weighed and dissolved with HNO_3 solution. The final solution samples will be analyzed by inductively coupled plasma – mass spectrometer instrument to determine the content of calcium. In the introduction of the sample, the sample liquid will be converted into an aerosol by nebulizer, and then ionized with inductively coupled plasma. The resulting ions will be extracted in the mass spectrometer, and separated by their mass-to-charge ratios. Finally, the ions will be received by a detector, with the output signals proportional to their contents.

3.2.2 Structural Analysis

Mass spectrometry technique can not only be used for determining the elemental composition of a molecule, but also can be used to analyze the structures of molecules. An empirical formula cannot present the arrangement of atoms, as different compounds are possible to have the same empirical formula. In comparison, a structural formula is able to show more information of a chemical compound, by providing the geometric representation of a molecular structure. Thus, for chemical compounds having the same chemical formula, it is available to use their structural formulas to show their differences. In an analyzing process, original molecules in samples can be converted into molecular fragments or charged molecules by an ion source. Then the resulting ions will be observed by a detector to determine the structure of a molecule. The observed data is commonly represented by the mass spectrum. And related compounds can be identified by comparing the resulting mass spectrum against a library of mass spectrum.

In our applications, the gas chromatography-mass spectrometry technique is used to determine the cholesterol in the biliary endoprosthesis, with the gas chromatography used for separating different compounds and the mass spectrometry used to analyze and detect the generated fragments. The liquid chromatography–mass spectrometry is used for the determination of chiral compounds, which use a different ion source to ionize molecules – it is electrospray ionization source (EIS). EIS is called as soft ionization technique, as there is very little fragmentation when ionizing molecules. Besides using mass spectrometry technique, the structure information of molecules can also be provided by nuclear magnetic resonance spectrum and infrared spectroscopy [118].

3.3 Implementation

3.3.1 Determination of Chiral Compounds

Chiral compounds are closely related with our daily life, with many pharmaceutically active ingredients being chiral compounds [119]. Chiral compounds have the same molecular mass, identical molecular formula, and approximately the same physical properties and chemical characteristics. One of the differences between chiral compounds is the performance of rotating plane-polarized light, with each enantiomer rotating the light in a different sense (clockwise or counterclockwise). However, while one compound is helpful to us, its enantiomer may be hazardous [120]. Thus, it is necessary to determination of the enantiomeric excess of chiral compounds, which is challenging. Classical measurement techniques usually require relative long analysis times.

A high-throughput measurement method had been developed without a time consuming chromatographic separation, which enables analysis times of 30 s using mass spectrometry [121], [122]. Though chiral compounds have the similar physical properties and chemical characteristics, but they can be discriminated by using their derivations, which are produced based on the different reaction kinetics between them and suitable auxiliary solutions. A parallel kinetic resolution method has been used to perform the derivatization of the chiral substrates, and its general principal is shown in Figure 20. In this derivatization process, there will generate four reaction products. As the auxiliaries have different molecular masses, the four derivations have two characteristic molecular masses. And as the reaction speeds between chiral compounds and auxiliary solutions are different, the ratio of intensities of the two characteristic compounds (the four derivations with two characteristic molecular masses) is different, which can be detected by mass spectrometry. In order to calculate the enantiomeric excess of samples with unknown chiral compositions, a calibration curve will be created by using groups of the chiral compounds with the known but different molar ratios.

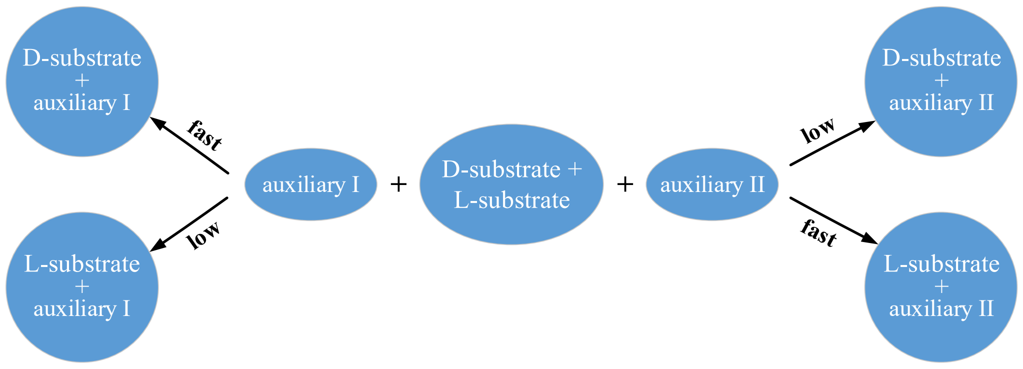

D-substrate + auxiliary I
fast
L-substrate + auxiliary I
low
auxiliary I
+
D-substrate + L-substrate
+
auxiliary II
low
D-substrate + auxiliary II
fast
L-substrate + auxiliary II

Figure 20. The principle of parallel kinetic resolution method.

After source materials, e.g. powders and liquids, are provided to the robot platform, the robot will follow a designed process to prepare samples automatically, which mainly includes dissolving powders, diluting solutions, distributing solutions, shaking solutions, preparing calibration solutions and transporting the final solutions to an Agilent LC-MS instrument for determining the chiral compounds.

3.3.2 Determination of Cholesterol

Cholesterol commonly exists in the cell membrane and is essential for animal life. The changes in cholesterol have a role in many diseases [123]. Gallstones are a common disease, with a mean prevalence rates of 10-12% in European populations [124] and cholesterol gallstones accounting for 80-90% of the gallstones [125]. The tendency to clog limits the use of biliary endoprosthesis, and it was pointed that the deposit ultimately leading to the occlusion was found mainly to contain bacteria, yeast cells, and plant fibers (food debris) [126]. In addition, cholesterol had also been determined in the obstruent blocking biliary endoprosthesis [127]. In order to understanding more about the role of cholesterol in blocking biliary endoprosthesis, an analysis is required.

For extracting cholesterol, the biliary endoprosthesis samples will be cut into small pieces. The incrustation is obtained and dried at room temperature to constant weight and then milled. Hexane is used to extracting cholesterol from the samples under ultrasonic treatment. Then the samples will be filtered to get pure sample solutions. In order to adjust the loss of analyte during sample preparation or inlet, an internal standard is needed. As 5α-cholestane is similar to cholesterol, it is chosen as the internal standard, which will be added to samples, the blank and calibration standards in a constant amount. Commercial cholesterol (≥ 99%) is used as the calibration standards to obtain the analyte concentrations. The samples will be analyzed by the GC-MS instrument.

After source materials, e.g. powders and liquids, are provided to the robot platform, the robot will follow a designed process to prepare samples automatically, which mainly includes dissolving powders, diluting solutions, distributing solutions, shaking solutions, ultrasonic treatment, filtering solutions, preparing calibration solutions and transporting the final solutions to an Agilent GC-MS instrument for determining the cholesterol.

Chapter 4 Aims of this Dissertation

4.1 Strategy

In order to make this dual-arm robot platform flexible, easy to use and with high degree of automation, and benefits to sample preparations in life science automations, there are five main aims having been conceived, as following:

a) **Copy processes.** As the raw materials should be transformed to a measurable form before analyzing, processes for transforming such materials are required. Corresponding to the analytical technologies and the materials themselves, the processes of preparing samples should be designed specifically. Normally, such a process is firstly designed as a manual procedure, and then the manual one is transformed to an automated one after it is verified, evaluated and improved. However, when automating a manual process, problems are frequently happened. As the automated tools are different with the manual ones, a process may have to be re-designed. Even worse is, a process cannot be automated due to some unavailable automated tools. If the manual tools used in a manual process can also be used in the automated process, such problems will be avoided. As they are using the same tools, so it is feasible that permitting an automated process which is almost the same with the manual one. In this way, it can be also expected that the prepared samples by the two processes will be the same. Thus, a manual process is copied to an automated one.

b) **Common tools.** The robot uses the same tools as that used by human users. Firstly, by using the same tools, the manual processes can be copied to automated ones. Secondly, therefore the design of special automated tools can be reduced. The manual tools are normally designed to fit our hands, but they may not fit the robot hands. We have five fingers per hand, but the most commercial automated grippers have only two fingers. Some adapters and racks are needed to make it easier for a robot to handle them, which can be easily printed by 3D printers.

c) **Reconfiguration.** The platform is able to be re-configured easily and quickly. In the life sciences, sample preparations are essential tasks. An automated platform, which can be re-configured to prepare different kinds of samples and prepare samples with different parameters, will be useful.

d) **Integration.** In order to obtain high degree of automation, analytical instruments have been integrated into the robot platform. After samples are prepared, they will also be transported to analytical instruments automatically and then measurements are started automatically. The user only needs to feel raw materials and some consumables to automated platform, set parameters to the instruments and design the process of sample preparation, and then measured data can be obtained.

e) **User-friendly.** Finally, the automated platform should be easy to use. In the plan, a dual-arm robot is used as the kernel of the platform. Not all users have the skills of robot programming, but a user-friendly has be given for operating the platform.

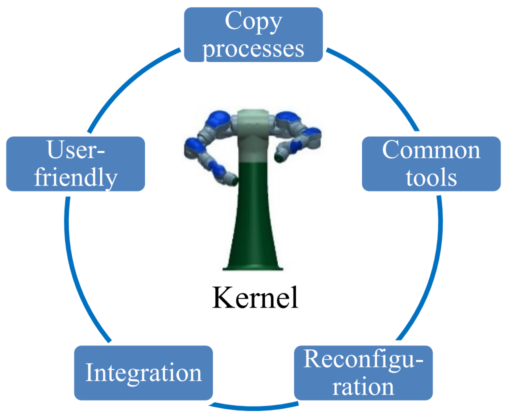


Figure 21. Aims of the automation platform.

4.2 Formulation

The Yaskawa dual-arm robot (the CSDA10F model) is used as the key role in the automated platform, as shown in Figure 22. As the robot has two robot arms, with each arm having redundant degrees of freedom for moving flexibly in the 3D space, the robot can do abundant motions and also can simulate human motions. This permits that the robot is able to use the manual tools, e.g. the pipettes and syringes. The robot can use one arm to hold a tool and the other one to process it.

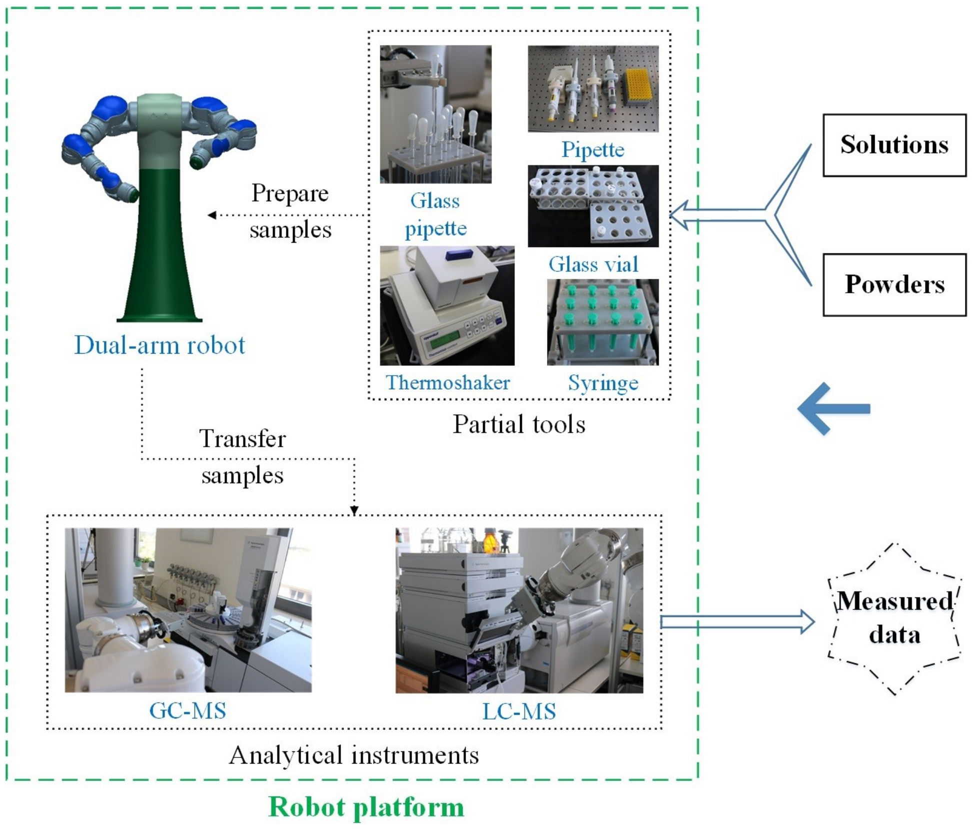


Figure 22. Concept of automation system.

System integration, by integrating existing hardware and software tools, is a fast way for laboratory automation. In the hardware aspect, besides the LC-MS, which is very useful due to its very high sensitivity and selectivity for analyzing samples, a gas chromatography–mass spectrometry (GC-MS) instrument has also been integrated into the platform for analyzing different kinds of samples. In order to reduce the requirements of programming skills of the operator and also to start the sample analysis process automatically after the samples are transported to analytical instrument, a user-friendly software is required to manage the whole platform [128]. The software of the robot, the software of LC-MS instrument and the software of GC-MS instrument have been integrated with the SAMI Workstation EX Software system (SAMI EX, Beckman Coulter Inc.), which incorporates optimized planning and data-driven dynamic rescheduling for pre validated schedules and run-time flexibility and uses graphical interface to make assay design easier [129]. Thus, the processes, including sample preparation, transport and analysis, can be automated and scheduled by SAMI software. And the user can design the processes through the friendly graphical interface of SAMI software.

4.2.1 Robot Programming

For preparing samples, the robot will use several different lab ware and devices, including pipette, syringe, microplate, vial, thermo shaker, ultrasonic machine and so on. To realize the robot motions for using the lab ware and devices, a series of robot programs (robot jobs) need to be built. Some of the robot jobs have been taught using a teaching pendant, while the others are based on calculations. The challenging of programming for the robot comes into three parts: 1) different lab ware and devices need to be used by the robot; 2) the processes of sample preparing is complex and the platform should be flexible to be quickly re-configured for preparing different samples; 3) the robot is located at narrow space, surrounded by analytical instruments and the shelf for hanging lab ware. For solving these problems, robot motions are divided into many small units. A motion unit is realized by a robot program. For each labware, there are several robot motion units built for handling it. In order to make it easier to manage these motion units, they are separated to different groups according to the lab ware (see Figure 23).

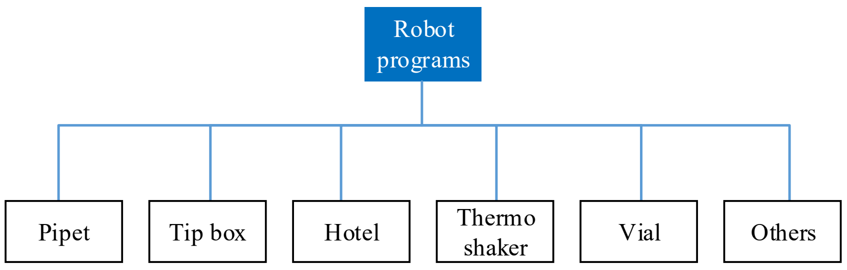


Figure 23. Groups of robot programs.

A motion unit has three properties:

1) **Relative independence**. A motion unit is relatively independent with the other units. So, when a motion unit is modified, replaced or removed or adding complete new motion units, the other motion units will remain unchanged. In this way, for adding new tools, new robot programs can be created without affecting the previous robot programs.

2) **Connection**. A motion unit is not completely isolated. Instead, it connects with several other units. When performing a task, several units will be combined in certain ways according to the content of the task. When there are more motion units, there are more ways of combining motion units. So, the robot can do more different tasks.

3) **Path**. There is a moving path of each motion unit: a fixed moving path or a changeable one.

4.2.2 Integration of Analytical Instruments

Besides the automation of sample preparation, the automated sample transport and sample analysis are also necessary. So, after the user provide raw materials (solutions and powders) manually, the measured data will be provided automatically, without the involvements of users in the whole process. The LC-MS and the GC-MS instruments have been integrated into the platform. The integration of analytical instruments includes two parts: hardware integration and software integration. The direct integration of analytical systems such as LC-MS is a challenging task since analytical systems are usually closed systems without any general interfaces for automated sample delivery from other automation systems or robots. The flexile multi-DOF robot arm can solve this problem. The challenge of the software integration comes from that the lack of available APIs (Application Programming Interface). For solving the problem, the software integration was achieved by simulating mouse and keyboard inputting to the graphical user interface of the analytical instrument's software.

4.2.3 User-friendly Interface

In order to make the platform easy to use, even for the user who has no knowledge about robot programming, a user-friendly software is integrated to manage the whole platform – it is SAMI Workstation EX Software. Through the SAMI software, the user can design the processes of sample preparation. And then the robot will follow the designed process to prepare and transport samples. The SAMI software controls the analytical instruments and the robot:

1) **Control instruments**. Interface programs has been written separately for integrating the LC-MS and GC-MS instruments with SAMI software. SAMI software will send commands to the interface program, and then the interface program will control the instruments through APIs or operating the GUI. Finally, SAMI software can start or stop the measurement automatically.

2) **Control robot**. Another interface program has been written for integrating the robot and SAMI software. A lot of robot programs have been built for realizing the basic robot motion units and they should be combined in certain ways for conducting different tasks. After commands from SAMI software is received, the interface program will combine the related motion units in correct ways. For different tasks, different motion units will be combined.

Chapter 5 System Overview

5.1 Robot Platform

The overview of the platform is given in Figure 24 (a). The CSDA10F dual-arm robot has 15 degrees of freedom: 7 axes for each arm and 1 rotation-axis, and has a high positional repeatability (±0.1mm). As the robot has two arms, it can use lab ware in a human-like way with one arm holding the lab ware and the other one processing it. This gives the robot the feasibility and flexibility to deal with difficult tasks, including preparing and transporting samples. And as the robot uses the same lab ware as the human operator uses, this makes it easier to transfer the manual process to an automatic one. For preparing samples, the lab ware, including pipette, glass pipette, glass vial, 96-well microplate, liquid container, tip box, syringe, needle, filter, a small shaker, an ultrasonic machine and a thermo shaker, will be used. More details about the lab ware will be introduced in the following part of this chapter.

After the samples are prepared, they will be transported to the analytical instruments for analysis. In this platform, a LC-MS instrument and a GC-MS instrument have been chosen to analysis samples, due to their very high sensitivity and selectivity. More details about the platform can be seen in Figure 24. Around the robot, there is the shelf for hanging lab ware, including pipettes and tip boxes, and the deck for laying out the lab ware required for the robot to preparing samples. Due to the big strength and high speed of movement, safety is vital. Light curtains have been used to detect if the body of the operator was in the work area of the robot. The robot will also be stopped immediately when the operator opens the door (marked as (7) in Figure 24 (b)) to get into the platform.

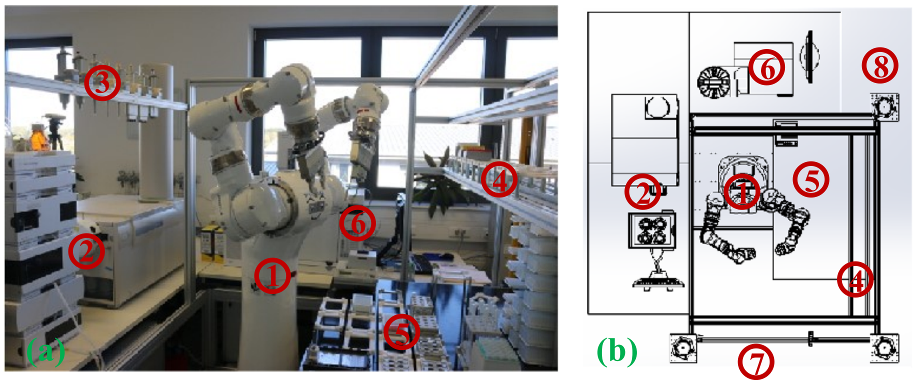


Figure 24. Picture of the platform (a): CSDA10F dual-arm robot (1), LC-MS (2), pipettes hanging on profile shelf (3), ALPs on profile shell for placing tip boxes (4), deck for laying out lab ware (5) and GC-MS (6). Schematic of the platform (b): the door (7) and light curtain (8).

Figure 25 shows the layout of the deck. While some lab ware has its fixed position on the deck, the position of other lab ware is changeable. This is achieved by the ALP, which can be placed with different lab ware, including tip box, vial rack, vial cap rack, liquid container, microplate, microplate lid, syringe rack, syringe filter, syringe needle and glass pipette rack. In this case, it is flexible to layout the lab ware for preparing different kinds of samples.

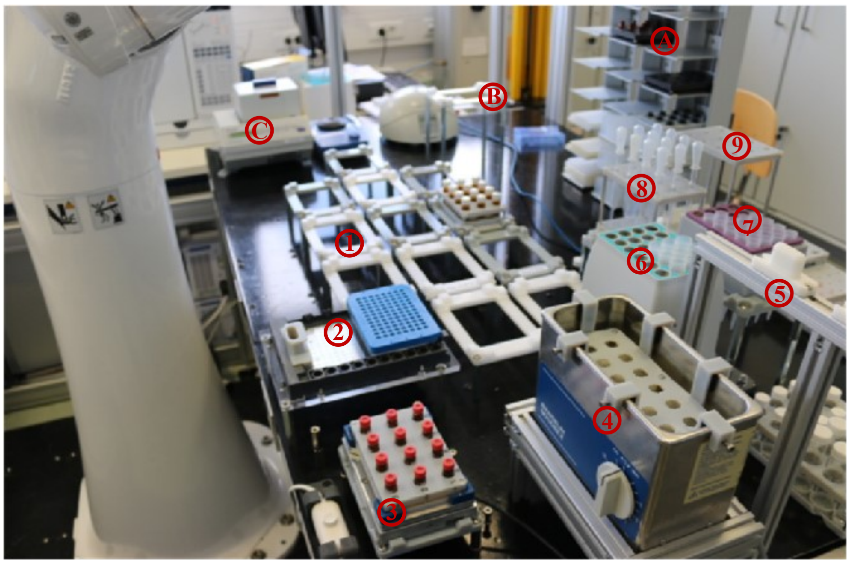


Figure 25. Layout of the deck. ALP (1), auto sampler tray (2), small shaker (3), ultrasonic machine (4), temporary position for hanging pipette (5), 10mL tip box (6), 5mL tip box (7), glass pipette rack (8), rack for placing waste pipette (9), hotels for storing microplates and vial rack (A), re-gripping ALP (B) and thermo shaker (C).

5.2 Robot Grippers

For gripping the lab ware, grippers are required. At the end of each arm, a gripper (SMC LEHF series) with two fingers is mounted. These fingers are made using 3D print technology, which is flexible in manufacturing the required design of the fingers. As shown in Figure 26, the grippers at the left arm and the right arm are designed for different usage. The gripper on the right arm is used for gripping pipettes, glass pipettes, vials, syringes, vial cap, thermo shaker lid and so on and pressing buttons. The gripper on the left arm is used for gripping microplate, tip boxes, vial rack, syringe rack and needle rack, squeezing the rubber head of glass pipette and holding vials.

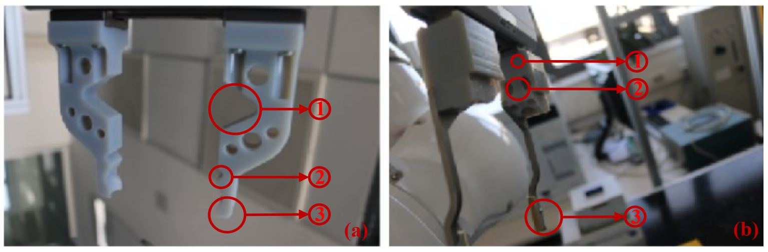


Figure 26. Fingers of right arm (a), usages: handling pipettes and other lab ware with larger volume (1), handling glass pipettes (2), handling vials and other lab ware with smaller volume and pressing buttons on instruments (3). Fingers of left arm (b), usages: squeezing the rubber head of glass pipette (1), holding vials (2), gripping microplates, tip boxes and racks (3).

Some examples of the grippers gripping lab ware can be seen in Figure 27. In (a), the left arm gripper holds the vial, and the right one grips the vial cap to open it. In (b), the right gripper is pressing the button of thermo shaker. In (c), the right arm is removing the lid of the thermo shaker. For using the pipette, the right gripper will hold the pipette and the left gripper presses the button, (see (d)). It works in the similar way for using glass pipette: the right gripper holds the glass pipette and the left one squeezes its rubber head, which can be seen in (e). While for sucking liquid using syringe, the left gripper holds the syringe and the right one drags its plug, see (f).

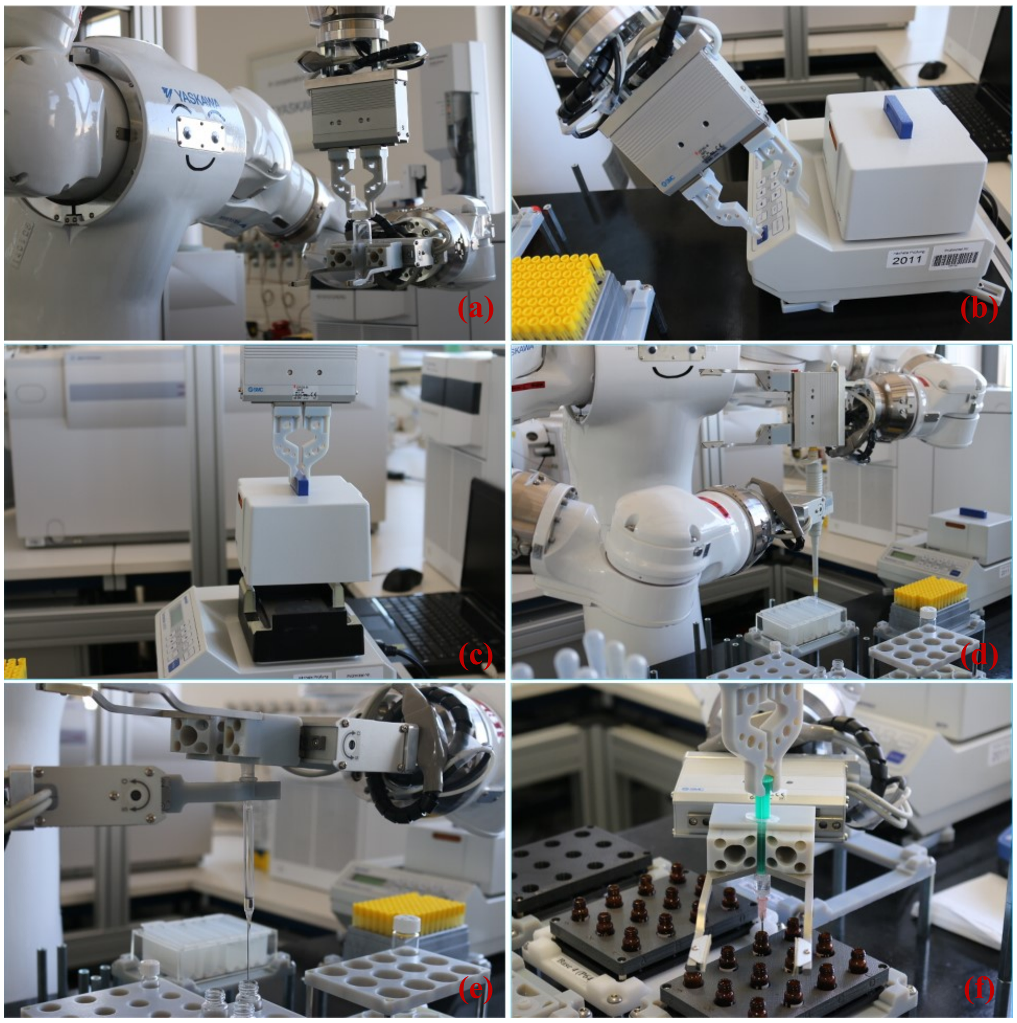


Figure 27. Handling lab ware.

For gripping an object, the gripper fingers will move to a certain position. This position should be configured first. Table 2 gives the main configuration of grippers for gripping different lab ware or open/close gripper and relative movements.

Table 2. Gripper position configuration.

Right gripper		Left gripper	
Object/task	Position (unit: mm)	Object/task	Position (unit: mm)
close	1.0	close	1.0
open	48.0	open	48.0
4mL vial-body	6.1	22mL vial-body	16.6
4mL vial-cap	7.7	10mL vial-body	11.7
10mL vial-body	10.8	4mL vial-body	7.2
10mL vial-cap	9.3	Microplate	22.9

22mL vial-body	15.8	Microplate lid	44.2
22mL vial-cap	15.5	Microplate (wider)	35.0
glass pipette	1.6	Tip box	39.1
glass pipette (wider)	4.3	Squeeze GP head	2.0
pipette	24.6	Release GP head	8.5
pipette (wider)	27.0	Relative (+)	0.3
thermo shaker lid	3.4	Relative (-)	-0.3
Relative (+)	0.3		
Relative (-)	-0.3		

5.3 Lab Ware

The pipette is one of the critical tools in life sciences for pipetting liquid. In the platform, eight types of pipettes are supported currently, and for these pipettes five types of tip boxes will be needed (see Figure 28). The pipettes are hung on the shelf. Some of tip boxes have a fixed position on the deck, while the other ones are placed on the ALPs, which are mounted on the shelf. When pipetting solution, if needed, the robot will transfer the tip box to the ALP on the deck first, and then pick the selected pipette for pipetting the liquid.

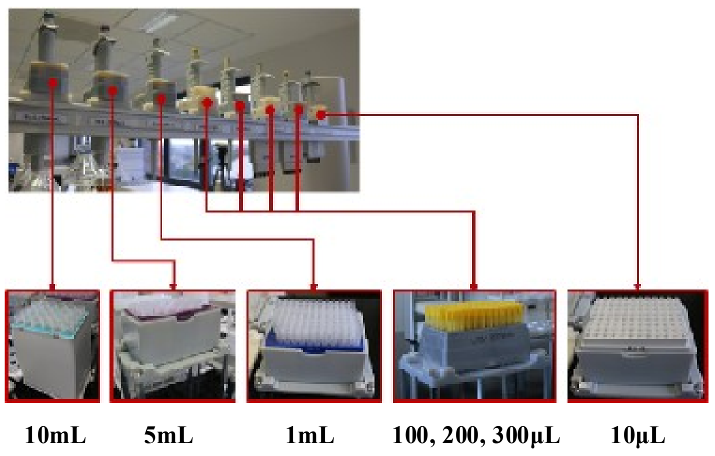


Figure 28. Pipettes and tip boxes.

Glass vials are also important for preparing samples. They can be used to mix different liquids, to dissolve powders and as sample carriers when analyzing with the GC-MS instrument. In our platform, four types of glass vials with different volumes are supported: 2 mL, 4 mL, 10 mL and 22 mL (see Figure 29). When preparing samples, the vials are placed in the rack. Different racks are needed for different vials. There are also racks for placing vial caps. For some glass vials, it is infeasible to transfer liquids using a pipette. In this case, glass pipette will be used. Figure 30 shows the rack for placing glass pipette. When pipetting, the robot will pick the selected glass pipette from rack (1). After pipetting, the glass pipette will be put to rack (2) for reclaiming.

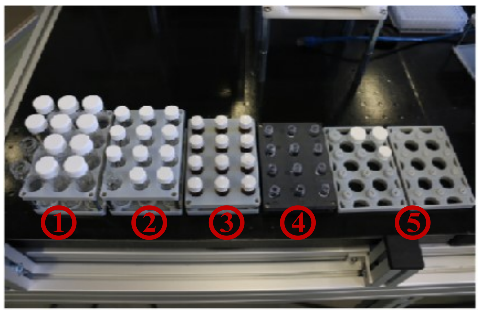


Figure 29. Vial rack. 22mL vial rack (1), 10mL vial rack (2), 4mL vial rack (3), 2mL vial rack (4), and vial cap rack (5).

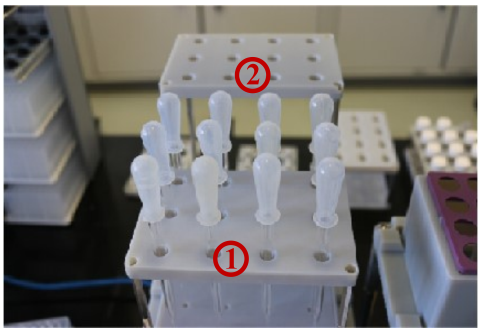


Figure 30. Glass pipette. Rack (1) is for placing the new glass pipettes and rack (2) is for placing the waste glass pipettes.

Another critical lab ware for life science automation is the microplate. After the standards about the microplate were published, the life science automation developed fast. A microplate could have 96, 384, 1536 or even more wells arranged in rows and columns. A well is a small test tube with a small volume, which could be 100 µL or even smaller. Thus, the solution used for assay can be save, and the cost can be reduced at the same time. As a well of the same microplate can contain a different solution with each other, so it is flexible to design contrast experiments. In the robot platform, the 96-well microplate has been used as the container for preparing samples. The microplates are stored in the hotels (see Figure 31). When pipetting, the robot will transport the select microplate to the deck. The waste microplate can also be transported back to the hotels. In some applications filtering is required. Using a syringe can do this. For drawing solution into the syringe, a syringe needle (cannula) will be needed. When filtering solutions, the needle will be changed to a filter. More details can be seen in Figure 32.

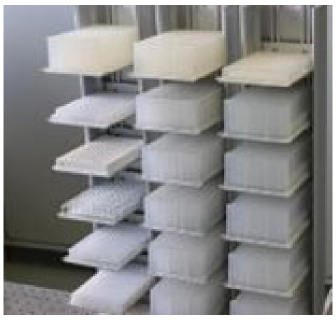


Figure 31. Hotels for storing microplate

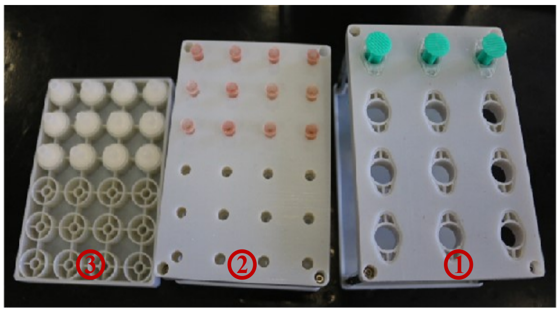
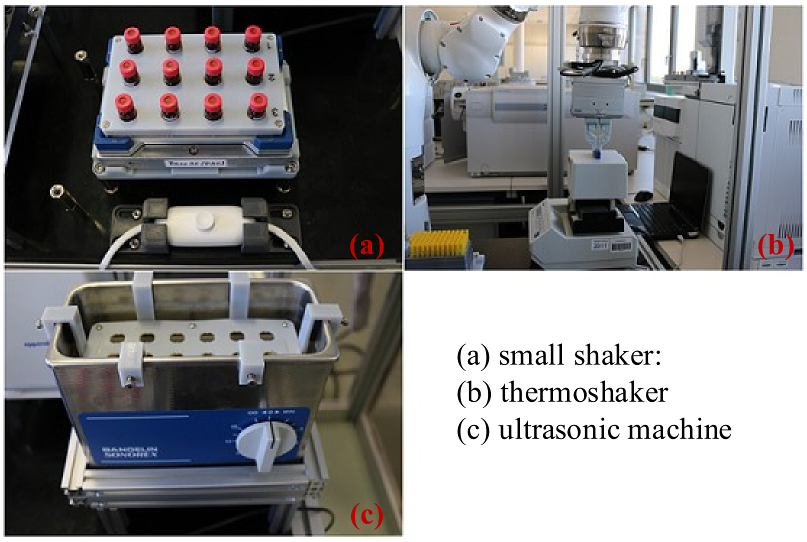


Figure 32. Syringe. Syringe rack (1), needle rack (2) and filter rack (3)

When mixing different solutions, a shaker will be helpful for mixing solutions. Figure 33 (a) shows a small shaker for shaking samples in vials. And after the vial rack is removed from the shaker, it can also be used to shake samples in a microplate. Sometimes the temperature will affect the reaction result. For shaking samples in different temperature conditions, a thermo shaker will be used, see Figure 33 (b). Besides, an ultrasonic device is also available for treating samples by using the ultrasonic wave. The picture of the ultrasonic machine can be seen in Figure 33 (c).



(a) small shaker:
(b) thermoshaker
(c) ultrasonic machine

Figure 33. Devices for shaking solutions.

Chapter 6 Concept of Motion Elements and Applications

The aims of the research are to automate the processes of sample preparation and analysis. To achieve this task, a CSDA10F dual-arm robot plays a critical role. However, programming for dual-arm robots is challenging. In the case of application in life science automation, the main challenges come from the complex experiment processes, which may require thousands of motion steps to fulfill them, and the narrow space environment restricting robot movement. In this case, in order to keep the flexibility of the dual-arm robot platform, robot motions have been designed in motion element units [130]. Motion element is an important concept in this dissertation, and will be explained in detail in this chapter. The programs for the robot motions have been built based on YASKAWA FS100 controller, by using INFORM III programming language [69].

6.1 Motion Elements

The design of the robot movement determines the flexibility and even feasibility of the dual-arm robot platform. As some motion steps[1] are reusable and some other motion steps can be shifted for repetitive movements, it is desirable to compose a series of motion steps as a unit, which is termed as a motion element in this dissertation. Thus, it will only need tens of motion elements to fulfill a complex experiment process instead of thousands of motion steps. When referring to robot motions, a motion element is a short movement consisting of several robot motion steps. When referring to programs, a motion element is a robot job. According to the characteristics of motion steps, motion elements are divided into two types: path motion element (called P-element for short) and relative motion element (called R-element for short). While a P-element consists of several fixed motion steps, an R-element is composed of several changeable motion steps. Different elements, no matter P-elements or R-elements, can be connected together by positions nodes. Besides, each element has a name, which is encoded with some logical information. By using the names, logical relationships between elements can be found. Each motion element was realized by a piece of codes wrote in INFORM language. One single motion element is not meaningful; several motion elements should be organized together to make functions. The concept of motion element can be seen in Figure 34. More details about motion elements will discussed in the following sections.

[1] A motion step means a movement of one robot arm or two arms from one position to another position.

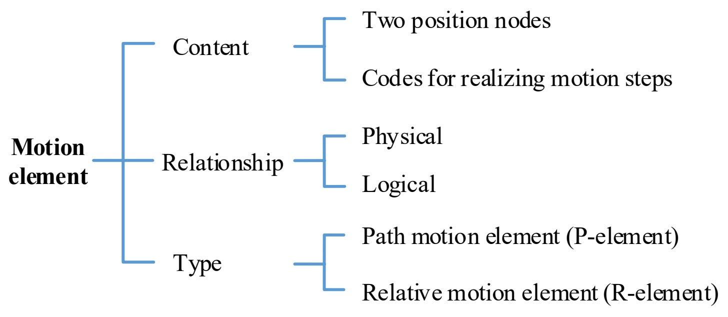

Motion element
Content
Two position nodes
Codes for realizing motion steps
Relationship
Physical
Logical
Type
Path motion element (P-element)
Relative motion element (R-element)

Figure 34. Concept of motion element

6.1.1 Path Motion Elements

A path motion element is composed of a series of motion steps, which has fixed position parameters. Programs for realizing P-elements are taught manually by using teach pendant [69]. By operating the teach pendant, robot arm/arms can be moved to the designed positions. When moved to a designed position, the parameters of the arm/arms are recorded to the robot controller. After a sequence of positions is recorded, a robot program (or robot job) is formed. When the robot job is executed, robot arm/arms will move following a designed path. A program example for realizing a P-element can be seen in Figure 38 (a).

As a path motion element has a fixed and designed moving path, they are suitable to be used to design the robot moving paths. As the robot has two arms and there is other lab ware around it, carefully designing moving paths will be very necessary to avoid collisions. One of the usages of P-elements is avoiding collisions between the robot arms and the surrounded environment and collisions between the two robot arms themselves especially in a narrow space environment. Another important usage is to move the robot arm/arms to a suitable position for executing relative motion elements. Figure 35 gives a comparison between a flexible posture and an inflexible one.

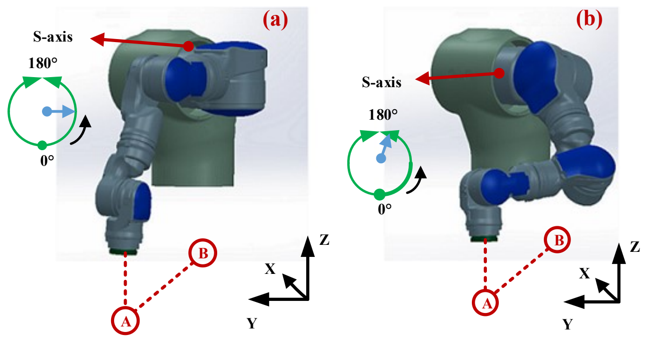


Figure 35. Arm posture: posture (a), posture (b).

In Figure 35, assume the S-axis has a motion range of ± 180 degree. In the posture (a) the S-axis had turned about 90 degrees in counterclockwise direction, while the S-axis had turned about 170 degrees in counterclockwise direction in the posture (b). When the robot arm is demanded to move to point A and then to point B following the red dotted line, assume that the S-axis will continue to turn in counterclockwise direction. In this case, the posture (b) becomes inflexible and the robot arm may even be not able to reach point B, because only about 10 degree of rotation range is still available for the S-axis. At this time, the posture (a) is more flexible. If there is one or more axis in the robot arm have a limited available rotation range, the arm posture is not very flexible. However, whether a posture is flexible is related with the direction where the arm is moving. Inflexible posture will limit the function of relative motion elements.

6.1.2 Relative Motion Elements

While a path motion element consists of a sequence of taught points, the motion steps of a relative motion element are relative to a certain coordinate system and based on calculations (a program example for realizing an R-element can be seen in Figure 38 (b)). They are suitable to be employed to deal with repetitive tasks (such as pipetting liquid to a microplate).

Figure 36 shows an example of a relative motion element. In (a), there is a user-defined coordinate system at the corner of the block. Assuming the vial has already been gripped by the robot and is being transferred from hole 1 to hole 2. As each motion step is relative to the user coordinate system, variables for setting the moving distance are used to configure each step. In this example, the relative motion element consists of 5 motion steps. After the first motion step is executed, the robot arm will pick the vial up 30 mm in Z direction. After the third step, the vial will be transferred 240 mm in X direction to above the hole 2. The distance parameter of each motion step is programmable. In this case, if the holes were moved to new positions in this user coordinate system, this motion element can be shifted to the new positions by reconfiguring the variables related with the moving distance of each step. As these motion steps are relative to the user coordinate system, the relative motion element will shift automatically when the coordinate system is shifted. This means, when the block is moved to a new position, this motion element can be also available. More details about R-elements will be discussed in the section of “6.2 Motion Frame”.

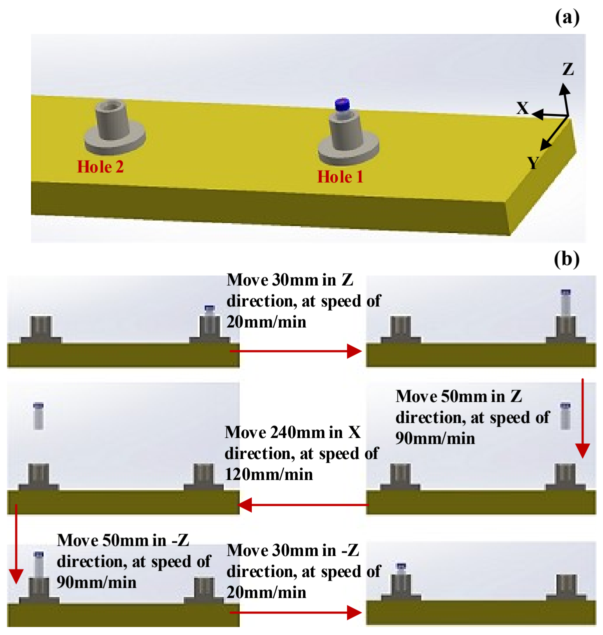

(a)
Z
X
Y
Hole 2
Hole 1
(b)
Move 30mm in Z direction, at speed of 20mm/min
Move 50mm in Z direction, at speed of 90mm/min
Move 240mm in X direction, at speed of 120mm/min
Move 50mm in -Z direction, at speed of 90mm/min
Move 30mm in -Z direction, at speed of 20mm/min

Figure 36. An example of relative motion element.

6.1.3 Position Nodes

A single motion element is not meaningful. In application, a series of motion elements should be combined together according specific tasks. Position nodes are used to combine motion elements. In each motion element, the position of its origin state and the position of its destination state are defined as position nodes separately. A position node about an origin state is named as “front node”, and a position node about a destination state is named as “end node”. A position node is the front node of one motion element and also the end node of the other motion elements. Besides, a position node could be one arm’s position or the two arms’ positions. In this way, different motion elements are linked together by position nodes.

6.1.3.1 Explanation of Position Nodes

As shown in Figure 37, there are three motion elements: element 1, element 2 and element 3. There are four motion steps in element 1. The position of P1 is defined as the front node of element 1. And the position of P4 is defined as the end node of element 1. It is a similar for element 2 (front node: P5, end node: P8) and element 3 (front node: P9, end node: P11). If the end node of element 1 is the same with the front node of element 2 (i.e. P4 is equal to P5), then element 1 and element 2 can be combined together to form longer motions. Case 1 gives a combination of motion elements, when P4 is equal to P5 and P8 is equal to P9. However, if P4 is equal to P9 and P11 is equal to P5, the combination will be different, as shown in case 2. In the case of equations between front nodes or between end nodes, combination cannot be performed. As in case 3 (end node P4 = front node P5 = end node P11), the three elements cannot be combined to one combination. Instead, they performed two combinations. In a combination, the element with its end node contributed has a higher priority. In the combination 1, the order of positions is P9-P10-P11 (P5)-P6-P7-P8, with element 3 being in front.

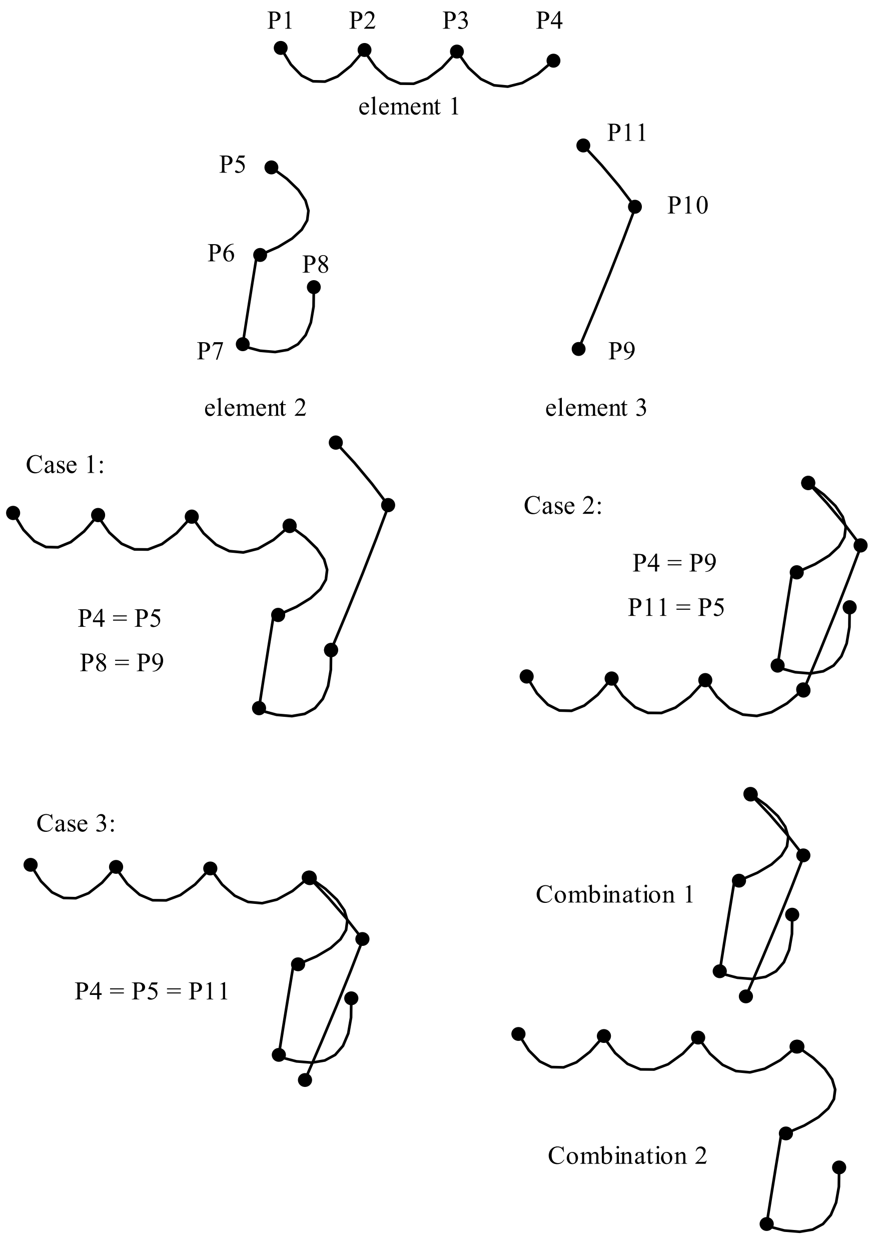
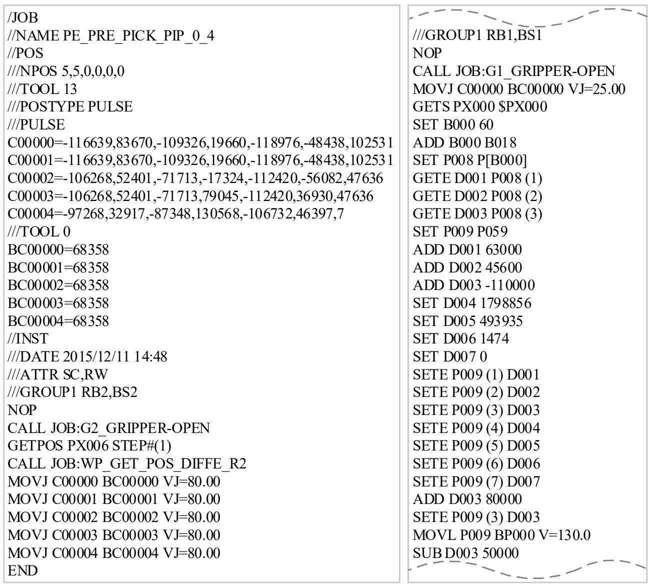


Figure 37. Combination of motion elements.

6.1.3.2 Position Nodes in Programs

As mentioned above, the positions of the first motion step and the last motion step are defined as position nodes separately. In Figure 38 (a), the position of the motion step "MOVJ C00000 BC00000 VJ=80.00", is defined as the front node, and its value is "C00000= -116639, 83670, -109326, 19660, -118976, -48438, 102531". The position of the motion step "MOVJ C00004 BC00004 VJ=80.00", is defined as the end node, and its value is "C00004= -97268, 32917, -87348, 130568, -106732, 46397, 7". For the relative motion elements, it is defined in the same way: as in Figure 38 (b), the position of the motion step "MOVJ C00000 BC00000 VJ=25.00", is defined as the front node, and it also has a similarly defined end node. The list of robot jobs for realizing motion elements can be found in Table 19 in Appendix B.1.



```
/JOB
//NAME PE_PRE_PICK_PIP_0_4
//POS
///NPOS 5,5,0,0,0,0
///TOOL 13
///POSTYPE PULSE
///PULSE
C00000=-116639,83670,-109326,19660,-118976,-48438,102531
C00001=-116639,83670,-109326,19660,-118976,-48438,102531
C00002=-106268,52401,-71713,-17324,-112420,-56082,47636
C00003=-106268,52401,-71713,79045,-112420,36930,47636
C00004=-97268,32917,-87348,130568,-106732,46397,7
///TOOL 0
BC00000=68358
BC00001=68358
BC00002=68358
BC00003=68358
BC00004=68358
//INST
///DATE 2015/12/11 14:48
///ATTR SC,RW
///GROUP1 RB2,BS2
NOP
CALL JOB:G2_GRIPPER-OPEN
GETPOS PX006 STEP#(1)
CALL JOB:WP_GET_POS_DIFFE_R2
MOVJ C00000 BC00000 VJ=80.00
MOVJ C00001 BC00001 VJ=80.00
MOVJ C00002 BC00002 VJ=80.00
MOVJ C00003 BC00003 VJ=80.00
MOVJ C00004 BC00004 VJ=80.00
END
```

(a)

```
///GROUP1 RB1,BS1
NOP
CALL JOB:G1_GRIPPER-OPEN
MOVJ C00000 BC00000 VJ=25.00
GETS PX000 $PX000
SET B000 60
ADD B000 B018
SET P008 P[B000]
GETE D001 P008 (1)
GETE D002 P008 (2)
GETE D003 P008 (3)
SET P009 P059
ADD D001 63000
ADD D002 45600
ADD D003 -110000
SET D004 1798856
SET D005 493935
SET D006 1474
SET D007 0
SETE P009 (1) D001
SETE P009 (2) D002
SETE P009 (3) D003
SETE P009 (4) D004
SETE P009 (5) D005
SETE P009 (6) D006
SETE P009 (7) D007
ADD D003 80000
SETE P009 (3) D003
MOVL P009 BP000 V=130.0
SUB D003 50000
```

(b)

Figure 38. Program examples of motion elements: path motion element (a) and relative motion element (b).

6.1.3.3 Prevention Mechanism

When there are a lot of motions elements, it is possible to make mistakes by combining two unmatched elements together, which can cause collisions. As shown in Figure 39, assume that element 1 and element 2 are unmatched elements, with neither the end node P4 being not equal to the front node P7 or the end node P9 being not equal to the front node P1. If the two elements are forced to combine together, an unknown path will be generated between P4 and P7 (assume element 1 is in the front). It could be path 1, path2, path 3 or others. Path 1 and path 2 can cause collisions with obstacles.

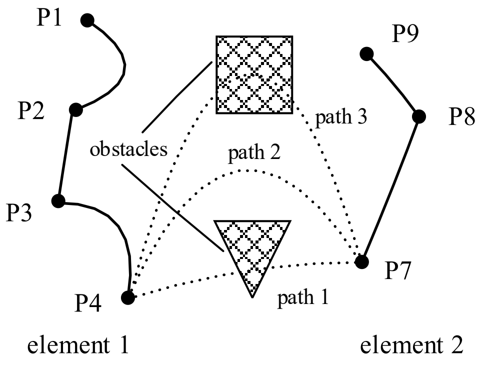


Figure 39. Combining unmatched elements can cause collisions.

In order to avoid collisions when unmatched elements are combined mistakenly, a prevention mechanism have been built, as shown in Figure 40. It is to check the matching degree between the current position of robot arm/arms and the front node of the motion element going to be executed. Before one motion element is executed, the current position of robot (if the element is for one arm, only the position of this arm will be read; if the element is for both arms, the positions of both arms will be read) and the front node of this element will be read. Then their values will be compared. If the difference between them is in acceptable range, the element will be allowed to access. Otherwise, it will be forbidden to execute this element and an alarm will be raised. The robot will stop working, until the alarm is release. This is for avoiding such collisions caused by executing unmatched motion elements. When the robot executes one motion element, it reaches the end node of this element and at the same time the robot reaches the front nodes of the other motion elements.

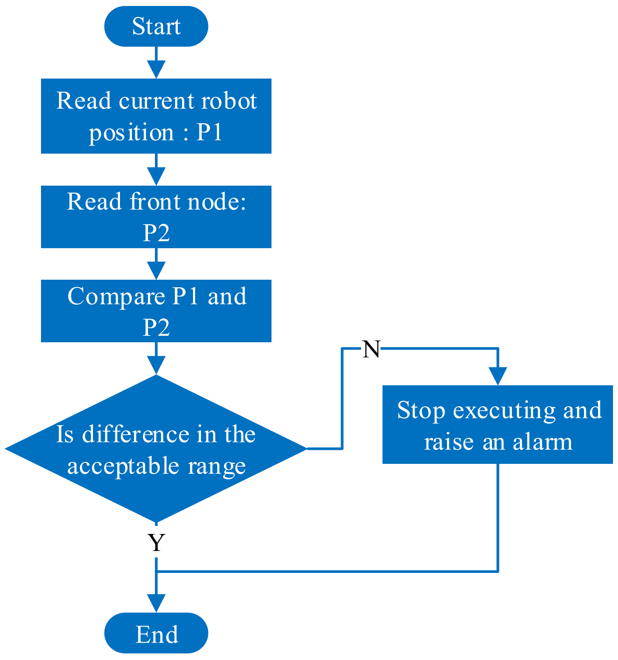

Start
Read current robot position : P1
Read front node: P2
Compare P1 and P2
Is difference in the acceptable range
N
Stop executing and raise an alarm
Y
End

Figure 40. Prevention mechanism of combining motion elements.

6.1.3.4 Naming for Motion Elements

Each motion element has a name, which is not only used to identify the element but also encoded with the information indicating its usage. The typical name of a motion element can be seen in Figure 41, which mainly includes three pieces of information: how, what and where. In the example, a motion element is named as "HO_PICK_MTP_F_HOTEL", of which the encoded information is "Hotel group_Pick_Microplate_From_Hotel". It can be seen that this motion element is used to pick a microplate from a hotel. The parameters determining the microplate and the hotel will be set by variables when using this motion element.

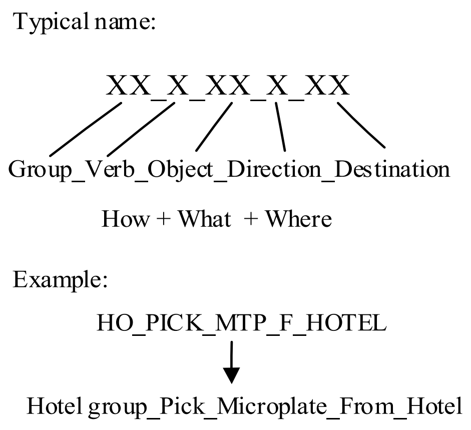

Typical name:
XX_X_XX_X_XX
Group_Verb_Object_Direction_Destination
How + What + Where
Example:
HO_PICK_MTP_F_HOTEL
Hotel group_Pick_Microplate_From_Hotel

Figure 41. Typical name of a motion element.

6.1.3.5 Combining Motion Elements

The following is an application example discussing how motion elements are used. Figure 42 shows the motion elements needed for pipette function module. There are 5 relative motion elements ("Pick pipette", "Put pipette", "Load tips", "Get solution" and "Release solution") and 9 path motion elements ("8 path motions" and "Release tips").

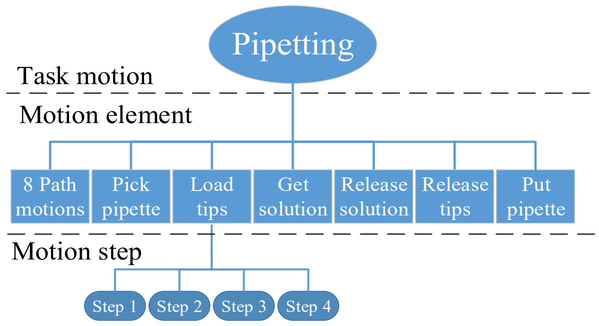


Figure 42. Motion elements for pipetting task ("8 Path motions" means 8 different path motion elements, and "Release tips" is also a path motion element).

While Figure 42 shows the composition of the motion elements for pipetting, Figure 43[2] describes the links between the motion elements. In Figure 43, a solid line represents a motion element of the right robot arm, a dash line represents a motion element of left robot arm, and a dot line represents a motion element of both robot arms. At the same time, a curved line represents a relative motion element, while a straight line represents a path motion element. The arrow of each line shows the motion direction from one node to another. Besides, a circle represents a position node (node (1) includes the right arm and the left arm positions). It can be seen the line between node (1) and node (2) has two arrows. This means there are two path motion elements: the one from node (1) to node (2) is used to pose the right robot arm to be ready for gripping a pipette and the other one from node (2) to node (1) is employed to pose the right robot arm to the standby position after putting the pipette back to its rack.

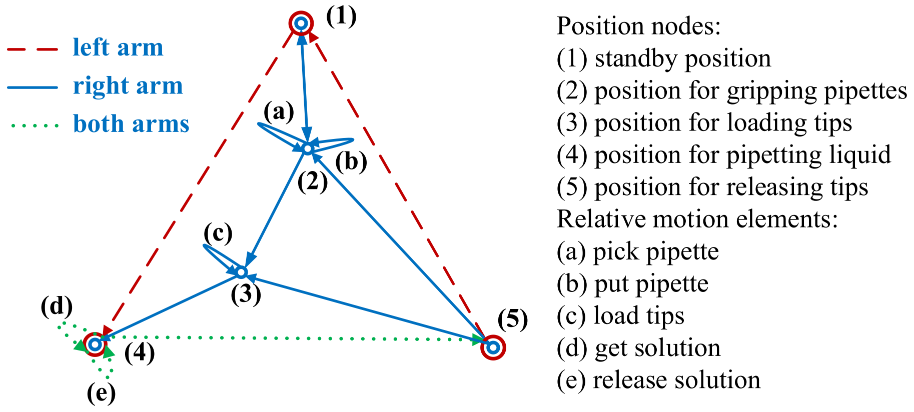


Figure 43. Motion element links showing relationships between motion elements.

From Figure 43, it can be seen that a relative motion element begins and ends at the same position node, e.g., the curved line (d) and curved line (e). This means that after drawing liquid the robot is ready

[2] The graphic shown in Figure 43 is different with that shown in Figure 49. Thus, it can be seen that, when the nodes of motion elements are modified, the relationships between these motion elements will also be changed.

to release the solvent and also after releasing the solvent the robot is ready to draw liquid again. The line from node (4) to node (5) means that the two arms move synergistically to release the pipette tip/tips. The line from node (5) to node (1) means that the left robot moves back to the standby position. However, after releasing the tip/tips, the right arm has three choices: moving from node (5) to node (3) to load new tip/tips and then to pipette liquid; moving from node (5) to node (2) to change a new pipette and then to pipette liquid or moving from node (5) to node (2) to put back the pipette and then go back to standby position. Figure 44 gives three examples of organizing motion elements, but the ways of organizing motion elements are not limited in the three cases.

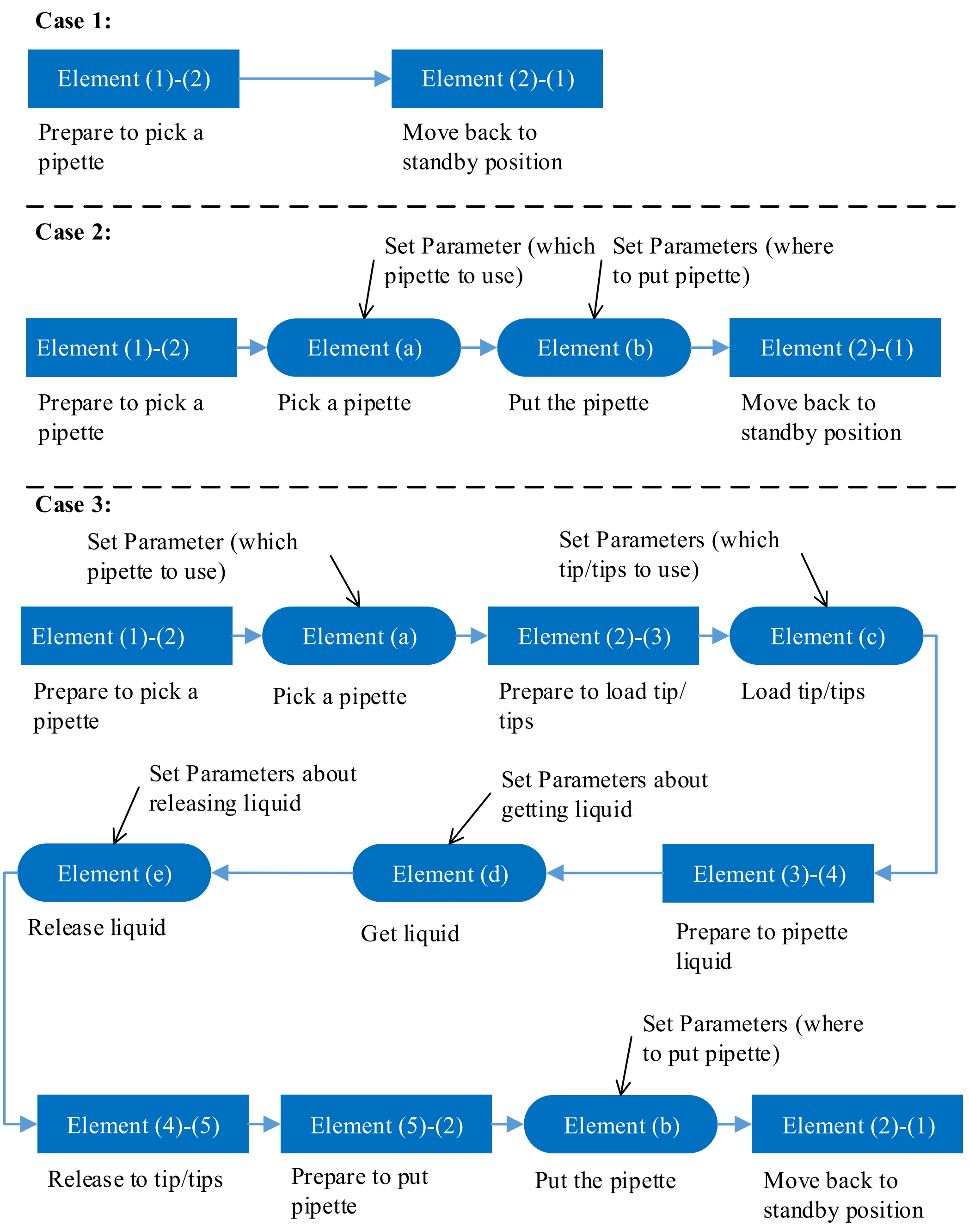

Case 1:
Element (1)-(2)
Element (2)-(1)
Prepare to pick a pipette
Move back to standby position
Case 2:
Set Parameter (which pipette to use)
Set Parameters (where to put pipette)
Element (1)-(2)
Element (a)
Element (b)
Element (2)-(1)
Prepare to pick a pipette
Pick a pipette
Put the pipette
Move back to standby position
Case 3:
Set Parameter (which pipette to use)
Set Parameters (which tip/tips to use)
Element (1)-(2)
Element (a)
Element (2)-(3)
Element (c)
Prepare to pick a pipette
Pick a pipette
Prepare to load tip/ tips
Load tip/tips
Set Parameters about releasing liquid
Set Parameters about getting liquid
Element (e)
Element (d)
Element (3)-(4)
Release liquid
Get liquid
Prepare to pipette liquid
Set Parameters (where to put pipette)
Element (4)-(5)
Element (5)-(2)
Element (b)
Element (2)-(1)
Release to tip/tips
Prepare to put pipette
Put the pipette
Move back to standby position

Figure 44. Examples of organizing motion elements.

In Figure 43, the motion element (1) -(2) has an arrowing pointing to node (2) and the motion element (2) -(1) begins from node (2) too, with node (2) as the shared node. Thus, the motion element (1) -(2)

and motion element (2) -(1) can be combined together as the case 1 shown in Figure 44. In this case, the robot will move to prepare to pick a pipette but then moves back to the standby position without executing a meaningful task. The motion element (a) also begins from node (2). So, motion element (a) can also be combined with motion element (1) -(2) too. As (a) is a relative motion element, related parameters should be set. The list of parameters has been given in Table 20. In the same way, the motion element (b) is also available to be combined. In the case 2, the robot will move to pick a pipette, put the pipette back and then move back to the standby position. Similarly, the motion elements can be combined together in the way as shown in the case 3, in which the robot will load tip/tips to the pipette, pipette liquid, put the pipette back after releasing the tip/tips and then move back to the standby position. Of cause, the ways of organizing motion elements are not limited in the three cases. In fact, there are an infinite number of combinations due to loops constituted by motion elements. There are two obvious loops between node (2), (3), (4) and (5), as shown in Figure 45. There are three simple rules for combining motion elements:

1) Rule 1: following the arrow to combine motion elements.
2) Rule 2: motion elements in loops can be reused circularly.
3) Rule 3: according the task to combine motion elements, as not all combinations are meaningful (e.g. case 1 and case 2 in Figure 44).

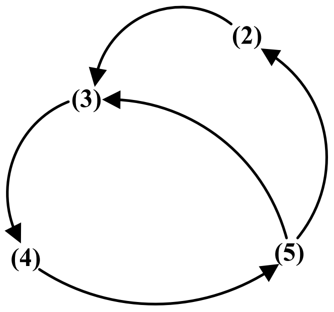


Figure 45. Loops constituted by motion elements.

6.2 Motion Frames

Different lab ware will be used to prepare samples, including different pipettes (single channel pipettes and 8-channel pipettes), glass vials (22mL, 10mL, 4mL and 2mL), microplates, reagent reservoirs, glass pipettes, syringes, thermo shaker and the other lab ware. Thus, a high number of robot motions have to be built. Besides, the sample preparation procedure is complex in certain degree. For example, the powder may be first dissolved in vials and then be pipetted to a microplate, solution may be first diluted in one vial and then be mixed with other solutions in different ratios and also certain other solutions may be added in order to promote or stop the chemical reaction. Both of them cause the programming for the robot being a big and complex work.

Though different lab ware and complex procedure are involved, there are still many similar parts. These pipettes are hanged on the shelf, so the motions for picking a pipette from the shelf or putting it to the shelf could be similar. Also, the motions for opening or closing vials could be similar, as the movements of unscrewing or screwing the cap of a vial have no directly relationship with the size of the vial. After similar parts were found, the complex work was divided into small units. A simple and short robot motion is a unit. Such a robot motion unit is defined as a motion element. According to the properties of the motion elements, they are divided into two types: path motion elements, with fixed moving path, and relative motion elements, with changeable moving path. In this paper, a relative motion element is also called “Motion Frame”, which will be detailed in the following content.

6.2.1 Structure of Motion Frame

The simple example (in Figure 36) demonstrates the nature of motion steps. However, it is not necessary to set the moving distance of motion steps. Otherwise, the motion frame will not be so easy to use. In this example, there is only one hole (hole 2). In the other situation, maybe there is another hole (e.g. hole 3) for transporting the vial to. Thus, it requires a variable to identify the hole. And, by setting a value to this variable, the vial will be transported to the related hole (hole 2 or hole 3). Such variables compose the variable interface of a motion frame. By setting values to these variables, a motion frame will be configured. In other words, by setting values to these variables the moving distance of each motion step is calculated. A reference point achieves the initialization of the calculation. In summary, a motion frame consists of three parts: motion steps, variable interface and reference point, as shown in Figure 46. The motion steps will be realized based on calculations using variables. Variables are important for setting parameters for the calculations and are named as variable interface. As a motion frame is realized based on calculations, it is necessary to provide an initial value for the calculations. This will be achieved by a reference point, which records a posture of a robot arm. Depending on different tasks, reference points are divided into static reference points and dynamic reference points. This will be discussed in the following parts.

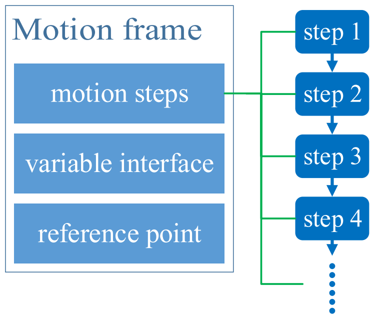


Figure 46. Motion frame structure.

6.2.2 Reference Points of Motion Frames

The reference point is used to initialize the calculation of the motion frames. The static reference point and the dynamic reference point have predefined values, while the virtual reference point is related with the actual position of robot arm.

A) Static Reference Point

In the platform, eight types of pipettes are supported and are positioned on the shelf through the 3D printed adapters, as shown in Figure 47. When pipetting, the robot will pick a selected pipette from the shelf. When finished, the robot will put it back to the shelf. A user coordinate system had been defined, to provide reference to the motion frame for picking or putting a pipette. The position of the first pipette (marked as "0") is selected as reference point. After the motion frame for "0" pipette is built, it is also available for other pipettes. As the pipettes are positioned in a straight line, only one variable will be required to configure the motion frame.

For picking a pipette, a value to the variable for configuring the picking-motion frame will be set. When picking a microplate from the hotel, one variable will not be enough to identify the microplate, as microplates are distributed in rows and columns (see Figure 31). A user coordinate system had also been defined for building motion frames. The position of the first microplate (right-top corner) is used as the reference point. The motion frame uses two variables as variable interface. After the picking-motion frame is built, by configuring two variables (one for row and one for column) the motion frame can be used to pick any microplate from the hotel. Also for putting a microplate to any position in the hotel, only one motion frame is needed. It is also possible to select other positions in the user coordinate system as the static reference point, e.g. the origin of coordinates. This will not affect the motion frame. The only difference is how to calculate each motion step.

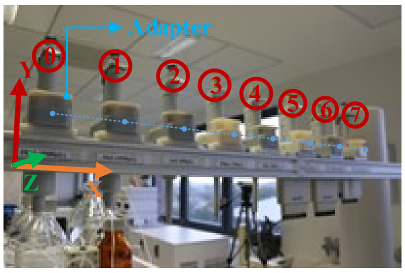


Figure 47. Different types of pipettes.

B) Dynamic Reference Point

The reference points used in the above examples are static reference points. This is due to the fixed position of the shelf and the fixed position of the hotels relative to the platform. If such position is not fixed, then dynamic reference points will be required instead. Figure 48 shows the deck layout. Beside other lab ware, several ALPs (Figure 48 (b)) were mounted, which are used to place lab ware, such as microplate, reagent reservoirs and vial racks. Such a lab ware can be placed on different ALPs. Thus, it is flexible to lay out lab ware for different experiment processes. As shown in Figure 48 (c), the wells in the microplate are distributed in the similar way as the microplates in the hotel. Thus, it is possible to build the motion frame in the similar way: define a user coordinate system, select a static reference point, build motion steps and then set variable interface. However, there will be some limitations. When there is more than one microplate used, motion frames have to be built for each microplate. When the microplate is placed on a different ALP, the motion frames have to be re-built again. To solve this problem, dynamic reference points are used instead of static ones.

Such a reference point is defined related to ALP. A sharp corner of an ALP is selected as the reference point. The corner of each ALP is used as the reference point. In Figure 48, it can be seen another user coordinate system defined on the deck, which is used to realize these reference points. As each corner has a fix position, each reference point has a fixed value relative to the coordinate system. In fact, such reference point is also static. Motion frames built using such corner as reference points, are flexible to use, as all the ALPs are the same and a corner has a fix position relative to the ALP. When the microplate is placed on a different ALP or when more the one microplate is used, the motion frames are also available. Beside the variables for configuring the motion frame itself (e.g. two variables for identifying a well on a microplate), it includes one more variable to identify which ALP it is on or saying to identify which reference point using. The referent point itself is fixed, but the reference point used in a certain motion frame is dynamic. The dynamic reference point is not only used for the motion frames about microplates, but also available to other lab ware which can be placed on the ALP, such as vials, tips and syringes (with racks for holding them). This brings another important flexibility. For one experiment, maybe one microplate and several vials are used, while for another experiment, maybe two microplates or syringes are used. However, the robot jobs (robot programs) do not have to be modified. It only needs a reconfiguration of the motion frames by setting new values to the variable interface.

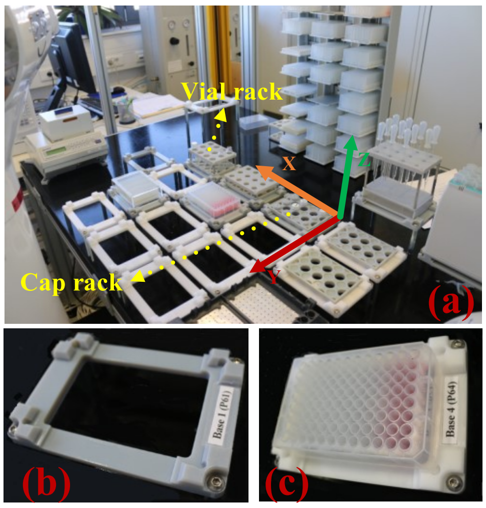


Figure 48. Deck layout. The deck mounted with ALPs (a), the ALP (b) and a microplate on an ALP (c).

C) Virtual Reference Point

A virtual reference point is not a real defined point. It is related to the current robot arm position. A static or a dynamic reference point is not so helpful to the motion frame of screwing or unscrewing vial caps. When opening or closing a vial, one robot arm (arm 1) is used to hold the vial, which will be kept immobile, and the other arm (arm 2) is used to screw or unscrew. The motion frames were built referencing to the position of arm 2. After the cap is put to the vial, the position of arm 2 will be read by the program and then used as the reference for the following unscrewing motion steps. Different types of vials need different loops of unscrewing motions, and the caps have different sizes for the gripper to grip. Thus, one variable has to be configured for the motion frame to identify the type of the vial. In our application, four types of vials with different volumes (22mL, 10mL, 4mL and 2mL) will be used. For opening them, one unscrewing-motion frame was built. For closing the vials, one screwing-motion frame was built.

6.3 Motion Elements in Applications

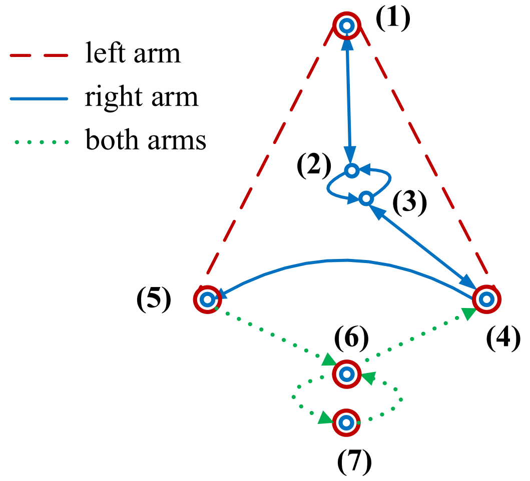
This platform has been designed with ten function modules, and in each module Yaskawa CSDA10F dual-arm robot plays as the leading role, by employing its two flexible arms to use the lab ware in the similar way as a human. They are pipette module, glass pipette module, glass vial module, hotel module, tip box module, syringe module, ultrasonic machine module, thermo shaker module, LC auto sampler module and GC auto sampler module separately. Each module consists of several motion elements. In Table 3, the functions of each module had been introduced.

Table 3. Functions of modules.

	Functions
Pipette module	1. Pick/put pipettes 2. Load/release tip/tips 3. Transfer liquid: a) from liquid reservior to microplate b) from liquid reservior to glass vial c) from vial to microplate d) from vial to reservior e) from microplate to microplate
Glass pipette module	1. Pick/put glass pipette 2. Transfer liquid from vial to vial
Glass vial module	1. Remove cap from vial 2. Screw cap on vial 3. Transfer vial
Hotel module	Transfer microplate between hotel and deck
Tip box module	Transfer tip box between shelf and deck
Syringe module	1. Pick/put syringes 2. Load/release cannula 2. Load/release filter 3. Filter liquid
Ultrasonic machine module	1. Transfer vials between ultrasonic machine and shaker 2. Start shaking 3. Stop shaking
Thermo shaker module	1. Remove/cover lid of thermoshaker 2. Tansfer microplate: a) from deck to thermoshaker b) from thermoshaker to deck c) from deck to autosampler tray
LC Auto sampler module	1. Open/close the door of autosampler 2. Transfer autosampler tray a) from deck to autosampler b) from autosampler to deck 3. Transfer microplate: a) from autosampler tray to hotel
GC auto sampler module	Transfer vial to GC autosampler

6.3.1 Pipette Module

The pipette module is used to operate pipettes to transfer liquid: a) from reservoir to microplate, b) from reservoir to glass vial, c) from vial to microplate, d) from vial to reservoir or e) from microplate to microplate. Eight types of pipettes are supported currently (see Figure 47). The robot motions, including pick/put pipette, load/release tip/tips, get/out liquid and some path motions, are realized by several motion elements. The motion elements and their relations have been shown in Figure 49.



1: Standby
1-2: Prepare to pick a pipette
2-1: Back to standby
2-3: Pick a pipette
3-2: Put a pipette
3-4: Prepare to load tip
4-3: Prepare to put pipette
4-5: Load tip
5-6: Prepare to get liquid
6-7: Get liquid
7-6: Out liquid
6-4: Release tip

Figure 49. Motion elements of pipette module.

In Figure 49, position node (1) is the standby position. Normally, the robot starts from the standby position to execute a task and goes back to the standby position when a task is finished. The straight line between node (1) and node (2) has two arrows, meaning there are two motion elements. The line from (1) to (2) is to prepare to pick a pipette. The line from (2) to (3) is a curved one, means that this element is a relative one. And this element is for picking a pipette. After picking a pipette, a motion element, from (3) to (4), is used to prepare to load tips. As usually there are many tips available in a tip box, so the motion element for loading a tip to a pipette is designed as a relative one. The motion element, from (5) to (6), is to prepare to get liquid (draw liquid into a pipette). The line from node (6) to (7) represents the motion element of getting liquid: from reservoirs, glass vials or microplates, while the line from node (7) to node (6) represents the motion elements of outing liquid (press liquid out from a pipette). After outing liquid, the user can continue to pipette liquid. The user can also choose to change a new tip/tips, before continue to pipette liquid. The motion element, from (6) to (4), is to release the tip/tips from a pipette. From Figure 49, it can be seen that after this element is executed, the robot will go back to position node (4). Thus, the robot is ready to load a new tip/tips, and then to pipette liquid. However, it is also feasible to go back to node (3) from node (4), which means to prepare to put the pipette back to the shelf. The motion element, from node (2) to node (3), is to put the pipette back to the shelf. After putting back the pipette, the user can choose to pick another pipette for pipetting liquid or go back to the standby position, by using the motion element from node (2) to node (1).

6.3.2 Glass Pipette Module

This module is designed to use glass pipette to pipette liquid. Beside some path motions, the robot will also do some other motions: pick/put GP (glass pipette) and draw/release liquid. The related motion elements and their relations have been shown in Figure 50. In this module, the robot also starts from the standby position. From node (1) to node (2) is to prepare to pick a glass pipette, while from node (2) to node (1) is to move back to the standby position. The motion element, from node (2) to node (3), is to pick a glass pipette from the rack. After that, the motion element, from node (3) to node (4), will be used to prepare to get liquid. When the robot is in the posture ready to get liquid, (4)-(5) element will be used to get liquid and then (5)-(4) element will be used to press out the liquid. After releasing liquid, it is feasible to use the (4)-(5) element and (5)-(4) element to repeat pipetting liquid or use the (4)-(6) element to prepare to put the glass pipette to the rack. The (6)-(2) element is to put a glass pipette to the rack. After this, it is feasible to use the (2)-(3) element to pick another glass pipette to pipette liquid or use the (2)-(1) element to move back to the standby position.

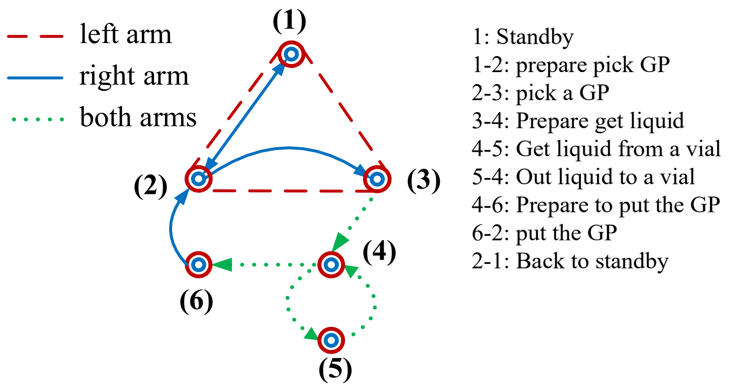


Figure 50. Motion elements of glass pipette module.

6.3.3 Glass Vial Module

For the robot to prepare sample, the human operator will provide some liquids and powders to the robot. And some parts of the liquids and powders are placed in vials. Also, some vials will be used to mix solutions. Thus, it is needed for the robot to deal with vials, including opening vial (removing the cap from the vial), closing vial (screwing the cap onto the vial) and transferring vial. For these tasks, 7 path motion elements and 4 relative motion elements are designed. These relative motion elements are "pick vial", "put vial", "pick cap" and "put cap" separately. The relationships of the motion elements can be seen in Figure 51.

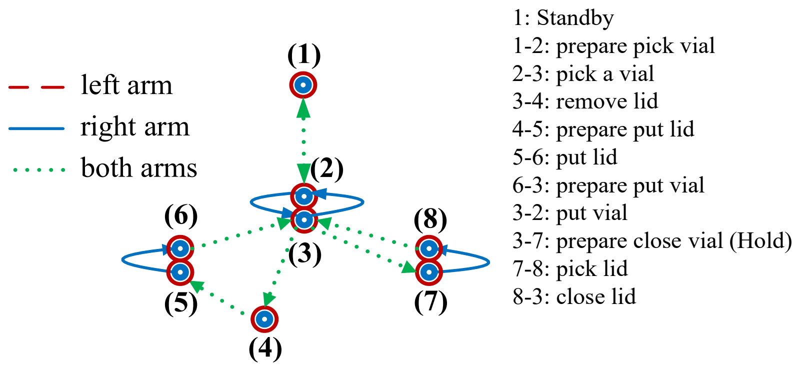


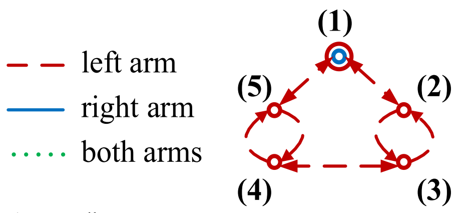
Figure 51. Motion elements of glass vial module.

The robot starts from the standby position – position node (1). The (1)-(2) element is to prepare to pick a vial and the (2)-(1) element is to move back to the standby position from the picking position. The (2)-(3) element is to pick a vial, while the (3)-(2) element is to put the vial. If a vial is closed, it is feasible to use the (3)-(4) element to remove its cap. Before using the (5)-(6) element to put the cap, the

(4)-(5) element has to be executed first. After putting a cap, the (6)-(3) element is to prepare to put a vial and the (3)-(2) element is to put the vial back to the rack. If a vial is open, it is feasible to use the (8)-(2) element to close the vial. For doing this, the (3)-(7) element and the (7)-(8) elements should be used first. After closing the vial, it is feasible to use the (3)-(2) element to put the vial back to the rack. After opening, closing or transferring a vial, the user can choose to continue to do these tasks, or use the (2)-(1) element to go back to the standby position.

6.3.4 Hotel Module

The hotels are used to store microplates. When pipetting, the robot will pick a new microplate from hotels and put it onto an ALP on the deck. And the hotels are also used for placing waste microplates. Figure 52 shows the motion elements for transferring microplates.



1: Standby
1-2: prepare pick a microplate from hotel
2-1: back to standby
2-3: pick a microplate from hotel
3-2: put the microplate back to hotel
3-4: change to a suitable posture for putting MTP on table
4-3: change to a suitable posture for putting MTP to hotel
4-5: put the microplate on the table
5-4: pick the microplate from the table
5-1: back to standby position
1-5: prepare to pick a microplate from the table

Figure 52. Motion elements of hotel module.

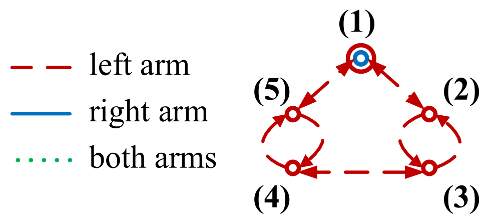
In Figure 52, the robot also starts from the standby position – position node (1). The (1)-(2) element is to prepare to pick a microplate from a hotel. The (2)-(1) element is to go back to the standby from the preparing position. The (2)-(3) element is to pick a microplate from a hotel, while the (3)-(2) element is to put a microplate to a hotel. For using the (4)-(5) element to put a microplate onto an ALP on the deck, a preparing motion element is needed – (3)-(4) element. After putting the microplate, the (5)-(1) element is used to move the robot to the standby position. This is the process of picking a microplate from a hotel and then putting it to the deck. It is also feasible to pick a microplate from the deck and then to put it to hotels. Then the process will be this way: (1)-(5) element, (5)-(4) element, (4)-(3) element, (3)-(2) element and then (2)-(1) element.

6.3.5 Tip Box Module

For different types of pipettes, different types of tips will be needed. Thus, the robot needs to be able to change the tip box. The graph in Figure 53 looks the same with the one in Figure 52, but they are related with different motion elements. However, the way of using the motion elements are also in a similar way. The difference is that the transportations of tip boxes are between the shelf and the deck.

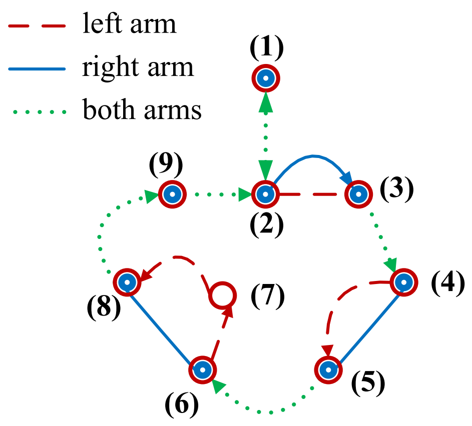
6.3.6 Syringe Module

For some kinds of solutions, a filter process is needed, which can be done by using syringes. Figure 54 shows the components of the robot motion elements for using syringes. First, the robot will pick a syringe and load a cannula to it for drawing the solution. Then, the cannula will be released and a filter will be loaded to the syringe to filer the solution in the syringe. Figure 54 show the relationships of the motion elements for using syringes to filter solutions. As the robot start from a standby position, a motion element from node (1) to node (2) is necessary to move the robot to a position ready for picking a syringe. The motion element from node (2) to node (3) is for picking a syringe from the rack. As it is a relative motion element, different syringes can be picked by setting related parameters. After picking a syringe, it will be moved to the left gripper to hold it to load a cannula. Then the two robot arms will work together to draw solutions from glass vials. Before loading a filer, the cannula will be released from the syringe by using the motion element from node (6) to node (7). After the filter is loaded, the solution in the syringe will be filtered to other glass vials. Then the syringe will be released to a recycling bag.



1: Standby
1-2: prepare pick a tip box from the profile shelf
2-1: back to standby
2-3: pick a tip box from the profile shelf
3-2: put the tip box back to the profile shelf
3-4: change posture to put tip box on table
4-3: change posture to put tip box to the profile shelf
4-5: put the tip box on the table
5-4: pick the tip box from the table
5-1: back to standby position
1-5: prepare to pick a tip box from the table

Figure 53. Motion elements of tip box module.

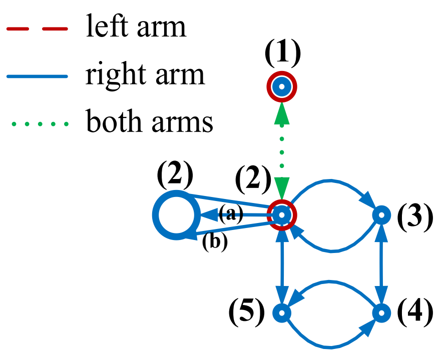


1: standby
1-2: prepare pick syringe
2-3: pick a syringe
3-4: put syringe to left gripper
4-5: load cannula to syringe
5-6: get liquid
6-7: Release cannula
7-8: load filter to syringe
8-9: out liquid
9-2: release syringe
2-1: back to standby

Figure 54. Motion elements of syringe module.

6.3.7 Ultrasonic Device Module

This module is used to provide the environment for mixing solutions using an ultrasonic device and a small shaker together. Turning the knob sets the ultrasonic vibration time and the shaker can be started or stopped by pressing the button. Figure 55 shows the relationship of the motion elements for using the ultrasonic cleaning machine and the shaker. The robot will start from the standby position. After picking a vial from the shaker by the motion element (from node (2) to node (3)), it is available to choose to put the vial to another position in the shaker by using motion element (from node (3) to (2)) or prepare to put the vial to the ultrasonic machine by using motion element (from node (3) to node (4)). For dealing with the vials in ultrasonic machine, it works in the same way. For using the ultrasonic machine, it is needed to set the time ultrasonic vibration time, which will be done by using motion element (2)-(2)-(b). And for starting or stopping the shaker, it will be done the by motion element (2)-(2)-(a).

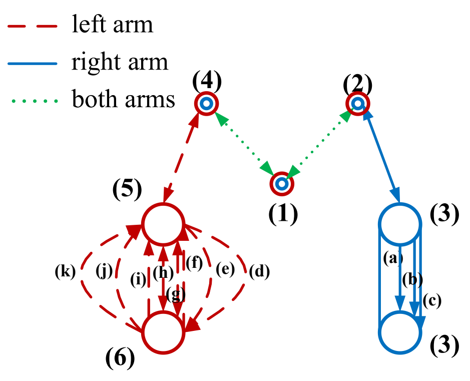


1: standby
1-2: prepare pick vial
2-3: pick vial from shaker
3-2: put vial to shaker
3-4: Prepare put vial to ultrasonic
4-3: Prepare put vial to shaker
4-5: Put vial to ultrasonic
5-4: pick vial from ultrasonic
2-5: Access to ultrasonic
5-2: After access to ultrasonic
2-2 (a): turn shaker on/off
2-2 (b): set time to ultrasonic
2-1: back to standby

Figure 55. Motion elements of ultrasonic module.

6.3.8 Thermo Shaker Module

This part is designed to use the thermo shaker to shake samples by the robot. For this module 17 path motion elements and 4 relative motion elements have been designed, see Figure 56. The element (a) is to remove the lid from the thermo shaker. Element (b) is to press the button to start/stop shaking. Element (c) is to cover the lid to the thermo shaker. Element (f) is to put microplate to the thermo shaker. When shaking, a lid on the microplate is needed to prevent evaporation of solution from microplate. Element (6)-(5)-(g) is to put the lid to microplate in the thermo shaker. Element (5)-(6)-(g) is to pick microplate from thermo shaker. Element (5)-(6)-(h) is to pick microplate lid from thermo shaker, and the element (6)-(5)-(h) is to put microplate to the drawer. The element (i) is to put microplate lid to the drawer. Element (d) is to pick a microplate from deck. Element (e) is to pick a microplate lid from the deck. Element (j) is to put the microplate to the deck. Element (k) is to put the microplate lid to the microplate in the drawer. The robot starts from standby position. Two robot arms are used to fulfill the different tasks independently. In order to avoiding collisions, when one arm is doing the task, the other arm is forbidden. This is achieved by moving the robot arms to certain positions before doing the tasks. The (1)-(2) element is to move the robot arms to the specific position node (2). As this position node does not connect with the motion elements of left arm, left arm is forbidden. The (2)-(3) element is to access to the thermo shaker. Then, it is feasible to remove the lid of thermo shaker, cover the lid of thermo shaker or press the button of thermo shaker.

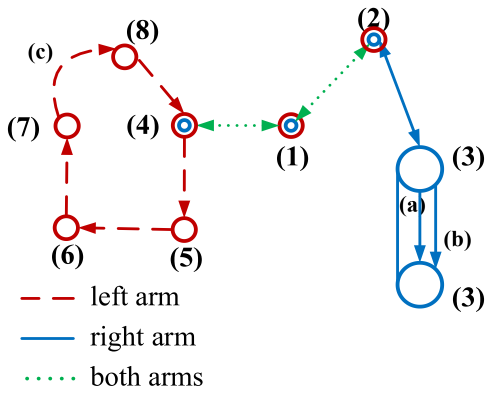


1: Standby
1-2: Enable right arm
2-1: Back to standby
2-3: Prepare to access to thermoshaker
3-2: Back to Enable position
3-3 (a): Remove the lid of thermoshaker
3-3 (b): Press the start/stop Button
3-3 (c): Cover the lid of thermoshaker
1-4: Enable left arm
4-1: Back to standby
4-5: Prepare to pick a microplate
5-4: Back to Enable position
5-6 (d): Pick a MTP from the deck
5-6 (e): Pick a MTP lid from the deck
5-6 (g): Pick the MTP from thermoshaker
5-6 (h): Pick the MTP lid from thermoshaker
6-5 (f): Put the MTP to thermoshaker
6-5 (g): Put the MTP lid to thermoshaker
6-5 (h): Put the MTP to drawer
6-5 (i): Put the MTP lid to drawer
6-5 (j): Put the MTP to Deck
6-5 (k): Put the MTP lid to deck

Figure 56. Motion elements of thermo shaker module.

6.3.9 LC Auto Sampler Module

This module is for transferring a microplate with samples to the LC auto sampler for analysis. For this aim, 12 path motion elements and 1 relative motion element have been built. Element (a) is the path motion element for take the drawer out from the LC auto sampler and then put it on the deck. Element (b) is the path motion element to pick up the drawer from the deck and then put it into the LC auto sampler. The relationships between these motion elements can been seen in Figure 57.



1: Standby
1-2: Enable right arm
2-1: Back to standby
2-3: Open the door of LC autosampler
3-2: Close the door of LC auto sampler
3-3 (a): Pick the drawer from LC autosampler and then put it on the table
3-3 (b): Pick the drawer from the table and then put it on the LC autosampler
1-4: Enable left arm
4-1: Back to standby
4-5: Prepare to pick a microplate
5-6: Pick the MTP from LC drawer
6-7: Prepare to put MTP to hotel
7-8: Put the microplate to the hotel
8-4: Back to Enable position

Figure 57. Motion elements of LC auto sampler module.

In Figure 57, the robot starts from the standby position – position node (1). The (1)-(2) element is to enable the right arm, while the (1)-(4) element is to enable the left arm. This is to avoid collisions when using wrong motion elements. After using the (2)-(3) element to open the door of LC auto sampler, it is feasible to use the (3)-(3)-(a) element to pick the drawer (MTP is on the drawer) out from the LC auto sampler and then put it to the deck or use the (3)-(3)-(b) to pick the drawer from the deck and then out put it into the LC auto sampler. After using (3)-(3)-(a) element or (3)-(3)-(b) element, the (3)-(2) element can be used to close the door of LC auto sampler. When the analysis is finished, it needs to put the microplate back to the hotel. The left robot arm can do this. After the (3)-(3)-(a) element is used, the drawer is put on to deck. The (4)-(5) element is to prepare pick the microplate from the drawer. After using (5)-(6) element to pick the microplate from the drawer, the (6)-(7) element is used to prepare to put the microplate to the hotel. The (7)-(8) element is to put the microplate into the hotel.

6.3.10 GC Auto Sampler Module

The main task of this module is to transport the prepared samples in vials to GC auto sampler for analysis. Figure 58 shows the relationship of the motion elements for transporting vials to GC auto sampler. After picking a vial from vial rack, the path motion element (from node (3) to node (4)) can be used to move the robot arm to a suitable position for putting vials to the tray of GC auto sampler. When the motion element (from (4) to (5)) is executed, the vial will be put into one well of tray. And it can be selected by setting related variables, to choose which well to put the vial.

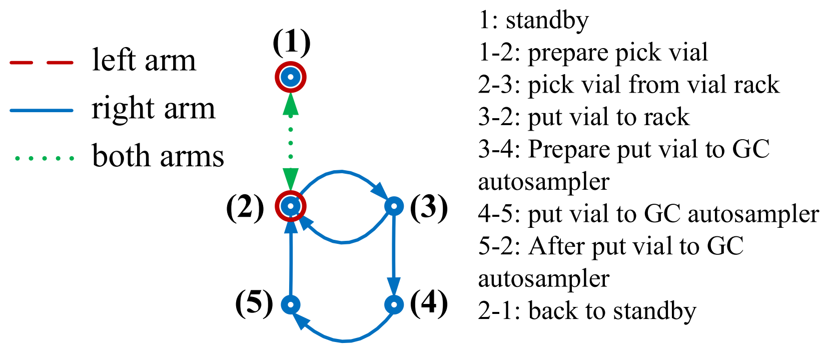

left arm
right arm
both arms
(1)
(2)
(3)
(4)
(5)
1: standby
1-2: prepare pick vial
2-3: pick vial from vial rack
3-2: put vial to rack
3-4: Prepare put vial to GC autosampler
4-5: put vial to GC autosampler
5-2: After put vial to GC autosampler
2-1: back to standby

Figure 58. Motion elements of GC auto sampler module.

Chapter 7 Integration of Grippers

7.1 Introduction

The robot uses grippers to transfer lab ware or to use lab ware. At each end of the robot arm, there is a gripper mounted (see Figure 26). For controlling the motions of each gripper, a gripper controller is used. The type of the gripper is LEHF20K2-48-R86P5 and the type of the gripper controller is LECP6. The gripper's motions have to be coordinated with robot motions, thus the gripper controllers are connected with robot controller. The communication between robot controller (FS100) and gripper controllers is achieved based on I/O signals, with the robot controller sending the controlling singles. The connection can be seen in Figure 59. More details about the connection can be seen in "Appendix -- A.3". Programming for grippers includes two aspects: one part is setting motion steps on the gripper controller; the other part is about robot programs running on the robot controller. These robot programs are mainly used to send I/O signals to gripper controllers and detect output signals sent from gripper controller.

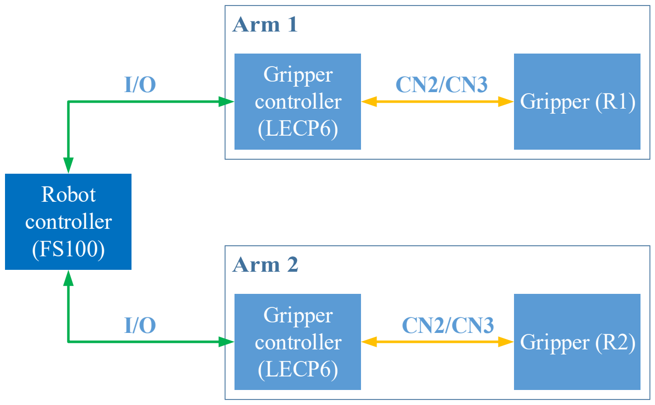
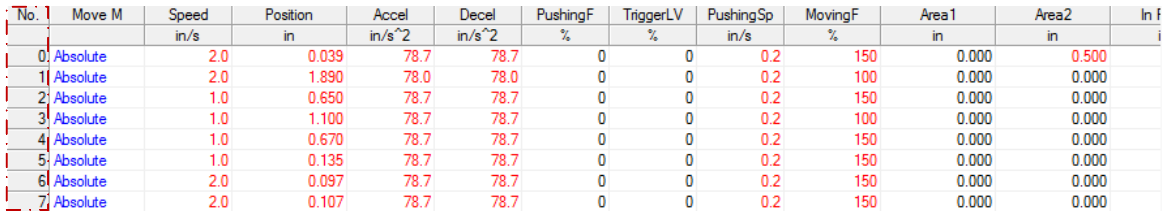


Figure 59. Connection between robot controller and gripper.

7.2 Configuration of the Gripper Controllers

In order to control grippers by the FS100 controller, parameters (including speed, position and so on) should be set and downloaded to LECP6 controller. Figure 60 shows the part of the parameters for controlling grippers.



No.	Move M	Speed	Position	Accel	Decel	PushingF	TriggerLV	PushingSp	MovingF	Area1	Area2	In F
		in/s	in	in/s^2	in/s^2	%	%	in/s	%	in	in	i
0	Absolute	2.0	0.039	78.7	78.7	0	0	0.2	150	0.000	0.500	
1	Absolute	2.0	1.890	78.0	78.0	0	0	0.2	100	0.000	0.000	
2	Absolute	1.0	0.650	78.7	78.7	0	0	0.2	150	0.000	0.000	
3	Absolute	1.0	1.100	78.7	78.7	0	0	0.2	100	0.000	0.000	
4	Absolute	1.0	0.670	78.7	78.7	0	0	0.2	150	0.000	0.000	
5	Absolute	1.0	0.135	78.7	78.7	0	0	0.2	150	0.000	0.000	
6	Absolute	2.0	0.097	78.7	78.7	0	0	0.2	150	0.000	0.000	
7	Absolute	2.0	0.107	78.7	78.7	0	0	0.2	150	0.000	0.000	

Figure 60. Example of program for gripper.

In the left side of Figure 60, the numbers relate with gripper motion steps, which can be chosen by setting specific I/O signals. For example, if the related input I/O signals are equant to "0", the No. 0

step is selected. When a driving signal is set, gripper fingers will be moved to position “0.039 inch”. If the specific I/O signals are equant to “3”, the No. 3 step is selected and then gripper fingers will be moved to position “1.100 inch”. The step and driving signals will be set by robot jobs. In order to monitor the grippers’ status, indicating if the grippers’ fingers reach the set position successfully, the output signals of gripper controllers will also be monitored by robot programs.

7.3 Robot Programs for Controlling Grippers

For operating grippers, robot jobs had been built running on FS100 controller. And these jobs will output I/O signals to LECP6 controllers and also read the I/O signals output from LECP6. Figure 61 gives the flow chart of the robot jobs for driving grippers. The list of robot jobs for driving gripper 1 (left arm) and gripper 2 (on right arm) can be seen in Table 21 and Table 22 in Appendix B.1.

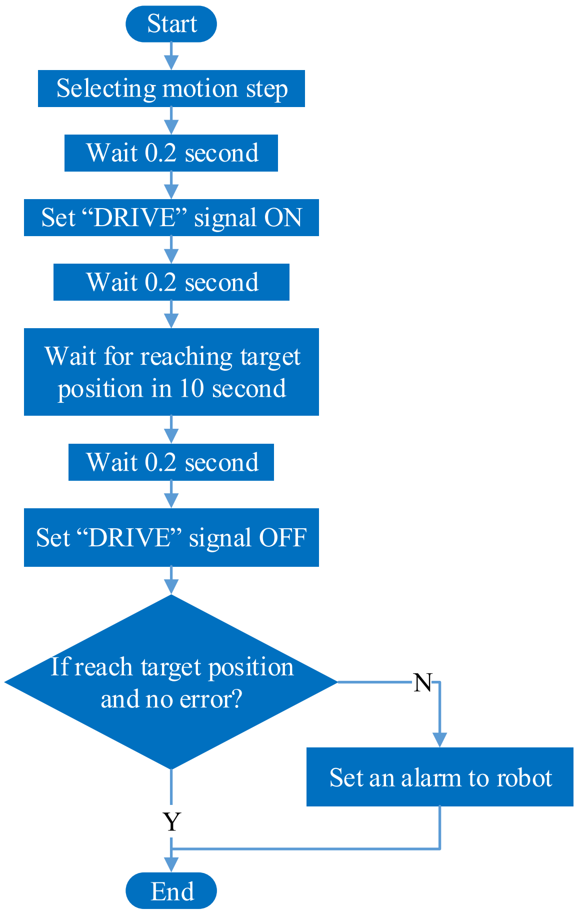
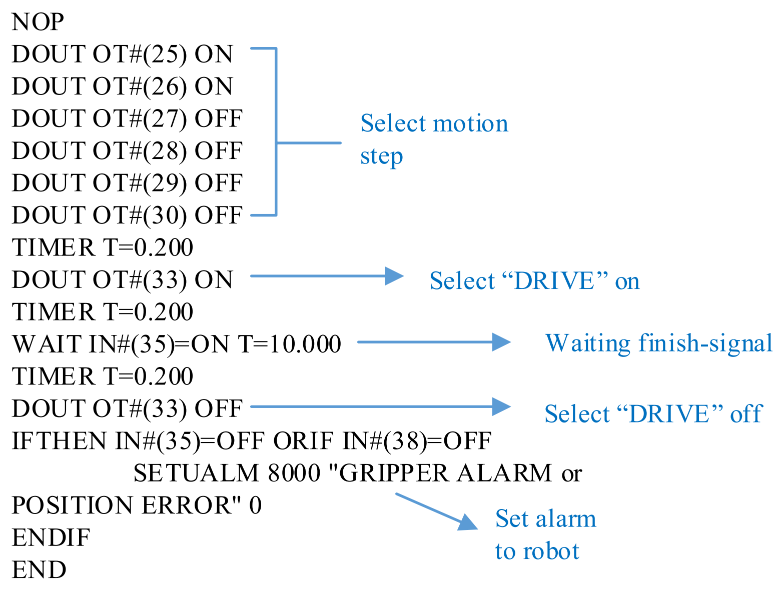


Figure 61. Flow chart of robot jobs for driving gripper.

First, the robot program will set values to 6 bits I/O ports, which are used to select a gripper motion step, from 0 to 63. As the gripper controller only supports 64 motion steps, 6 bits I/O signals are compatible. After waiting for 0.2 second (LECP6 cannot be accessed very frequently), the “DRIVE” signal will be set ON. Then the gripper fingers will start to move. After waiting another 0.2 second, the robot job starts to monitor the output signals from LECP6 controller indicating whether fingers reach the set position. The robot job will only monitor the signals for no more than 10 seconds, in order to avoid moving robot arms away before gripping an object completely. Normally, the gripper can reach the set position in 10 seconds, except some errors happened (e.g. if the size of an object is too bigger than the set value, the gripper will be unable to reach the set position, and be blocked by the object.). The limitation of 10 second, which is much longer than the time needed for grippers to finish any movements, is necessary for avoiding programming blocking. In this way, if some errors happen, the correct output signals from LECP6 controller will not be able to be monitored, and thus the robot job can terminate waiting the signal and go on to execute the following codes.

After the fingers reach the set position or the 10 seconds limitation is finished, the "DRIVE" signal will be set OFF. Then output signals of LECP6 will be checked. If the grippers do not reach the set positions or encounter with other errors, an alarm will be raised to stop the robot. There two main reasons for the errors: the grippers are not supplied with power or the gripper cannot grip the object due correctly. The faults of grippers are possible to cause the lab ware destroyed. In this case, it is critical to stop the robot. This is achieved by setting an alarm. If the gripper fingers are not moved to the target position, an alarm will be set to FS100 controller. The user will see the alarm message and also the robot will terminate moving immediately. Unless the alarm is reset, the robot cannot be started again. Figure 62 shows the example of a robot job for driving grippers. In this example the step 3 is selected ("000011" = "3").



```
NOP
DOUT OT#(25) ON
DOUT OT#(26) ON
DOUT OT#(27) OFF
DOUT OT#(28) OFF
DOUT OT#(29) OFF
DOUT OT#(30) OFF
TIMER T=0.200
DOUT OT#(33) ON
TIMER T=0.200
WAIT IN#(35)=ON T=10.000
TIMER T=0.200
DOUT OT#(33) OFF
IFTHEN IN#(35)=OFF ORIF IN#(38)=OFF
                SETUALM 8000 "GRIPPER ALARM or
POSITION ERROR" 0
ENDIF
END
```

Figure 62. Example of a robot job for driving grippers.

Chapter 8 Method for Integrating Software of Analytical Instruments

When samples are transferred to the LC-MS or GC-MS instrument by the robot, an analysis process needs to be started automatically. Thus, a high degree of automation will be gained, without interrupting the operators from their work on hand. This is achieved by integrating the software systems of the LC-MS and GC-MS instruments with the SAMI Workstation EX Software system (SAMI EX, Beckman Coulter Inc.) separately. SAMI software is user friendly, which reduces the need of the operators' programming skills, and is used for process scheduling. However, the integration of software is difficult due to the proprietary software interfaces of the LC-MS and GC-MS instruments. To avoid such limitations, the integration has been realized by employing software resources of the Microsoft Windows operating system. Reusing commercial off-the-shelf Microsoft Windows software applications for integration purposes had been researched using Java technology [131]. C sharp (C#), which is a fairly well-documented technology enabling rapid development of applications running on the .Net Framework [132], is used to prepare the programming interface for integrating the software. The program interface consists of two parts. One part is used to actualize the controls of the LC-MS or GC-MS instrument based on its graphical user interface. XML has been widely used and is considered as a powerful tool for system integrations [133]. The other part is for dealing with XML data, in order to integrate with SAMI software. In the application, XML data encoded with command information will be sent from the SAMI module through the local network. In this way, the software has been integrated to SAMI software.

8.1 Software Integration Method Based on GUI

Usually the application software of an instrument has a graphical user interface (GUI) for a user to operate it, such as clicking some buttons using a mouse or inputting some text using a keyboard. Such a user interface is very important, as the user can control the instrument without the need to know how the code of its software runs. For the integration, knowing how the code works will be helpful, as this can make the integration easier. However, the codes of a commercial software are likely to be kept secret, only providing the APIs (Application Programming Interfaces) for a system integration purpose. In this case, when there are no documents detailing the way of using the APIs or no available APIs, it will be very confined to make integrations or even impossible. However, if the software integration was achieved through the graphical user interface, there will be no such problems. Instead, it only needs to know how to use a graphical user interface. Another advantage is that a commercial software always has a user manual detailing how to use its graphical user interface. Integrating the software of an analytical instrument is also necessary, as an analysis will be able to start automatically and the analysis process can be monitored. However, the integration of the software is difficult, due to the proprietary software interfaces. For integrating the software, a program interface based on the graphical user interface has been built, which consists of two modules: the XML module and the GUI module [134]. Figure 63 shows the structure of the program interface for integrating software of LC-MS with SAMI software. The program interface for integrating the software of GC-MS has a similar structure.

SAMI Workstation EX Software incorporates optimized planning and data-driven dynamic rescheduling for pre validated schedules and run-time flexibility and uses graphical interface to make assay design easier [129]. The integration between SAMI software and the LC-MS software has been realized based on the communications between the SAMI module and the program interface: sending/receiving XML data through the local network.

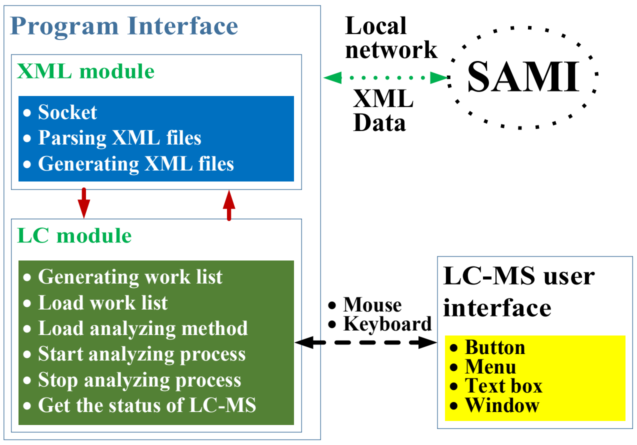
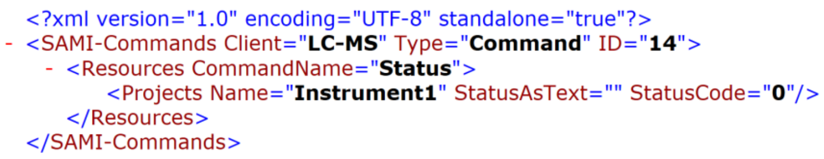


Figure 63. Structure of program interface for integrating LC-MS.

The XML module is used for dealing with XML data, which will be sent through the local network. The XML data sent from the SAMI module is encoded with commands separately, such as start, stop or status. Figure 64 shows an example of a XML data encoded with the "Status" command, for asking the status of the instrument. The XML data fed back from the program interface is encoded with the results about executing the commands. After commands in the XML data is parsed, related methods in the LC module will be called. The LC module consists of a sequence of methods for dealing with the LC-MS software system, including generating a work list, loading a work list, loading an analyzing method, starting an analyzing process, stopping an analyzing process, and receiving a status. After the XML data as shown in Figure 64 was received and parsed, the method for getting status will be called to obtain the status of the LC-MS: busy, idle or error.



```
  <?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <SAMI-Commands Client="LC-MS" Type="Command" ID="14">
  - <Resources CommandName="Status">
        <Projects Name="Instrument1" StatusAsText="" StatusCode="0"/>
    </Resources>
  </SAMI-Commands>
```

Figure 64. An example of XML data, encoding with the command for getting status.

By clicking buttons or menus in a graphical user interface using a mouse or inputting texts using a keyboard, an operator can control an instrument. The LC module is designed to carry out the control of the LC-MS in a similar way by simulating the mouse motions and keystrokes. While the simulation of the mouse motions has two contexts: moving the mouse pointer to certain position on the computer screen and simulating left button single click motion, the programs for simulating keystrokes are used to send texts or hotkeys to the graphical user interface. Coordinate is important. All objects such as buttons and mouse pointer have their specific coordinates on a screen. When operating the graphical user interface, dialog boxes are often used for the interactions. Thus, methods for dealing with windows are also necessary. So, there are four elements for controlling the user interface: mouse motion simulation, keystroke simulation, object's coordinate and windows handling (see Figure 65).

When there is several application software running on the same computer, the specific graphical user interface may not be shown in the computer screen, as covered by other software interfaces. However, if the graphical user interface was not shown, it cannot be operated completely. Thus, showing the user interface and checking if it was shown on the computer screen are critical. Dealing with a window can been achieved through its handle, and in the "*user32.dll*" file of the Microsoft Windows there are ample functions available, such as *Findwindow*, *SwitchToThisWindow*, *WindowFromPoint* and *ChildWindowFromPoint*. When simulating the mouse click motions to some components (e.g. a button), their positions should be known. Showing the user interface in the full screen mode will be useful, as in this case components of the user interface will have a fixed coordinate relative to the screen and thus these coordinates can be easily recorded in a XML file for the configuring purpose. However, dealing with the dialog box will be more difficult, as the dialog box can be shown at different positions on the screen and thus components of the dialog box are not fixed relative to the screen. And for such components, their coordinates relative to the dialog box were recoded. Keystrokes can be achieved by the "*Send*" method and "*SendWait*" method in the "*Sendkeys*" class of the .NET Framework library.

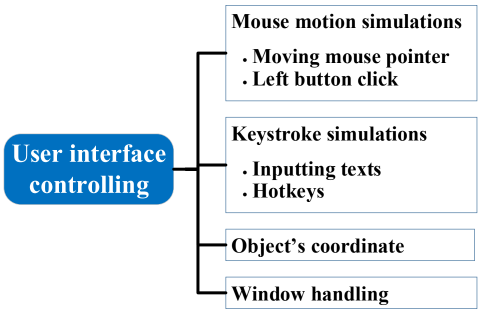


Figure 65. Elements for controlling a graphical user interface.

When simulating mouse click motions to the components on a dialog box, coordinate converting will be necessary, as the components' coordinates are relative to the dialog box and the mouse coordinate is relative to the screen. For coordinate converting, "*ScreenToClient*" and "*ClientToScreen*" functions are available. The "*mouse_event*" function and the "SendInput" function can be used to move mouse points to the specific position and to simulate button clicks. The coordinate of the mouse absolute movement is specific to values between 0 and 65,535. A calculation for getting the mouse movement position from the component coordinate is also necessary. Figure 66 shows the process for simulating the mouse motions for clicking a component on a dialog box.

In Figure 66, getting the handle of the dialog box is necessary, because the handle is needed for coordinate converting. The (Xc, Yc) is the recorded component's coordinate relative to the dialog box, the (Xs, Ys) is the component's coordinate relative to the screen, and the (Lx, Ly) is the coordinate for moving the mouse pointer on the component. For calculating Lx and Ly, the screen width Ws and height Hs (in pixel unit) were used.

By seeing the text message or color information showing on the screen, the operator can know the status of the instrument: busy, idle or error. The programs can also obtain the text message and the colors shown on the screen. For obtaining the text message, the "*GetWindowText*" function can be used, which is not only used to obtain the title of a window, but also can be used to get the text of the other components, such as a text box, a label or a button. For obtaining the color at a specific position on a window, the "*GetPixel*" method is used. For using this method, a "*Bitmap*" object of the screen picture has to be created, which is done in the following way. First, send the hotkey "*Print screen*" using "*SendWait*" or "*Send*" method to capture the current screen. Then read the captured data of the current screen from the clipboard to build a "*Bitmap*" object. After this, the pixel at specific position can be obtained by using "*GetPixel*" method.

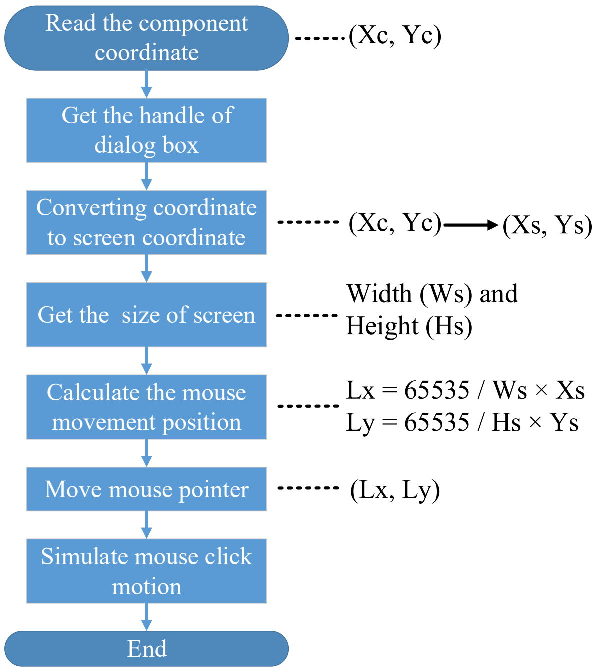

Read the component coordinate
(Xc, Yc)
Get the handle of dialog box
Converting coordinate to screen coordinate
(Xc, Yc) → (Xs, Ys)
Get the size of screen
Width (Ws) and Height (Hs)
Calculate the mouse movement position
Lx = 65535 / Ws × Xs
Ly = 65535 / Hs × Ys
Move mouse pointer
(Lx, Ly)
Simulate mouse click motion
End

Figure 66. Process for clicking a component on a dialog box.

8.2 LC-MS Program Interface

This section is about the program interface (LC-interface) for integrating the data-acquisition software of LC-MS instrument with SAMI software. The integration of software is necessary, as in this way an analysis process by the LC-MS instrument is able to start and monitor automatically by SAMI software, instead of by a user. On one side, the LC-interface will communicate with SAMI software, receiving commands and sending replies. On the other side, after received SAMI commands, the LC-interface will carry out related "operations" to the data-acquisition software, such as loading a worklist, starting a measurement or reading the status of LC-MS instrument. However, the integration of software is difficult, due to the proprietary software interfaces. Parts of such "operations" are able to be realized through the APIs of the software of LC-MS instrument, e.g. creating a worklist. While they others, which cannot be realized in this way due to the lack of available APIs, e.g. loading a worklist, have been realized by simulating mouse and keyboard inputs and capturing the information showing on the GUI. The realizations of the "operations" will be discussed in the following parts.

8.2.1 Creating LC-MS Worklists

A worklist is a file, which includes the information of a list of samples, such as the sample name, sample position, method for acquiring data and method for analyzing data. This file is critical, as only after this file is loaded to the data acquisition software of LC-MS, the measurement can be started. The program for creating worklist is based on the APIs in "*Interop.AGTPICWORKLISTLib.dll*" file of LC-MS software. The program flow chart for creating a worklist can be seen in Figure 67. In the process of creating a worklist, the class "*AgtPicWklDataManager*" is required to instantiate an object. And this object will be used to do initialization, creating an empty worklist, add information to the worklist, set attributes, save worklist and then close worklist. When an empty worklist is created, some important information will be added, such as the path of methods, the path of data and the plate name. Then, the information of samples will be added one by one. Besides, the column values, e.g. the length and width, should also be set. When a name is given to the worklist and a path for saving the worklist is set, the worklist can be save. After this, the worklist has to be closed to finish the worklist creating process.

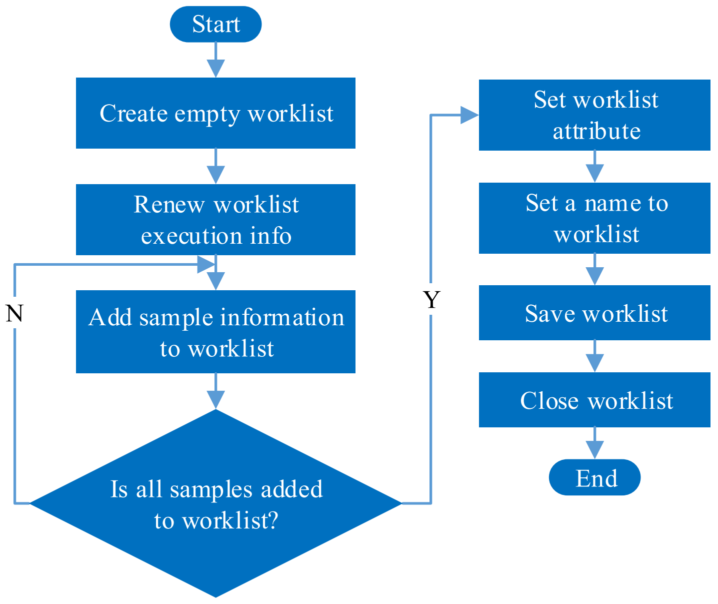


Figure 67. Flow chart of creating worklist.

8.2.2 Load LC-MS Worklist and LC-MS Method

When starting a new measurement, normally a worklist has to be loaded to the data-acquisition software of LC-MS. The function of loading a worklist to the software of LC-MS is achieved based on simulating mouse and keyboard inputs to the graphical user interface. Figure 68 shows the program flow chart of loading a worklist. In order to send hotkey or characters to the software of LC-MS, it is necessary to get the handle of the GUI first, which is the identifier of the GUI. Then the GUI can be set to the foreground to receive inputs. After sending the hotkey, which is used to open the dialog of loading worklist, the program will check the appearances of the dialog. If not appear, the program will be interrupted and return an error message. If the dialog appears, its textbox will be focused to receive input characters, including the path and the name of the worklist. After inputting of these characters, another hotkey will be sent to close the dialog. If the dialog is closed, a success message will be sent. Otherwise, an error message will be sent.

For different samples, different data acquiring methods are possible to be used. Thus, the program interface needs to be able to load a LC-MS-Method. The program for loading a method is also based on user interface. And it has a similar structure with the program for loading a worklist.

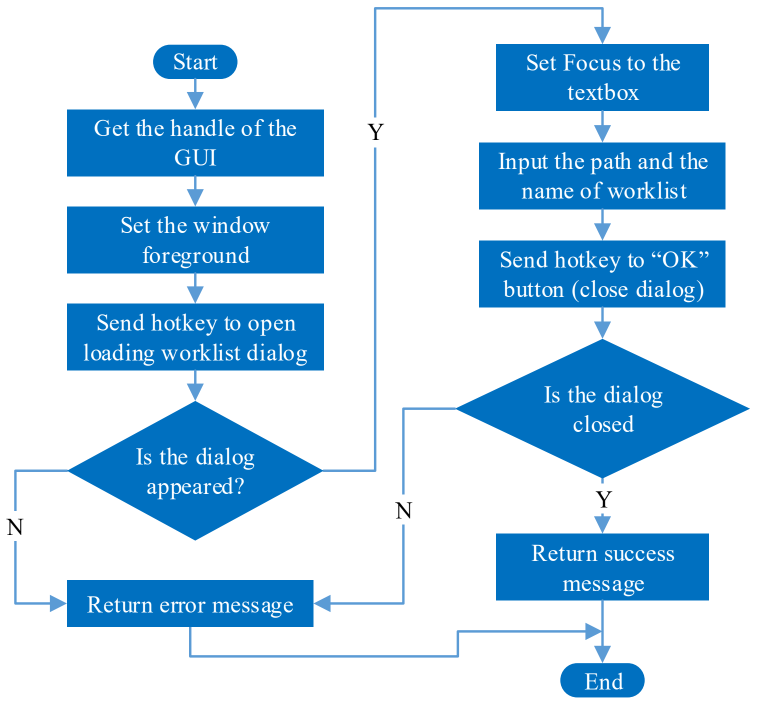

Start
Get the handle of the GUI
Set the window foreground
Send hotkey to open loading worklist dialog
Is the dialog appeared?
Y
N
Return error message
Set Focus to the textbox
Input the path and the name of worklist
Send hotkey to "OK" button (close dialog)
Is the dialog closed
Y
N
Return success message
End

Figure 68. Flow chart of loading worklist.

Figure 69 is the example of loading a worklist file to the LC-MS software system. First, the program will send the hotkey ("*Ctrl* + *w*") to the user interface of the LC-MS software by using "*SendWait*" method. Then, the window shown in Figure 69 is expected to be pop-up.

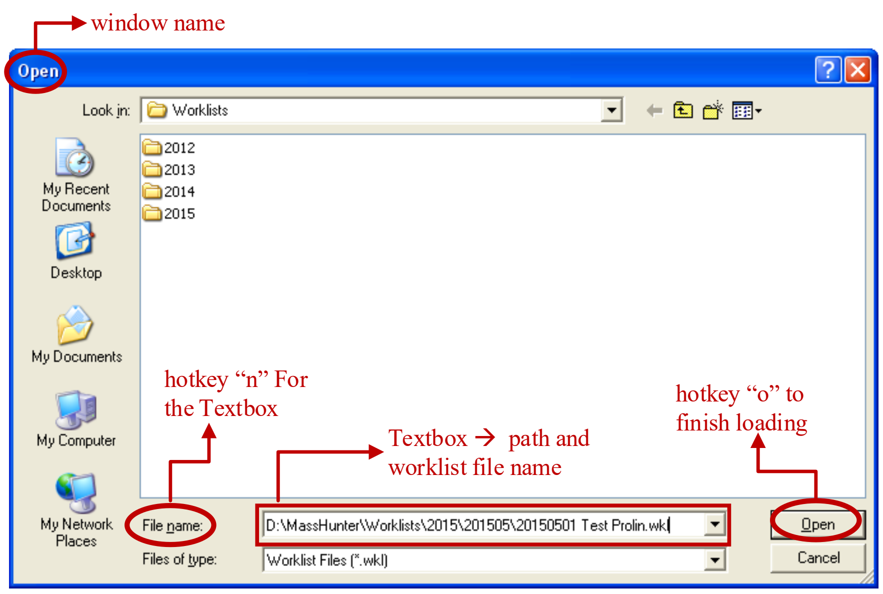

window name
Open
Look in: Worklists
2012
2013
2014
2015
My Recent Documents
Desktop
My Documents
My Computer
My Network Places
hotkey "n" For the Textbox
Textbox → path and worklist file name
hotkey "o" to finish loading
File name: D:\MassHunter\Worklists\2015\201505\20150501 Test Prolin.wkl
Files of type: Worklist Files (*.wkl)
Open
Cancel

Figure 69. Example of loading a worklist.

However, if the GUI did not receive this hotkey successfully, the window will not arise. Thus, it is necessary to check that if this window has appeared. The checking is done by using the "*FindWindow*" function, with window name "*Open*" as the parameter. If the "*Open*" window was found, the hotkey ("*n*") will be sent to this window to set focus to the "*Textbox*" and then input the path and the name of the worklist file to it by using "*SendWait*" method. Then, the hotkey "*o*" is sent to this window to finish the loading of the worklist. If the worklist was loaded successfully, the "*Open*" window will be closed.

However, if the path or the file name was not correct or the worklist file was not found, the "*Open*" window cannot be closed. Thus, after sending the hotkey "*o*", the program will check again. If the "*Open*" window is still found, an exception will be thrown to inform the error of loading worklist.

8.2.3 Starting and Aborting LC-MS Measurements

When the samples are put into the LC auto sampler, a worklist for the samples have been created and loaded to LC-MS software, and a LC-MS-method has been loaded (if necessary), then it is ready to start the measurement. The program for starting the measurement is based on APIs in "*Interop.AGTPICWORKLISTLib.dll*", "*Interop.AGTPICENGINELib.dll*" and "*Interop.AGTPICL-AUNCHERLib.dll*". Figure 70 shows the flow chart of the program for starting a measurement.

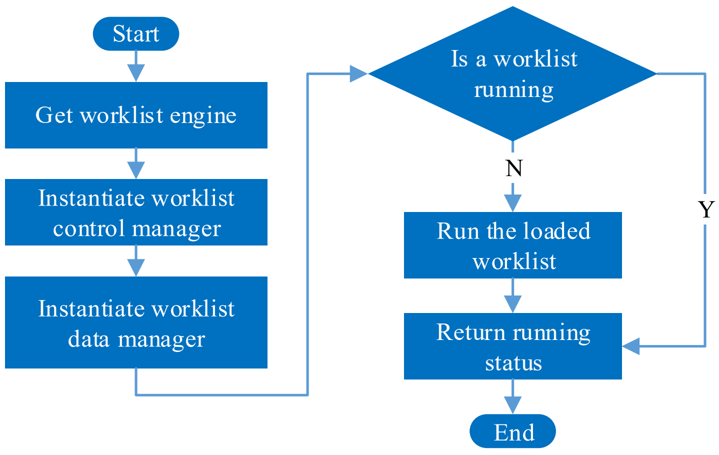


Figure 70. Flow chart of starting a measurement.

According the APIs, first the "*engine*" of worklist will be got, and then be used to instantiate the "*worklist control manager*", which provide the APIs for running a worklist and for aborting the execution of a worklist, and "*worklist data manager*", which includes the data of a worklist. Before running a worklist, the program will check if the other worklist is running and return the running status. If it is necessary, the measurement can also be aborted. The program for aborting the measurement has a similar flow chart with the program for starting a measurement. The only different is that, after checking the worklist running status, an API for aborting execution of a worklist is used, instead of the API for running a worklist.

8.2.4 Pausing and Resuming LC-MS Measurements

When the measurement is running, the user can also stop the measurement temporarily and then start it again. The program for pausing a worklist is based on user interface: get the handle of the GUI, set GUI foreground and send hotkey to pause the execution of the worklist. When the measurement is paused, it can be resumed again. The program for resuming a worklist is based on APIs of LC-MS software system. Figure 71 shows the flow chart of the program for resuming the measurement. It is in the similar way with that of the programs for starting or stopping a worklist. If there is no worklist running, the program will resume the execution of the worklist by using the APIs.

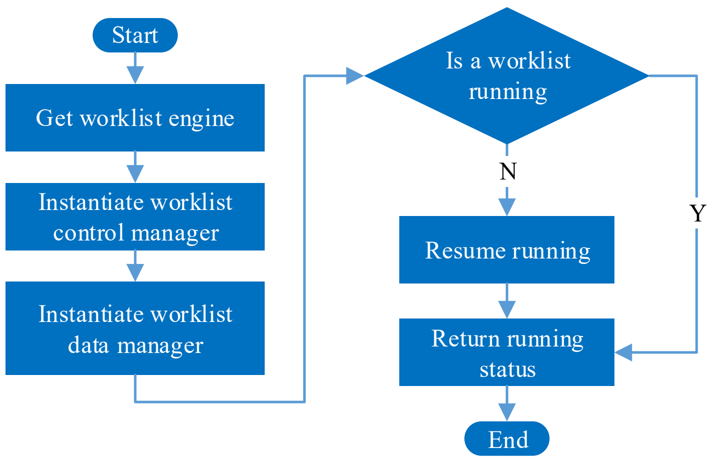


Figure 71. Flow chart of resuming the measurement.

8.2.5 Reading the LC-MS Status

Knowing the status (busy, idle or error) of the instrument is required. For obtaining the status, the programs will read the text message and the colors shown on the monitor screen. The "*GetWindowText*" function can be used to read text message from a window, a textbox, a label or a button. The "*GetPixel*" method can be used to obtain the color at a specific position on a window. The program for reading the status of LC-MS instrument, is based on the graphic user interface. Figure 72 shows the flow chart of the program for reading the text indicating the status of LC-MS instrument. After getting the handle of GUI and setting it to the foreground, the handle will be used to read its text.

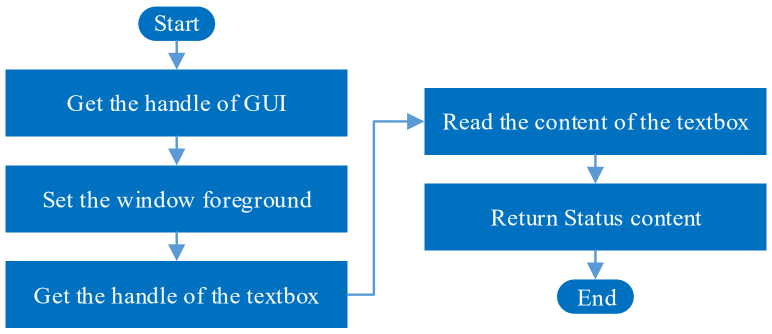

Start
Get the handle of GUI
Set the window foreground
Get the handle of the textbox
Read the content of the textbox
Return Status content
End

Figure 72. Flow chart of reading LC-MS status.

Figure 73 gives an example capturing parameters and colors. In the UI, the instrument status is shown by text parameters and colors to the operator. For knowing the status, the program will capture the text parameters and colors from the UI. The frame of the "Test window" was prepared, but its content (parameters and status colors) was captured from the UI by the program. The text parameters were obtained by using "*GetWindowText*" function, and the status colors (such as green, gray and yellow) were captured by using the "*GetPixel*" method. For capturing the text or the color, target (e.g. textbox) position should be given in advance. Other windows should not cover the target area of the UI, otherwise the captured text or color will not be correct.

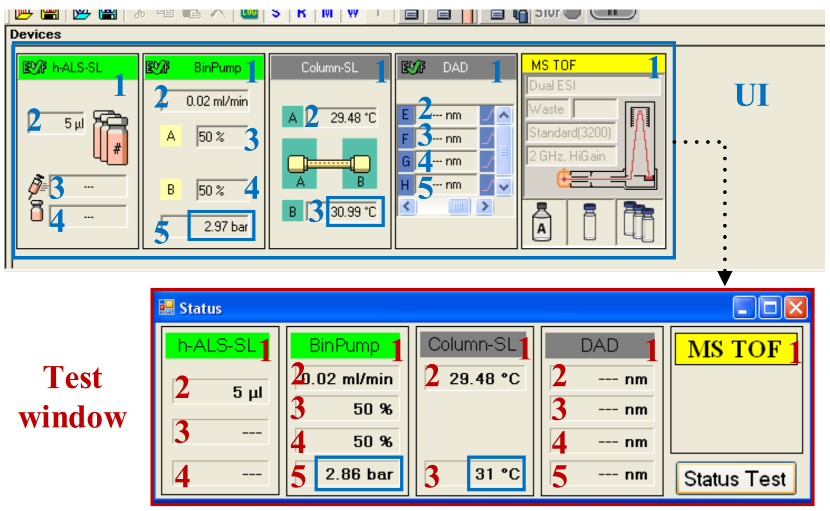


Figure 73. Example of capturing parameters and status colors.

In Figure 73, the target texts and colors in the UI and the captured texts and colors in "Test window" were marked with numbers. It can be seen that there were two parameters having different values between the UI and the "Test window". E.g., it is "2.97 bar" in the UI, while it is "2.86 bar" in the "Test window". This is because these parameters are changing with time, and those shown in "Test window" were values at the moment when they were captured. In order to avoid misunderstanding, it should be pointed out that the "Test window" in Figure 73 is only used to show how the program works, but not an essential part of the program. The program only needs to capture the texts and colors to analyze the status of the instrument, and it is not necessary to show them to the user.

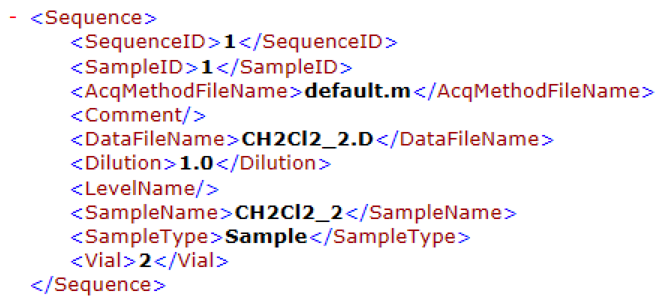
8.3 GC-MS Program Interface

This section is about the program interface (GC-interface) for integrating the data acquisition software of GC-MS instrument with SAMI software, which is a different software with that of LC-MS instrument, different GUI and different APIs. But the program interface is built in a similar way, with part of program interface built based on simulating "operations" to the GUI and the other part based on the APIs. The program of reading the status of GC-MS instrument is similar with the program of reading the status of GC-MS instrument, and will not discussed in this section.

8.3.1 Creating GC-MS Sequence

A sequence is a file in XML format, including the information of a list of samples, such as the sample name, sample position and method for acquiring data (see Figure 74). This file is critical, as only after this file is loaded to the GC-MS software, a measurement can be started.

Before staring a measurement, normally a sequence file should be created. As a sequence file is a XML file, so creating a GC-MS sequence is to create a XML file with certain tree elements. And the information about the samples will be included in the tree elements. The process for building a sequence can be seen in Figure 75. Firs, the name of the sequence and the path for saving it will be given. Then the information of each sample will be added to the sequence.



```
- <Sequence>
    <SequenceID>1</SequenceID>
    <SampleID>1</SampleID>
    <AcqMethodFileName>default.m</AcqMethodFileName>
    <Comment/>
    <DataFileName>CH2Cl2_2.D</DataFileName>
    <Dilution>1.0</Dilution>
    <LevelName/>
    <SampleName>CH2Cl2_2</SampleName>
    <SampleType>Sample</SampleType>
    <Vial>2</Vial>
  </Sequence>
```

Figure 74. Information of a sample in a sequence.

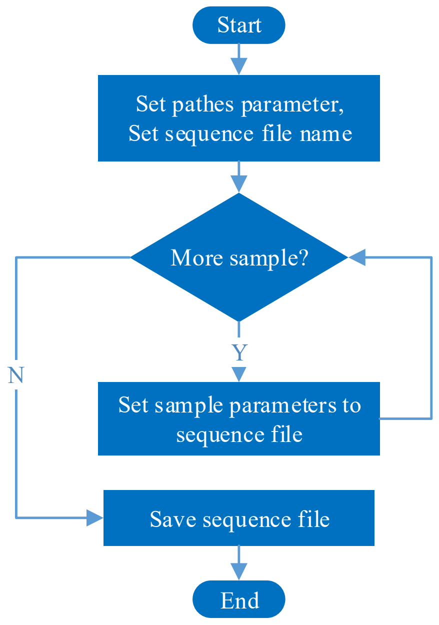

Start
Set pathes parameter, Set sequence file name
More sample?
N
Y
Set sample parameters to sequence file
Save sequence file
End

Figure 75. Flow chart of creating sequence.

8.3.2 Loading GC-MS Sequence and GC-MS Method

When start a new measurement, normally a sequence is need to be loaded to GC-MS software system. For different samples, different GC-MS methods, includes the parameters for acquiring samples, are possibly used. Thus, the program interface also needs to be able to load a GC-MS Method. The program for loading a sequence or a GC-MS method is based on APIs of the software, which uses some instructions to execute operations. For example, the "*Loadsequence*" instruction can be used to load a sequence, with the sequence name and its path as the parameters. And for loading a method, the "*Loadmethod*" instruction, with the method name and its path as the parameters, will be used. Figure 76 shows the flow chart of the program for loading a sequence. First, a channel will be registered with 5977 as the channel number. Then the remote server will be initialized with the IP address of the data-acquisition software of GC-MS as the parameter. Thus, the API ("*DDEExecute*") can be used to execute "*Loadsequence*" and "*Loadmethod*" instructions.

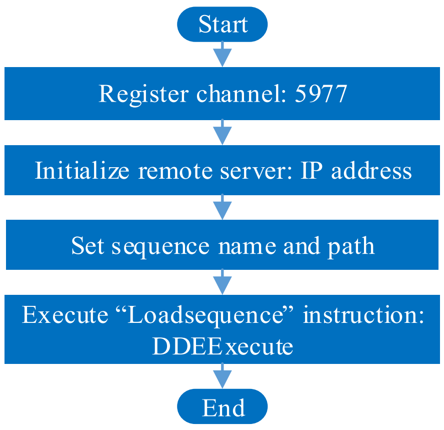

Start
Register channel: 5977
Initialize remote server: IP address
Set sequence name and path
Execute “Loadsequence” instruction: DDEExecute
End

Figure 76. Flow chart of loading a sequence.

8.3.3 Starting and Aborting GC-MS Measurements

When the samples are put into the GC auto sampler, a sequence for the samples have been created and loaded to GC-MS software, and the GC-MS-method has been loaded (if necessary), then it is ready to start the measurement. If it is necessary, the measurement can also be aborted. The programs for starting and aborting a measurement are based on simulating “operations” to the graphical user interface. The flow chart of the program for starting a measurement can be seen in Figure 77. First, it needs to get the handle of the GUI of the data-acquisition software of GC-MS instrument, in which there is a button for showing the “Start Sequence” dialog. After getting the handle of this button, a “click” motion can be simulated to it and then a dialog named as “Start Sequence” should be appeared (as shown in Figure 78).

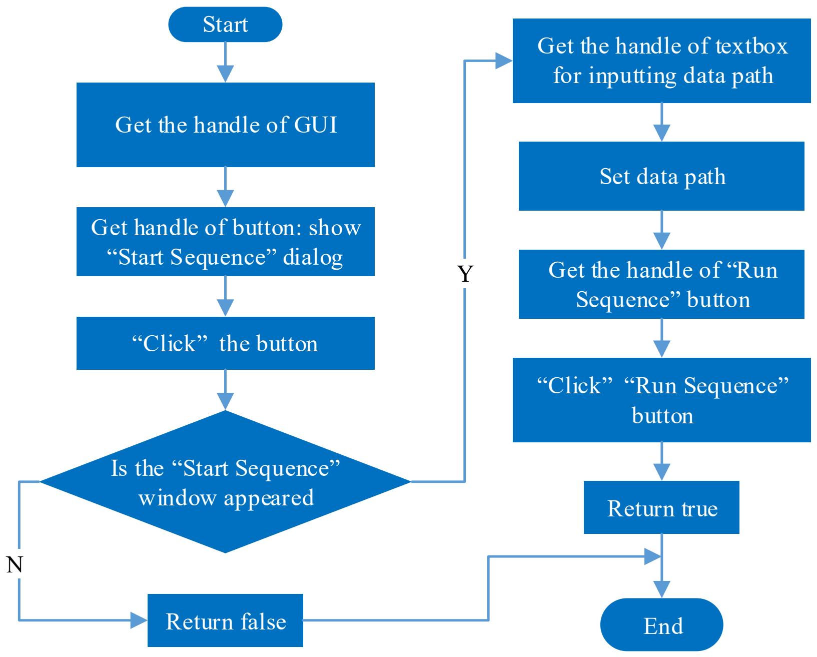
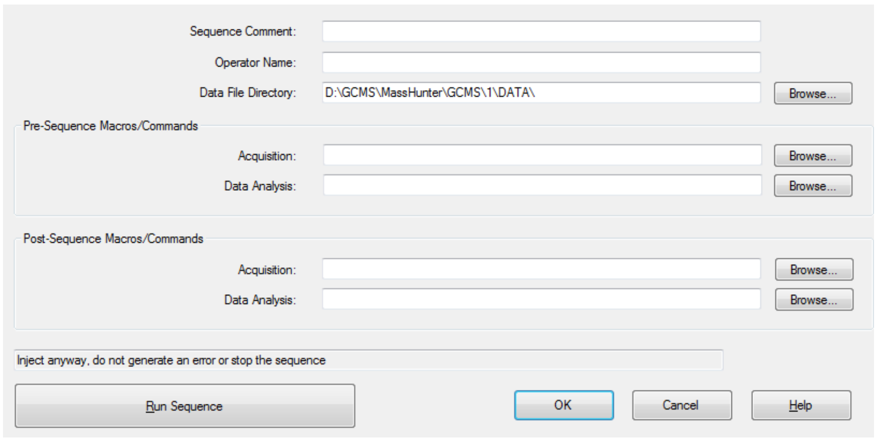

Start
Get the handle of GUI
Get handle of button: show “Start Sequence” dialog
“Click” the button
Is the “Start Sequence” window appeared
Y
N
Get the handle of textbox for inputting data path
Set data path
Get the handle of “Run Sequence” button
“Click” “Run Sequence” button
Return true
Return false
End

Figure 77. Flow chart of starting GC-MS measurement.

Through this dialog, the user can input some parameters for acquiring data manually, such as the path for saving measured data files. This can also be done automatically by the program interface. After the program confirms the appearance of the dialog, it will get the handle of the textbox ("Data File Directory") and input the path for saving the files of measured data. Then get the handle of the button ("Run Sequence"). After simulating the "click" motion to it, the measurement will start. The handle of a textbox or a button can be got through its position on the dialog. The program for aborting the measurement works in a similar way.



Sequence Comment:
Operator Name:
Data File Directory: D:\GCMS\MassHunter\GCMS\1\DATA\

Pre-Sequence Macros/Commands
Acquisition:
Data Analysis:

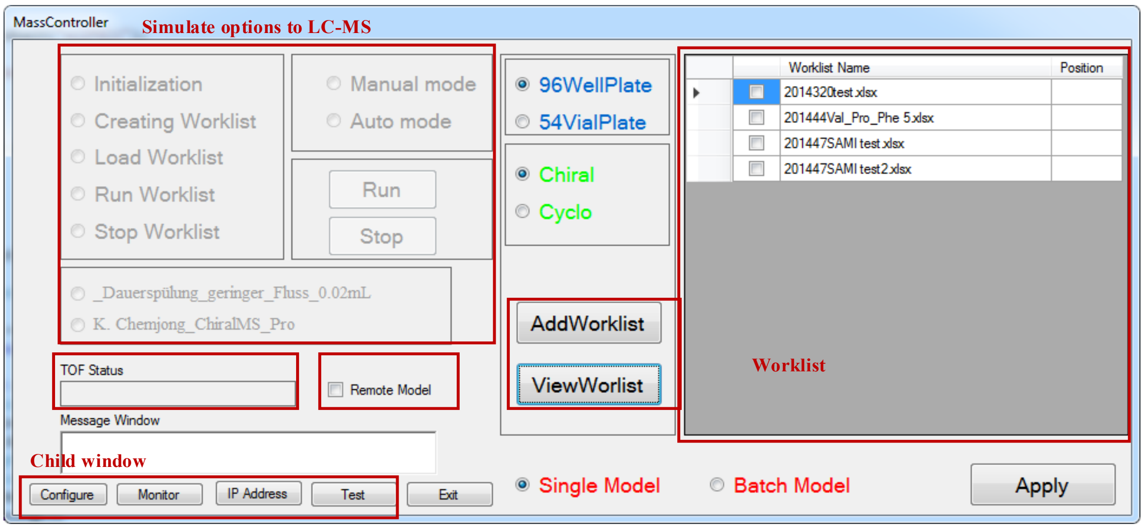
Post-Sequence Macros/Commands
Acquisition:
Data Analysis:

Inject anyway, do not generate an error or stop the sequence

Figure 78. The "Start Sequence" dialog of data-acquisition software of GC-MS.

8.4 Auxiliary Tools

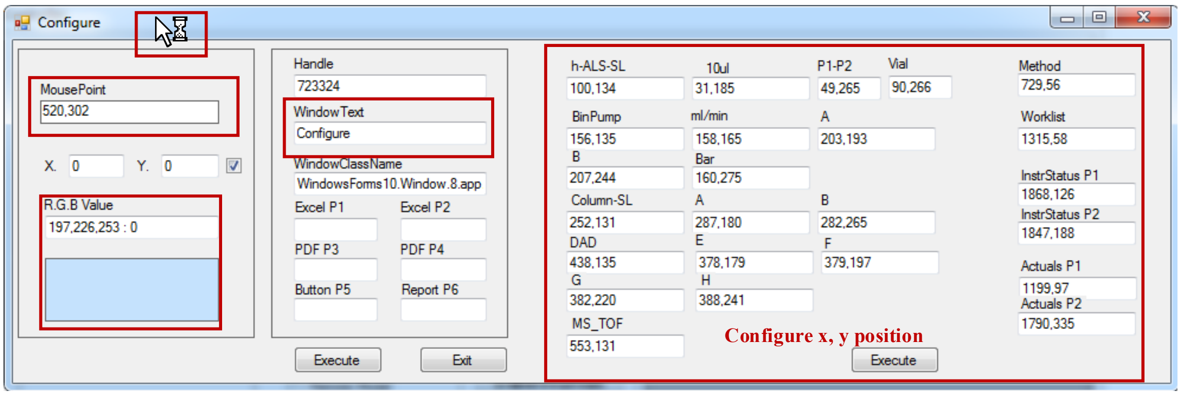
Even though the LC-interface is used to integrate the software of LC-MS with SAMI software, LC-interface also has a user interface for helping to test, make configuration and do other things. The main user interface of LC-interface is shown in Figure 79. By using it, the user can test the simulation options to the software of LC-MS (e.g. testing operations of "load worklist", "Run worklist" or "Load method"), open some child windows and so on.



Worklist Name	Position
2014320test.xlsx	
201444Val_Pro_Phe 5.xlsx	
201447SAMI test.xlsx	
201447SAMI test2.xlsx	

Figure 79. The main user interface of LC-interface.

Figure 80 shows the configure tool, which can be used to capture text and color information and to set the x, y positions of objects (such as a button or textbox) on the GUI of software of LC-MS. The text and color under the mouse pointer can be captured. It can be seen from Figure 80 that, the title of the window ("Configure") and the color of the title bar was captured (RGB: 197, 226, 253).



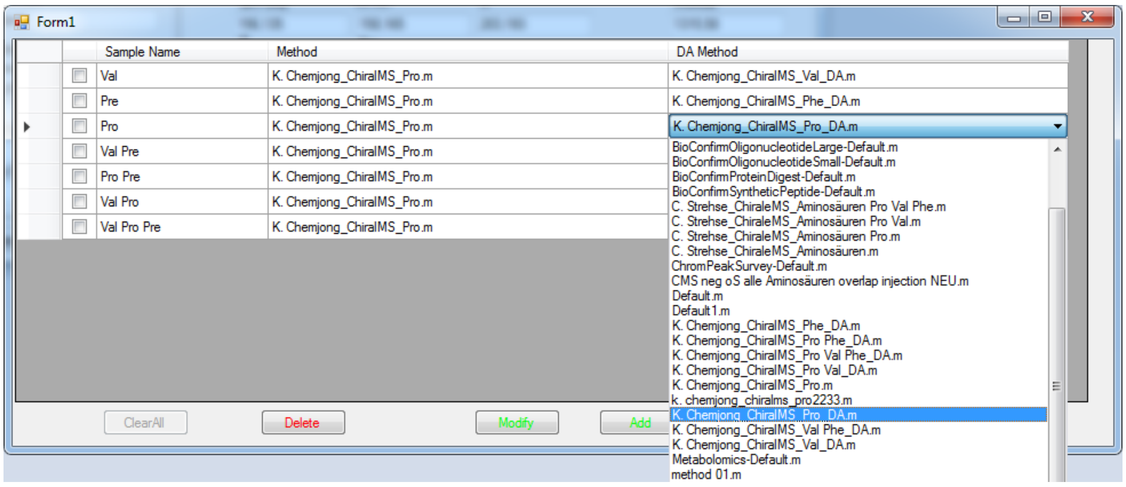
MousePoint	520,302
X.	0
Y.	0
R.G.B Value	197,226,253 : 0
Handle	723324
WindowText	Configure
WindowClassName	WindowsForms10.Window.8.app
Excel P1	
Excel P2	
PDF P3	
PDF P4	
Button P5	
Report P6	

h-ALS-SL	100,134
10ul	31,185
P1-P2	49,265
Vial	90,266
BinPump	156,135
ml/min	158,165
A	203,193
B	207,244
Bar	160,275
Column-SL	252,131
A	287,180
B	282,265
DAD	438,135
E	378,179
F	379,197
G	382,220
H	388,241
MS_TOF	553,131
Method	729,56
Worklist	1315,58
InstrStatus P1	1868,126
InstrStatus P2	1847,188
Actuals P1	1199,97
Actuals P2	1790,335

Configure x, y position

Figure 80. Configure tool.

Figure 81 shows the tool for adding the information (including sample name, method name of acquiring analyzed data and method name of processing analyzed data) of samples to a “Library”. Using it, the user can add new samples, delete samples or modify samples. And the user can choose a “Method” or “DA Method” from the list. When the tool window is loaded, the list for “Method” or “DA Method” will be renewed with the names of new methods added to the list automatically.

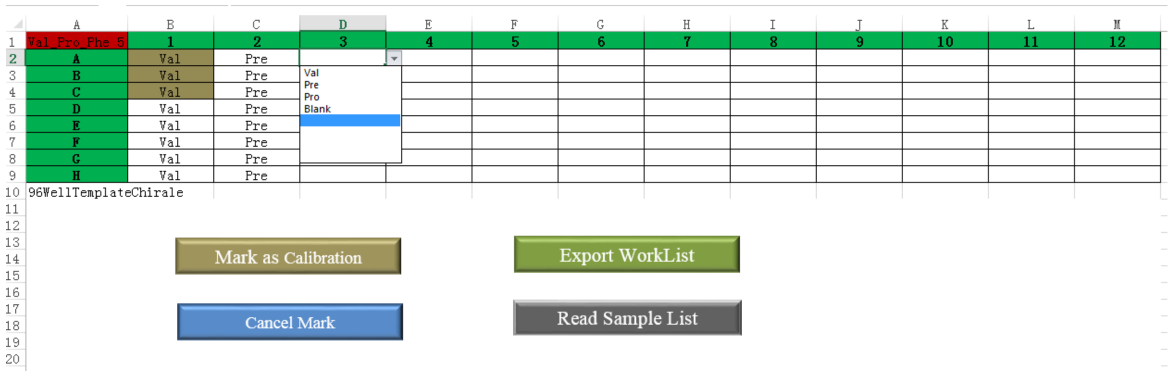


Sample Name	Method	DA Method
Val	K. Chemjong_ChiralMS_Pro.m	K. Chemjong_ChiralMS_Val_DA.m
Pre	K. Chemjong_ChiralMS_Pro.m	K. Chemjong_ChiralMS_Phe_DA.m
Pro	K. Chemjong_ChiralMS_Pro.m	K. Chemjong_ChiralMS_Pro_DA.m
Val Pre	K. Chemjong_ChiralMS_Pro.m	
Pro Pre	K. Chemjong_ChiralMS_Pro.m	
Val Pro	K. Chemjong_ChiralMS_Pro.m	
Val Pro Pre	K. Chemjong_ChiralMS_Pro.m	

BioConfirmOligonucleotideLarge-Default.m
BioConfirmOligonucleotideSmall-Default.m
BioConfirmProteinDigest-Default.m
BioConfirmSyntheticPeptide-Default.m
C. Strehse_ChiraleMS_Aminosäuren Pro Val Phe.m
C. Strehse_ChiraleMS_Aminosäuren Pro Val.m
C. Strehse_ChiraleMS_Aminosäuren Pro.m
C. Strehse_ChiraleMS_Aminosäuren.m
ChromPeakSurvey-Default.m
CMS neg oS alle Aminosäuren overlap injection NEU.m
Default.m
Default1.m
K. Chemjong_ChiralMS_Phe_DA.m
K. Chemjong_ChiralMS_Pro Phe_DA.m
K. Chemjong_ChiralMS_Pro Val Phe_DA.m
K. Chemjong_ChiralMS_Pro Val_DA.m
K. Chemjong_ChiralMS_Pro.m
k. chemjong_chiralms_pro2233.m
K. Chemjong_ChiralMS_Pro_DA.m
K. Chemjong_ChiralMS_Val Phe_DA.m
K. Chemjong_ChiralMS_Val_DA.m
Metabolomics-Default.m
method 01.m

Figure 81. Tool for building sample library.

Figure 82 shows the interface of VBA tool for helping to create a worklist for 96-well microplate. Benefiting of flexibility of Microsoft Excel software, the user can fit the information to each “well” easily. After the information is filled, clicking “Export Worklist” button will generate a worklist.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Val Pro Phe 5	1	2	3	4	5	6	7	8	9	10	11	12
2	A	Val	Pre										
3	B	Val	Pre										
4	C	Val	Pre										
5	D	Val	Pre										
6	E	Val	Pre										
7	F	Val	Pre										
8	G	Val	Pre										
9	H	Val	Pre										
10	96WellTemplateChirale												

Figure 82. VBA tool for helping to create worklist.

Chapter 9 Robot Program Interface

In order to integrate the robot system with the SAMI system, a robot program interface is built [135], which is wrote in C#. The robot program interface, called as R-interface, has two main functions: dealing with the commands from SAMI and dealing with the robot jobs. The commands are used to describe the tasks for the robot to do, e.g. pipetting solutions or transferring microplates. As it is discussed in Chapter 6, a lot of robot jobs (around 150 jobs or 150 motion elements[3]) have been built to realize robot motions for executing the tasks. Thus, how to manage and use the jobs will affect the effectivity and flexibility of the system. R-interface is a very important work of this dissertation, which mainly include five parts. Its architecture is shown in Figure 83. Details about R-interface will be discussed in the following parts of this chapter.

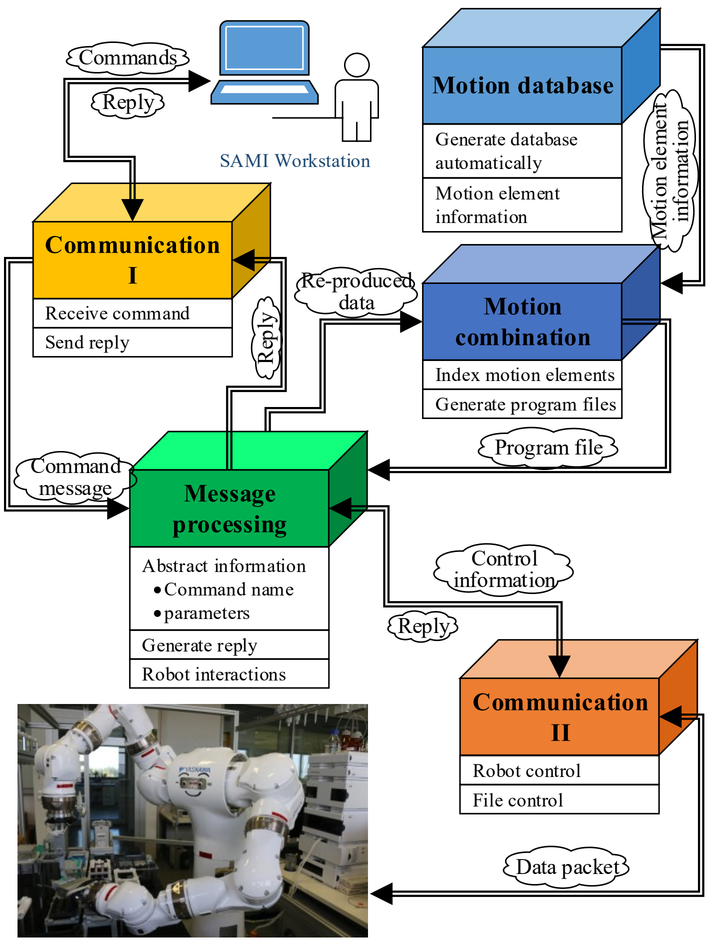


Figure 83. Architecture of interface program.

a) Motion database (M-database). As there are a lot of motion elements, a database is necessary for managing them. The motion element database provides the information of each motion

[3] More information about motion elements can be found in Chapter 6.

element (mainly including the job name, encoded information in the name, the value of position nodes and the relationship with other motion elements) to the motion combination module. The motion database is generated automatically, thus it can be refreshed easily and quickly when new motion elements are created or some motion elements are deleted.

b) **Motion combination.** One motion element is meaningless. For doing certain tasks, it is needed to integrate related motion elements. While the information about motion elements can be found from the M-database, this part is about how to integrate the motion elements according specific tasks. According to the information encoded in the command, related motion elements will be indexed from the motion database. The names and parameters of these motion elements will be used to generate a program file, which can be executed on the robot controller.

c) **Message processing.** It is mainly used to abstract the encoded information from commands, to generate reply messages according to execution results and to handle the data for interacting with the robot. The communications between the SAMI software and the R-interface are based sending and receiving XML format data. The information of tasks is encoded in the commands sent from SAMI. The interactions between the robot controller and R-interface are realized by sending and receiving data packet with specific formats.

d) **Communication I.** It is to communicate with SAMI software, including receiving commands from SAMI software and sending back reply messages to SAMI software. It is based on the internet protocol.

e) **Communication II.** It is responsible for the communication with the robot controller, and is based on the High-speed Ethernet protocol (Yaskawa). Data packets with special format are sent between the computer (the interface program) and the robot controller, for realizing robot control (e.g. moving the robot or reading robot status) and file control (e.g. downloading files to the robot controller).

9.1 Motion Element Database

9.1.1 Creating Motion Database

The motion element database (M-database) will be created dynamically by R-interface. The process of creating M-database by R-interface mainly included seven steps, as shown in Figure 84. First, R-interface will get the list of robot jobs from the robot controller. Thus, the M-database will reflect the newest information about robot jobs. When new jobs (jobs for realizing motion elements) are created, the M-database can be updated automatically. If some motion elements are deleted from the robot controller, the information about these motion elements will be removed when the M-database is renewed by R-interface. This is helpful to avoid missing out new built motion elements or including invalid motion elements. After the job list is obtained, invalid jobs will be removed from the list according its position nodes. The jobs, built for testing, showing or other aims, which are not for realizing motion elements, having no available position nodes will be removed from the list. As it is shown in Figure 41, the name of a motion element encodes the information of its usages. The R-interface will parse these names and maps them to the ones, which are closer to nature language (more examples can be found in Table 19). The translate names will be used when indexing and integrating motion elements. While some motion elements are for the movements of both arms, some motion elements are for single one (the left arm or the right one). In the next steps, the information of related arms, front node and end node of each motion element will be added to the half-finished database successively. Such information is used to find the next available motion elements. A motion element, of which the front node is coincided with the end node of the current motion element, is the next available motion element. The finding of next available motion elements will be detailed in section 9.1.3.

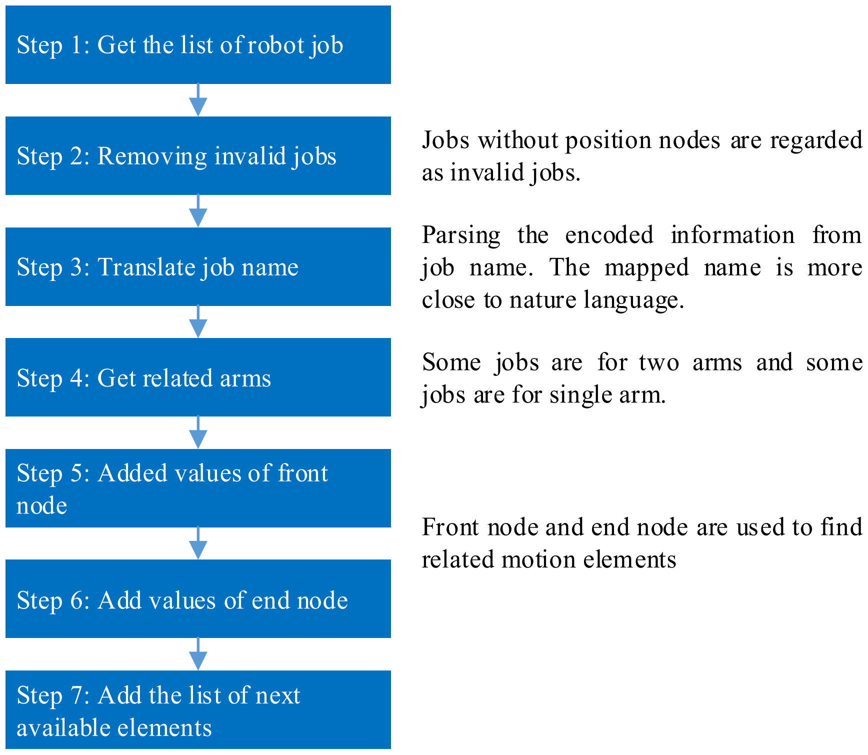

Step 1: Get the list of robot job
Step 2: Removing invalid jobs
Step 3: Translate job name
Step 4: Get related arms
Step 5: Added values of front node
Step 6: Add values of end node
Step 7: Add the list of next available elements
Jobs without position nodes are regarded as invalid jobs.
Parsing the encoded information from job name. The mapped name is more close to nature language.
Some jobs are for two arms and some jobs are for single arm.
Front node and end node are used to find related motion elements

Figure 84. Steps of creating M-database.

9.1.2 Properties of Motion Elements

In the M-database, each motion element has nine properties. Their descriptions can be seen in Table 4. The “JobName” gives the real name of a job and can be used to deal with the job, such as deleting, downloading or starting the job. The “TranslatedName” describes the usage of the job and is very close to human language. E.g., “HO_PICK_MTP_F_HOTEL” is the name of the job for picking a microplate for the hotel, and its translated name is “Pick microplate from hotel”. As it is infeasible to give a very long name to a job, the real name of a job is encoded and needs to be translated. The “ArmCount” is about the count the robot arms used in the job. Its value could be “1” or “2”. The “FirstArm” gives the master robot arm. If the value of “ArmCount” is “1”, the “SecondArm” will be null. The “FrontNode” is the front node of the motion element, while the “EndNode” gives its end node. When the value of “ArmCount” is “1”, each of them is one array recording values about one robot arm’ position. When the value of “ArmCount” is “2”, each of them is two arrays recording with values about the positions of both arms’. The “ArmCount”, “FirstArm”, “SecondArm”, “FrontNode” and “EndNode” are used in the process of generating the M-database. The “NextElementCount” gives the count of motion elements, which are available after this motion element is executed. And the “NextElementList” gives the list of names of the next available motion elements.

Table 4. Property of motion elements in M-database.

Name	Description
JobName	The identifier or real name of a job.
TranslatedName	Describes the usage of a job and it is close to nature language.
ArmCount	Gives the related count of robot arms in a job. It could be "1" or "2". Together with "FirstArm", "SecondArm", "FrontNode" and "EndNode", they are used to find the next motion elements in the process of generating M-database.
FirstArm	If the "ArmCount" equals to "1", "FirstArm" gives the related arm. If the "ArmCount" equals to "2", "FirstArm" gives the master arm. Its value could be "RB1" or "RB2".
SecondArm	If the "ArmCount" equals to "1", "SecondArm" is null. If the "ArmCount" equals to "2", "SecondArm" gives the slave arm. Its value could be "RB1" or "RB2".
FrontNode	Gives the values of front node. If "ArmCount" equals to "1", "FrontNode" includes the position of one arm. If the "ArmCount" equals to "2", "FrontNode" includes the position of both arms.
EndNode	Gives the values of end node. If "ArmCount" equals to "1", "EndNode" includes the position of one arm. If the "ArmCount" equals to "2", "EndNode" includes the positions of both arms.
NextElementCount	Gives the count of next available motion elements.
NextElementList	Gives the list of names of next available motion elements.

9.1.3 Finding Next Motion Elements

When one motion element is executed, there will be at least one motion element available to be executed next. Such motion elements are the next available motion elements (called next-elements). Next-elements are used, when integrating motion elements. The finding of next-elements is through position nodes. For finding the next-elements of a **c**urrent motion element (called C-element), the end node of C-element will be used to compare with the front nodes of the **o**ther motion elements (called O-element). If coincided, the O-element is a next-element of C-element. However, one motion element could be about the motions of left arm, right arm or both arms. Motion element for the left arm is marked as left-element. Similarly, it is marked as right-element for right arm, and marked as both-element when motions referring to two arms. Some rules have been used when comparing position nodes:

a) A left-element could be an available next-element of another left-element, but is not an available next-element of a right-element. A right-element could be an available next-element of a right-element, but is not an available next-element of a left-element too. So, the compare between left-element and right-element will not be conducted.
b) A left-element or right-element could be an available next-element of a both-element. The end node of a both-element consists of two parts: the position related with left arm, which will be used to compare with the front node of the left-element, and the position related with right arm, which will be used to compare with the front node of the right-element.
c) A both-element could be an available next-element of a left-element or right-element. The front node of a both-element also consists of two parts: the position related with left arm, which will be used to compare with the end node of a left-element, and the position related with right arm, which will be used to compare with the end node of a right-element.

There is no problem for a left-element or right-element being an available next-element of a both-element, as only one arm will be moved when a left-element or right-element is executed. However, there may be some problems, when a both-element is regarded as an available next-element of a left-element or right-element according the compare. In this case only one position of the arm is checked and the position of the other arm in the front node of the both-element is ignored. The problem is discussed with an example shown in Figure 85. In Figure 85, there is a left-element and a both-element. When the left-element is executed, the left arm will move from its front node to its end node. But the right arm will keep static (ignoring the movement of the base axis). When the both-element is executed, the left arm and the right arm will move from its front node to its end node at the same time. Assuming that the both-element is an available next-motion element of the left-element, there will be two cases when integrating them. In the case 1, the current position of right arm is coincided with the front node of the both-element and there is no problem, as there is no "gap". In case 2, when the current position of right arm is not coincided with the front node of the both-element, there will be an unknown path generated between the right arm and the front node of the both-element.

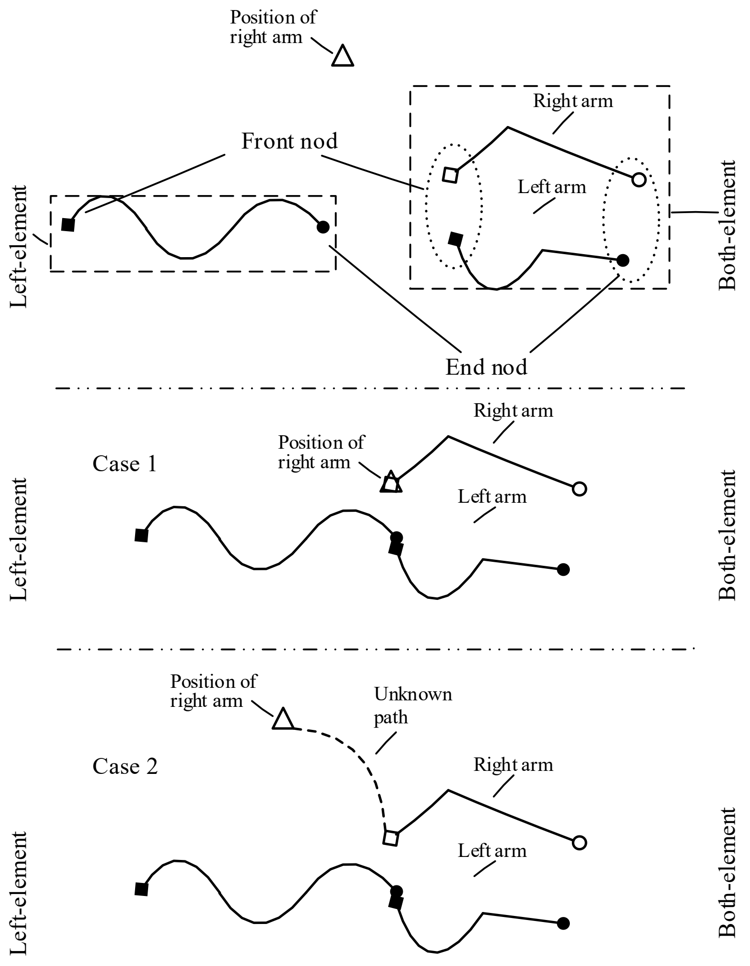


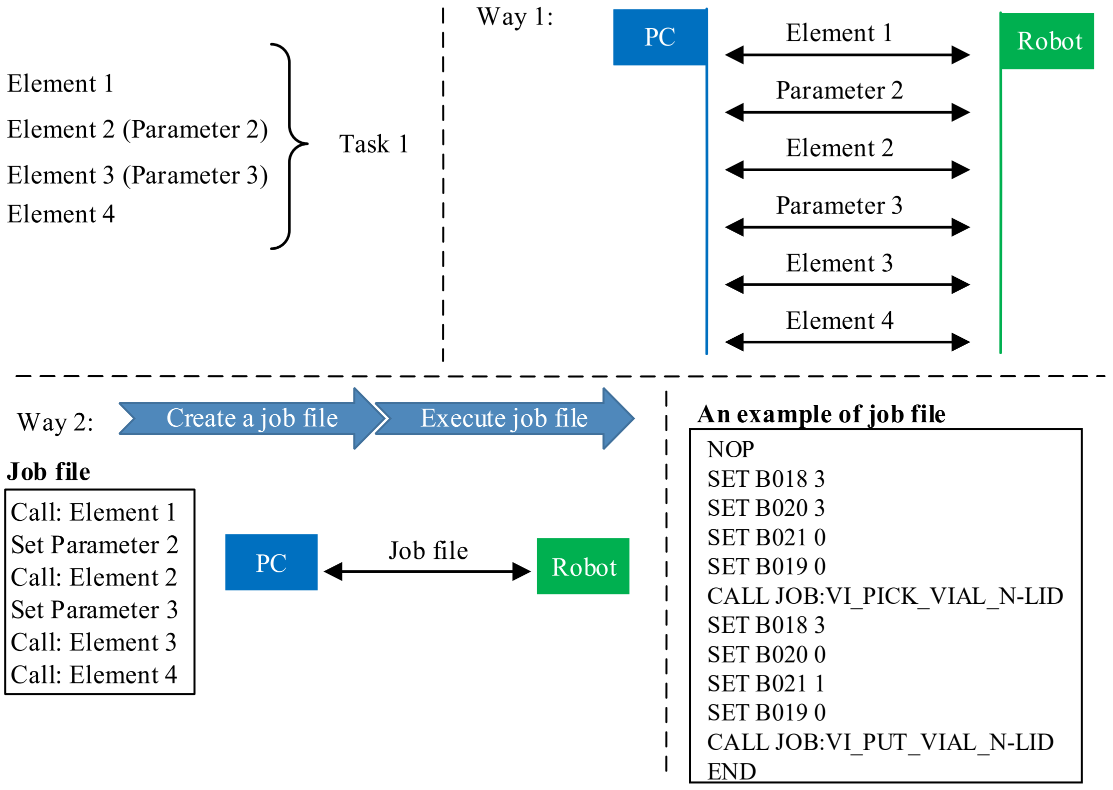
Figure 85. Potential problem of next available motion elements.

The potential problem shown in Figure 85 is solved in two aspects. Firstly, each motion element name is encoded with its usage. From the names; the logical relationships between them can be found. Motion elements with no logical relations will not be integrated together to avoid such potential problems. Thus, the case 2 situations can be avoided. Assuming the case 2 situation happened in some reasons, it would be prohibited to execute the next-element. The second solution is checking the matching between the current positions of robot arm or arms and the front node of the next-element (see section 6.1.3.3).

9.2 Motion Element Integration

9.2.1 Ways of Using Motion Elements

For moving the robot to execute tasks, such as transporting lab ware, opening vials or pipetting, normally four or much more motion elements will be required and some parameters (e.g. vial's position) are also needed to be set to certain motion elements. As motion elements are stored on the robot controller, so it is necessary to access the robot controller for executing them and for setting values to variables[4]. There are two ways of using motion elements, as shown in Figure 86.



```
NOP
SET B018 3
SET B020 3
SET B021 0
SET B019 0
CALL JOB:VI_PICK_VIAL_N-LID
SET B018 3
SET B020 0
SET B021 1
SET B019 0
CALL JOB:VI_PUT_VIAL_N-LID
END
```

Figure 86. Methods of using motion elements.

In the example, four motion elements are needed to execute task 1. In the way 1, each motion element is executing separately. The PC communicates with robot controller to execute element 1, communicates again to set parameter 2 and goes on until the element 4 is executed. The communications increase with the count of motion elements. When there are tens or hundreds of motion elements to be executed, the efficiency will be reduced due to the lots of commutations. And the robot motions are discontinuous, as the robot will stop moving until the next motion element is selected and then executed by the PC. In this case, the using of motion elements is changed to the way 2: creating a job file to include related motion elements and their parameters. Such a job file can be executed on robot controller and is named as call-file. When it is executed, the included motion elements in this file will be executed in succession. In this case, the PC only needs to generate such a call-file, download it to robot controller and execute it on robot controller. No matter how many motion elements are included in the call-file,

[4] INFORM language supports different types of variables. The B type variables are used as the interface for setting parameters to motion elements.

the communications will be not massive. And the robot will keep moving until the task is finished. The time costed for generating and downloading a call-fill is around tens of milliseconds, depending on the size of the call-fill (or the number of motion elements being integrated). An example of call-file in the application is given in Figure 86, which is to transfer a vial. The content of such a call-file can be completely different and much longer according different tasks. There are three ways of generating such a call-file, as following:

1) **Online:** Create call-files online by using a teach pendant: add the instructions for calling related motion elements and add variables for setting parameter to related motion elements.
2) **Offline:** Create call-files offline on personal computers. It is easier to create such call-file on computers. As by using the keyboard and mouse, the user can easily copy, paste, delete or modify the content of a call-file.
3) **Real-time:** R-interface can create such call-files in real-time according SAMI command. In the online and offline methods, the user needs to be very clear about the usages of each robot jobs, related variables and the means of their values. In the real-time method, the processes of generating such call-file will be performed automatically by R-interface. For some tasks, such a call-file may include more than 1000 instructions. In this case, it is not convenient to create such a call-file in offline method, as it will need tens of minutes to think and check about which motion elements and variables should be used, to add instructions as the content of the call-file. It will take more time in the online method. In the real-time method, this will be finished in milliseconds by R-interface.

Such a call-file will be generated by the R-interface automatically when executing SAMI commands. Figure 87 gives the process of generating a call-file. First, the task information will be parsed from the SAMI command.

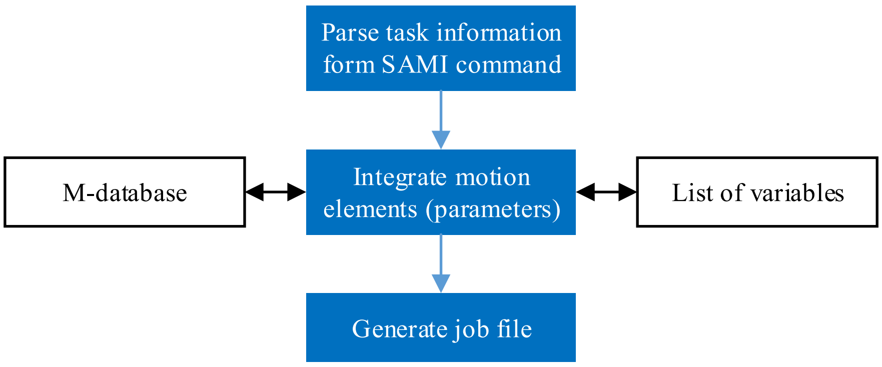
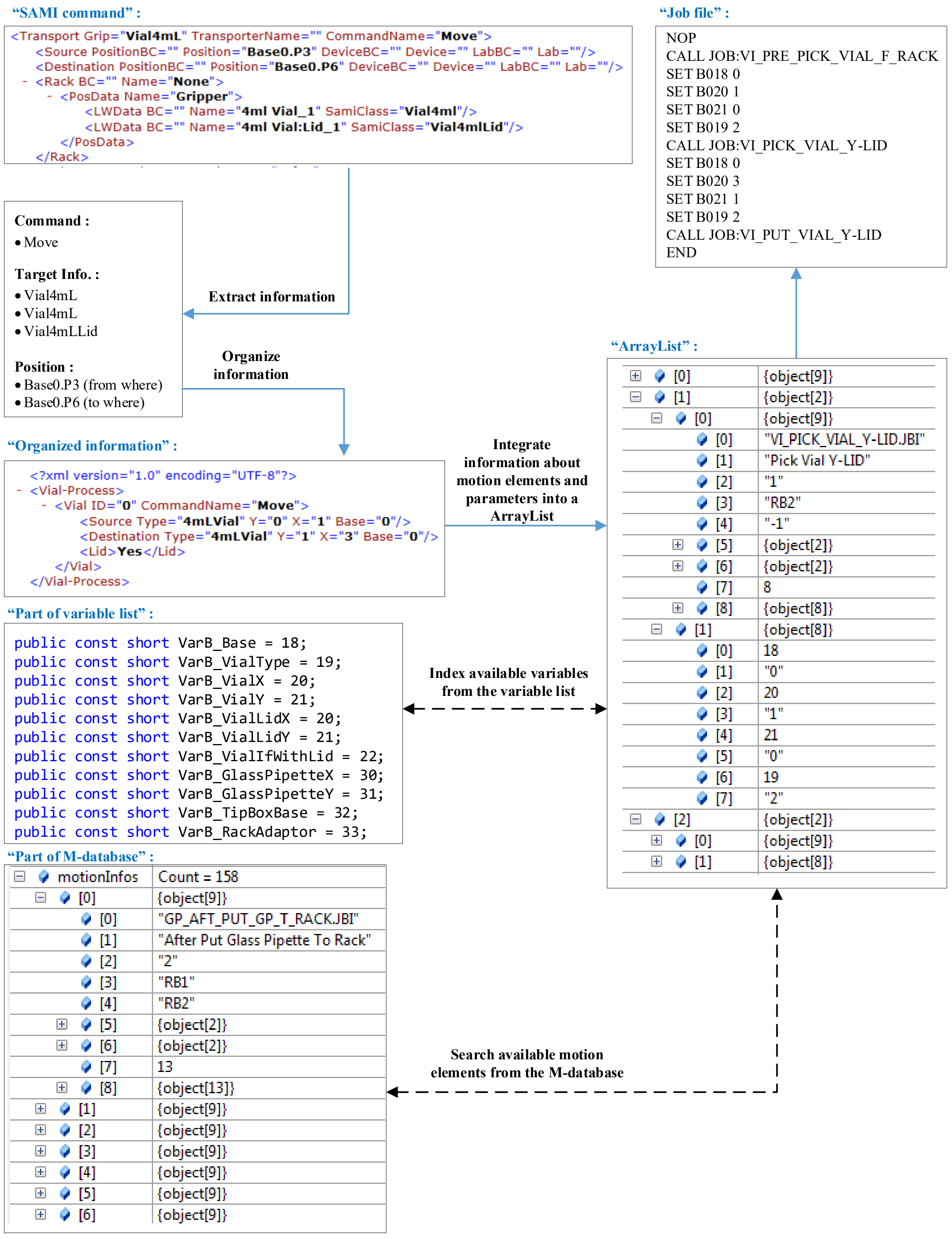

Parse task information form SAMI command
M-database
Integrate motion elements (parameters)
List of variables
Generate job file

Figure 87. Process of generating a call-file.

Second, the information will be used to find the related motion elements and set values to related variables. In this step, the names of motion elements, related variables and their values will be added to an "*ArrayList*" object. Then a call-file will be generated (an example of the content of such a call-file can be seen in Figure 86). In this step, the names of motion elements will be added with "CALL[5]" instructions and arranged in order and variables and their values will be added with "SET" instructions. The details of the process of generating a call-file are given in Figure 88.

[5] In INFORM language, "CALL" instruction is used to call a job and "SET" instruction is used to set value to a variable.



Extract information
Organize information
Integrate information about motion elements and parameters into a ArrayList
Index available variables from the variable list
Search available motion elements from the M-database

"SAMI command" :

```
<Transport Grip="Vial4mL" TransporterName="" CommandName="Move">
    <Source PositionBC="" Position="Base0.P3" DeviceBC="" Device="" LabBC="" Lab=""/>
    <Destination PositionBC="" Position="Base0.P6" DeviceBC="" Device="" LabBC="" Lab=""/>
  - <Rack BC="" Name="None">
      - <PosData Name="Gripper">
            <LWData BC="" Name="4ml Vial_1" SamiClass="Vial4ml"/>
            <LWData BC="" Name="4ml Vial:Lid_1" SamiClass="Vial4mlLid"/>
        </PosData>
    </Rack>
```

Command :
- Move

Target Info. :
- Vial4mL
- Vial4mL
- Vial4mLLid

Position :
- Base0.P3 (from where)
- Base0.P6 (to where)

"Organized information" :

```
  <?xml version="1.0" encoding="UTF-8"?>
- <Vial-Process>
   - <Vial ID="0" CommandName="Move">
        <Source Type="4mLVial" Y="0" X="1" Base="0"/>
        <Destination Type="4mLVial" Y="1" X="3" Base="0"/>
        <Lid>Yes</Lid>
     </Vial>
  </Vial-Process>
```

"Part of variable list" :

```
public const short VarB_Base = 18;
public const short VarB_VialType = 19;
public const short VarB_VialX = 20;
public const short VarB_VialY = 21;
public const short VarB_VialLidX = 20;
public const short VarB_VialLidY = 21;
public const short VarB_VialIfWithLid = 22;
public const short VarB_GlassPipetteX = 30;
public const short VarB_GlassPipetteY = 31;
public const short VarB_TipBoxBase = 32;
public const short VarB_RackAdaptor = 33;
```

"Part of M-database" :

⊟ motionInfos	Count = 158
⊟ [0]	{object[9]}
[0]	"GP_AFT_PUT_GP_T_RACK.JBI"
[1]	"After Put Glass Pipette To Rack"
[2]	"2"
[3]	"RB1"
[4]	"RB2"
⊞ [5]	{object[2]}
⊞ [6]	{object[2]}
[7]	13
⊞ [8]	{object[13]}
⊞ [1]	{object[9]}
⊞ [2]	{object[9]}
⊞ [3]	{object[9]}
⊞ [4]	{object[9]}
⊞ [5]	{object[9]}
⊞ [6]	{object[9]}

"Job file" :

```
NOP
CALL JOB:VI_PRE_PICK_VIAL_F_RACK
SET B018 0
SET B020 1
SET B021 0
SET B019 2
CALL JOB:VI_PICK_VIAL_Y-LID
SET B018 0
SET B020 3
SET B021 1
SET B019 2
CALL JOB:VI_PUT_VIAL_Y-LID
END
```

"ArrayList" :

⊞ [0]	{object[9]}
⊟ [1]	{object[2]}
⊟ [0]	{object[9]}
[0]	"VI_PICK_VIAL_Y-LID.JBI"
[1]	"Pick Vial Y-LID"
[2]	"1"
[3]	"RB2"
[4]	"-1"
⊞ [5]	{object[2]}
⊞ [6]	{object[2]}
[7]	8
⊞ [8]	{object[8]}
⊟ [1]	{object[8]}
[0]	18
[1]	"0"
[2]	20
[3]	"1"
[4]	21
[5]	"0"
[6]	19
[7]	"2"
⊟ [2]	{object[2]}
⊞ [0]	{object[9]}
⊞ [1]	{object[8]}

Figure 88. Details of the general process of generating a call-file.

In Figure 88, an example about moving a vial is used to explain the process of generating a call-file. But the processes of generating such a call-file for all the other tasks, including pipetting, transferring microplates, transferring tip boxes, transferring samples, opening/closing vials, dealing with thermo shaker and so on, are in the same way. First, the important information will be extracted from the SAMI command. In this example, the command name, information about the vial and position information will be extracted. Then, the information will be organized to another XML format data, in which the position information will be parsed: "Base0.P3" is replace with "X=1, Y=0, Base = 0" and "Base0.P6" is changed to "X=3, Y=1, Base = 0" (as the vials are arranged in rows and columns, thus it needs two parameters to identify a vial in the rack, see Figure 29). Next, the organized information will be used to search needed motion elements for moving this vial and to index the variables for identifying the positions of this vial. The information about these motion elements and variables will be added into an "*ArrayList*" object. In this example, the "*ArrayList*" has three "members", indicating that three motion elements are included. With the "member 1" expanded, the information about the motion element and it parameters can be seen. At last, the "*ArrayList*" is used to generate a call-file. The content of the call-file can be seen in Figure 88. The names of motion elements, the variables and their values come from the "*ArrayList*". The list of motion elements has been given in Table 19 and the list of variables can be found in Table 20.

9.2.2 Indexing Motion Elements

For a given task, certain motion elements will be needed to fulfil it. For example, three motion elements (*element 1*, *element e* and *element 9*) are needed to be integrated for a task, as shown in Figure 89. For indexing data of *element 1*, the program will use its mapped name to search the M-database and will stop searching until the target element is indexed or the whole M-database is searched.

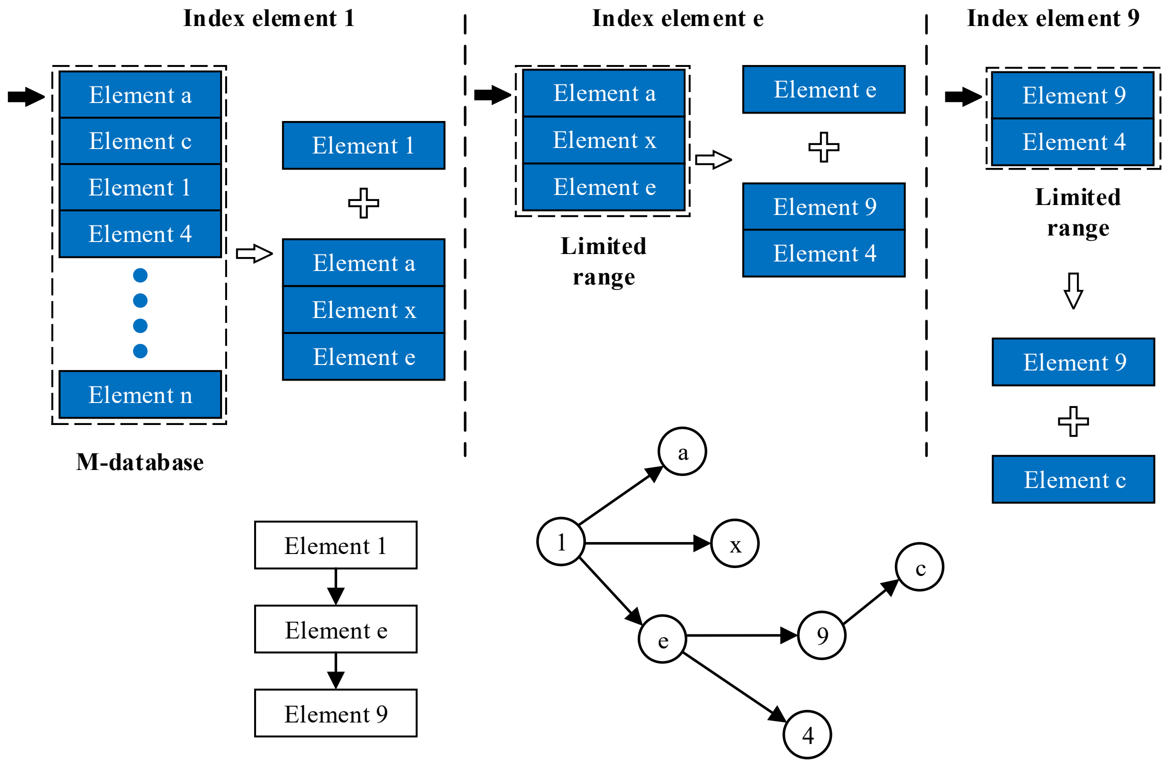


Figure 89. Indexing motion elements.

In this example, *element 1* has three next motion elements (*element a*, *element x* and *element e*). The names of these three motion elements are already recorded in the data of *element 1* when M-database is generated, thus the data of these three motion elements can also be indexed subsequently. So, the *element e* will be indexed from the limited range instead of the whole M-database. It is the same for indexing *element 9*. When a motion element is indexed, the data of its next motion elements will also be indexed and be used to form the limited range for indexing a next motion element. This is a safety precaution. In this example, the correct order of executing motion elements is *element 1*, *element e* and *element 9*. If a wrong order is given, e.g. *element 1*, *element 9* and *element e*, it will not be able to index *element 9* from the limited range. Thus, integrating wrong motion elements can be prevented.

9.2.3 Generating Call-files

The data of indexed motion elements will be added to a temporary array according to the search order. For some motion elements, they require variable settings. E.g., for picking a microplate from the hotel, variable values have to be set to identify a position in the hotel. The information of variables, including their names and values, will also be added to the array with the data of the motion elements. Once all motion elements are indexed, the data in the temporary array will be used to generate a call-file, which will be finally uploaded to the robot controller and executed. The first indexed motion element will be first added to the call-file. In such a job file, the names of motion elements will be called and variables will be set. The process of generating such a robot file is as follow:

```
Generating Job file:
1  Create an empty file
2  Fill basic content
3  For each motion element
4      if variables are required
5          for each variable
6              SET   variable name   value
7      CALL JOB: name of motion element
8  Finish file creating
9  Return True or False
```

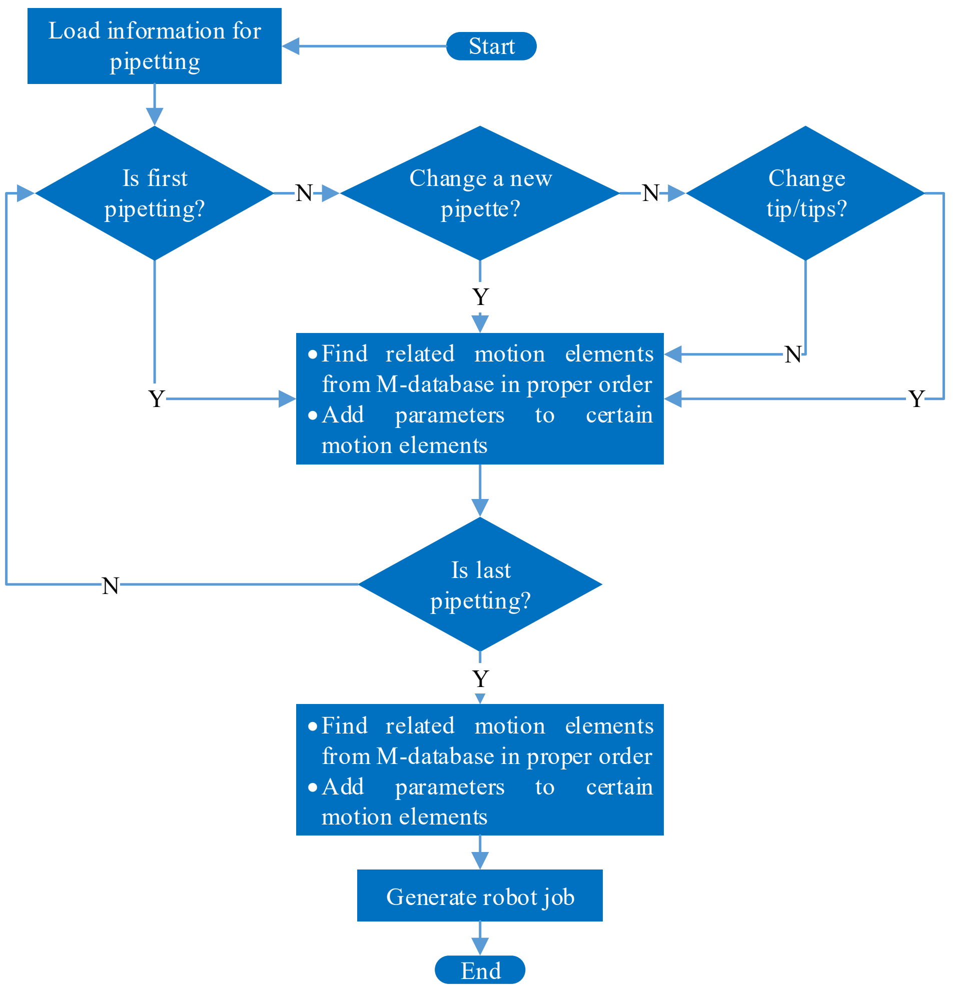
9.2.4 Integrating Motion Elements

In section 6.3, the motion elements of each module have been shown. This section will discuss the integrations of these motion elements for different tasks, such as opening vials pipetting solutions, transfer samples or do other tasks. These motion elements will be integrated by call-files, which will be generated by R-interface according SAMI commands. The generations of call-files for different modules are discussed in the following part of this section.

9.2.4.1 Pipette Module

Several motion elements for using pipettes to pipette solutions have be shown in section 6.3.1. R-interface will integrate these motion elements to conduct the pipetting tasks, according to SAMI commands. After the information about the pipetting task is parsed from the SAMI command, R-interface will use this information to generate an "*ArrayList*" object, which include the names of related motion elements, related variables and the values for these variables, and then use this "*ArrayList*" object to generate a call-file. When this call-file is uploaded to the robot controller and executed, the robot will move to pipette liquids. For one pipetting task, one call-file will be generated. Figure 90 shows the flow chart of generating call-files for using pipettes. In one pipetting task, there may be many times of pipetting: pipetting different solutions to different destinations. R-interface will deal with each pipetting separately:

1) If it is the first pipetting, R-interface will judge if it is needed to add pre-wetting parameters to the "*ArrayList*" according to the SAMI command. The pre-wetting to tips is useful when pipetting some solutions. Then R-interface will index the related motion elements from M-database and add them to the "*ArrayList*" in order. For some motion elements, it needs to set parameters to them. R-interface will index the variables from variable list and then add them and their values to the "*ArrayList*" just before the motion element is added.
2) According to the SAMI command, R-interface will also judge if it is needed to change pipettes or change tips before going on pipetting. If needed, the motion elements and parameters (variables and values) for changing pipettes or tips will be added to the "*ArrayList*", together with the motion elements and parameters for pipetting solutions. If not needed, only the motion elements and parameters for pipetting solutions will be added.
3) If it is the last pipetting, R-interface will add the motion elements and parameters (for putting the pipette back to the shelf) to the "*ArrayList*", then use this "*ArrayList*" to generate a call-file. If not, R-interface will go on to process the next pipetting.


Load information for pipetting
Start
Is first pipetting?
N
Change a new pipette?
N
Change tip/tips?
Y
N
Y
Y
• Find related motion elements from M-database in proper order
• Add parameters to certain motion elements
N
Is last pipetting?
Y
• Find related motion elements from M-database in proper order
• Add parameters to certain motion elements
Generate robot job
End
Figure 90. Flow chart of integrating motion elements for pipetting tasks.

9.2.4.2 Glass Pipette Module

The motion elements for using glass pipettes have been shown in section 6.3.2. This part will discuss the integration of these motion elements by R-interface. It is similar with the pipette module. First, R-interface will generate an "*ArrayList*" object using the information encoded in the SAMI command, which include the names of motion elements and parameters (variables and the values). Then this

"*ArrayList*" object will be used to generate a call-file. Figure 91 shows the flow chart of generating call-files for using glass pipettes:

1) If it is the first pipetting, R-interface will judge if it is needed to add pre-wetting parameters to the "*ArrayList*" according to the SAMI command. Then, the motion elements needed for this pipetting and their parameters will be added to the "*ArrayList*".
2) If it needs to change glass pipettes, the motion elements and parameters for changing glass pipettes will be added to the "*ArrayList*", together with the motion elements and parameters for pipetting solutions. If not, only the motion elements and parameters for pipetting solutions will be added.
3) If it is the last pipetting, R-interface will add the motion elements and parameters (for putting the glass pipette back to the rack) to the "*ArrayList*", then use this "*ArrayList*" to generate a call-file. If not, R-interface will go on to process the next pipetting.

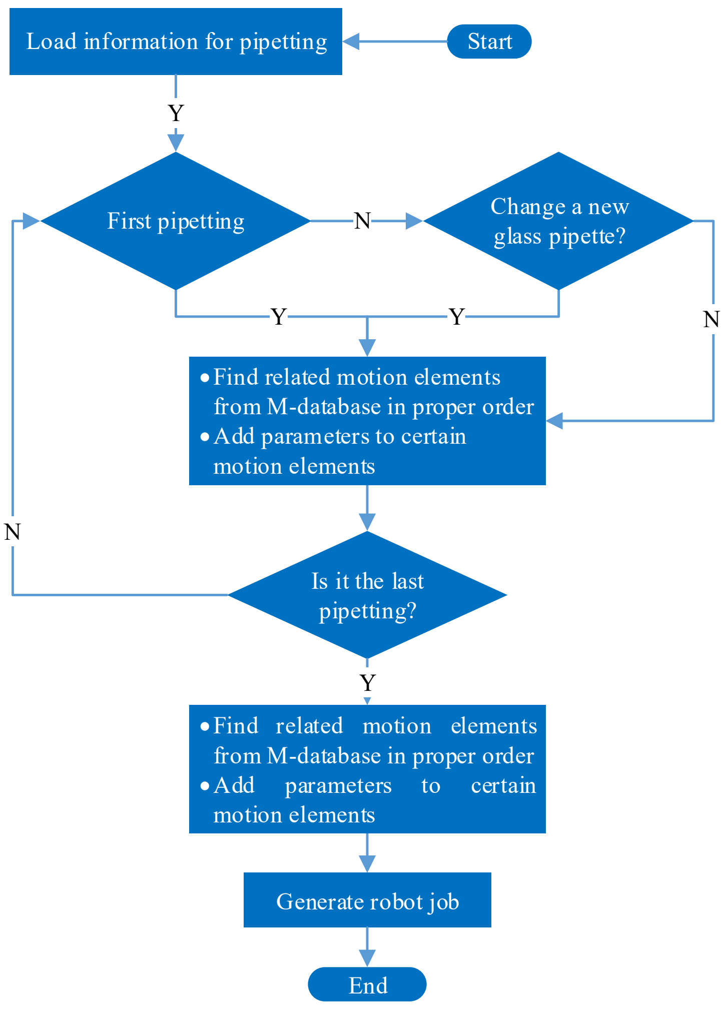

Start
Load information for pipetting
Y
First pipetting
N
Change a new glass pipette?
Y
Y
N
• Find related motion elements from M-database in proper order
• Add parameters to certain motion elements
N
Is it the last pipetting?
Y
• Find related motion elements from M-database in proper order
• Add parameters to certain motion elements
Generate robot job
End

Figure 91. Flow chart of integrating motion elements of glass pipette module.

9.2.4.3 Glass Vial Module

Motion elements of glass vial module are mainly used to do tree kind of tasks: opening vial, closing vial and moving vial (see section 6.3.3). R-interface will also integrate these motion elements in three ways. Also, an "*ArrayList*" object will be generated for creating a call-file. Figure 92 shows the flow chart of generating call-files for dealing with glass vials:

1) After the encoded information is parsed from the SAMI command, R-interface will judge if the task is to open a vial, close the vial or just move a vial. If the task is to open a vial, the motion elements needed for opening a vial are added to the "*ArrayList*". And the parameters indicating the position of the vial, the vial type and the position for putting the cap will be added to the "*ArrayList*" too. Then a call-file will be generated using this "*ArrayList*".
2) If the task is to close a vial, the motion elements needed for closing a vial are added to the "*ArrayList*". And the parameters indicating the position of the vial, the vial type and the position for the cap will be added to the "*ArrayList*". Then a call-file will be generated using this "*ArrayList*".
3) If the task is to move a vial, the motion elements needed for moving a vial be added to the "*ArrayList*". And the parameters indicating the position of the vial, the vial type and whether there is a cap with the vial will be added to the "*ArrayList*". Then a call-file will be generated using this "*ArrayList*".
4) If the task is unexpected, an exception will be thrown.

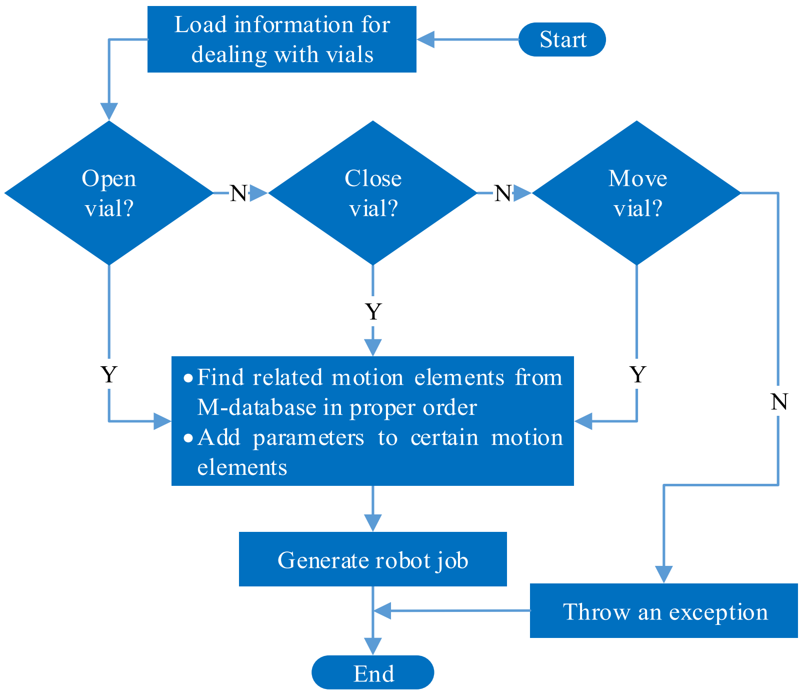

Load information for dealing with vials
Start
Open vial?
N
Close vial?
N
Move vial?
Y
Y
Y
N
• Find related motion elements from M-database in proper order
• Add parameters to certain motion elements
Generate robot job
Throw an exception
End

Figure 92. Flow chart for dealing with glass vials.

9.2.4.4 Hotel and Tip Box Modules

Microplates are stored in hotels. When used in pipetting processes, they will be transported to the table from the hotels. Tip boxes are placed on the shelf. Different pipettes may need different tips. When needed, tip boxes will be moved to the table too. From Figure 52 and Figure 53, it can be seen their motion elements has a similar relationship. The R-interface will also integrate them in a similar way. Figure 93 shows the flow chart of generating call-files for dealing with microplate. There are four cases of transferring microplates: from hotel to table, from table to hotel, from hotel to hotel and from table to table. For different cases, different motion elements will be used. After parsing the SAMI commands, R-interface will index the related motions from M-database and add them to an "*ArrayList*". The parameters indicating the position of the microplate on the table or in the hotel, will also be added to the "*ArrayList*". At last, a call-file will be generated.

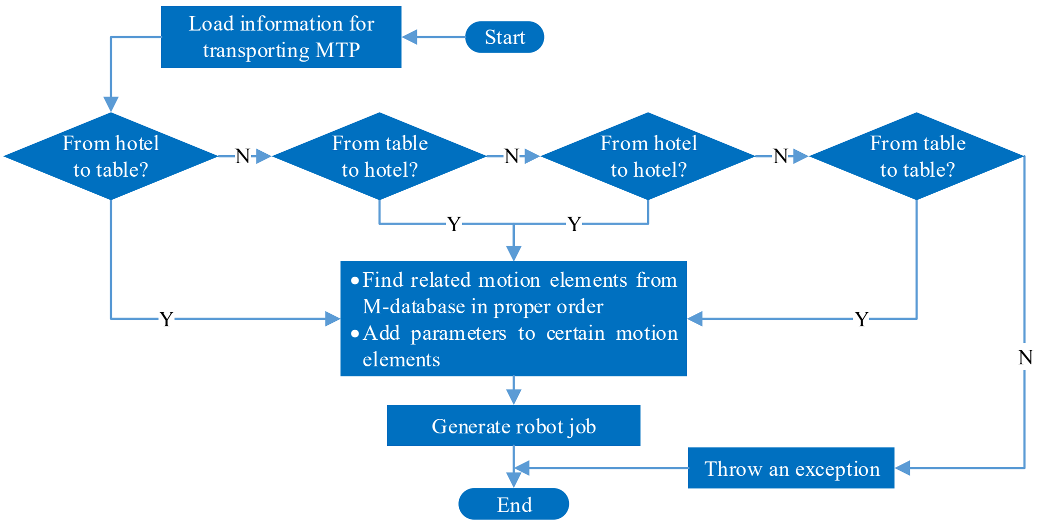

Start
Load information for transporting MTP
From hotel to table?
From table to hotel?
From hotel to hotel?
From table to table?
N
N
N
Y
Y
Y
Y
N
•Find related motion elements from M-database in proper order
•Add parameters to certain motion elements
Generate robot job
Throw an exception
End

Figure 93. Flow chart integrating motion elements to transfer microplate.

9.2.4.5 Syringe Module

This part is about using syringes to filter solutions, which are contained in glass vials. After the SAMI command is parsed, this information will be used to integrate motion elements to filter solutions. The process of filtering solution in one vial includes picking a syringe, loading a cannula to it, drawing solution into the syringe, releasing the cannula, loading a filter to the syringe, pressing the solution out and then putting the syringe to the waste bag. For filtering the solution in next vials, the process will be repeated again. The related motion elements and the parameters for locating the positions of syringes, cannulas, filters and vials will be added to an "*ArrayList*" object. When all vials are processed, the "*ArrayList*" object will be used to generate the call-file, as shown in Figure 94.

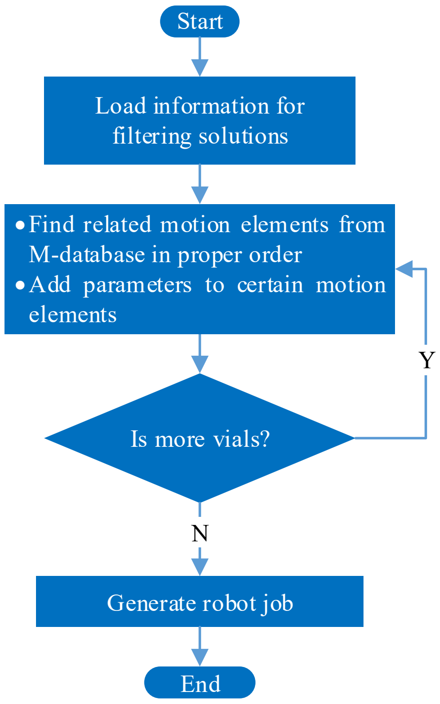

Start
Load information for filtering solutions
•Find related motion elements from M-database in proper order
•Add parameters to certain motion elements
Is more vials?
Y
N
Generate robot job
End

Figure 94. Flow chart for using syringes to filter solutions.

9.2.4.6 Ultrasonic Device Module

The ultrasonic device is used to promote extraction of cholesterol, together with a small shaker. First, vials containing solutions will be transferred the small shaker; after shaken, they will be vibrated under the ultrasonic; then shaken again on the small shaker. Figure 95 shows the flow chart of generating related call-files.

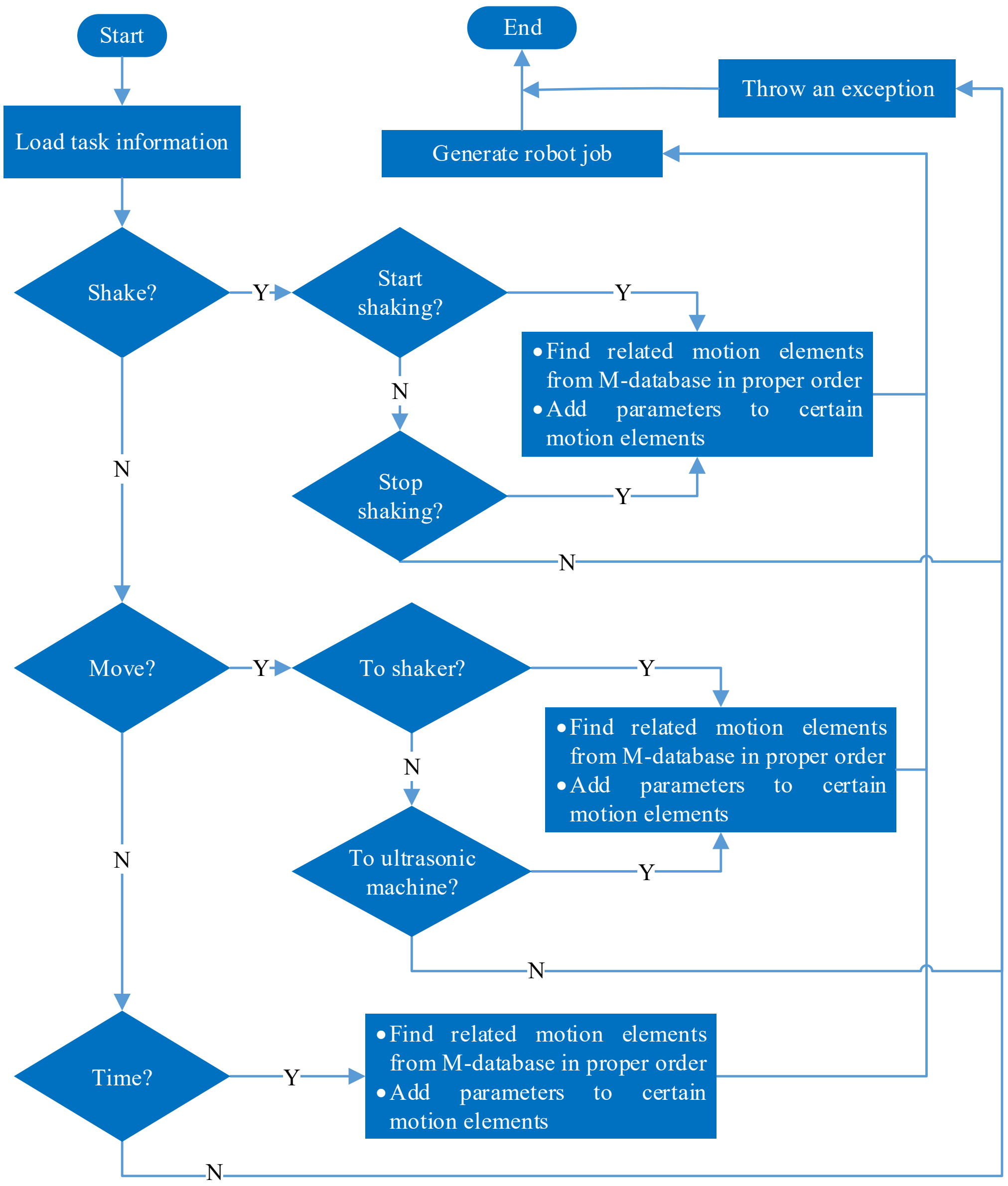

Start
End
Throw an exception
Load task information
Generate robot job
Shake?
Y
Start shaking?
Y
•Find related motion elements from M-database in proper order
•Add parameters to certain motion elements
N
N
Stop shaking?
Y
N
Move?
Y
To shaker?
Y
•Find related motion elements from M-database in proper order
•Add parameters to certain motion elements
N
N
To ultrasonic machine?
Y
N
Time?
Y
•Find related motion elements from M-database in proper order
•Add parameters to certain motion elements
N

Figure 95. Flow chart of integrating motion elements related with ultrasonic machine.

After the information from SAMI command is obtained, R-interface will generate different call-files according the commands. The “Move” command is used to transfer vials between the small shaker and the ultrasonic device. With different parameters, different call-files will be generated for transferring vials from the small shaker to the ultrasonic machine or from the ultrasonic machine to the small shaker. When the “Shake” command is executed, a call-file will be generated by including the names of motion elements for pressing a button to start the shaking or stop the shaking depending on the parameters. The “Time” command is used to set the time of ultrasonic treatment.

9.2.4.7 Thermo Shaker Module

The thermo shaker is used to shaking solutions in a sample preparation process (related motion elements can be found in section 6.3.8). In this part, R-interface will do these tasks: to open or close the thermo shaker (open: remove the lid from thermo shaker, close: cover the lid to thermo shaker), to move microplate (from table to thermo shaker, from thermo shaker to table, from table to the tray of LC auto sampler or from the tray to table) and to lid or delid microplate (lid: cover the lid to microplate, delid: remove the lid from microplate). This class is mainly using for shaking microplate with samples and transporting microplate to the tray of LC auto sampler. Figure 96 shows the flow chart of generating call-files related with thermo shaker:

1) According to the SAMI command, the R-interface will judge the command is “Open”, “Close”, Move”, “Delid” or “Lid”. If the command is “Open”, R-interface will add the motion elements for removing the lid from thermo shaker to the “*ArrayList*”. Only after the lid is removed, it is able to move a microplate into the thermo shaker. If the command is “Close”, R-interface will add the motion elements for covering the thermo shaker to the “*ArrayList*”. After transporting the microplate into the thermo shaker and covering the thermo shaker with the lid, the shaking can be started. SAMI will send commands to thermo shaker to start the shaking motions. In order to prevent the evaporation of the solution when shaking, a lid will be covered on the microplate. The related command is “Lid”. If the command is “Delid”, R-interface will add the motion elements for removing the lid from the microplate to the “*ArrayList*”.
2) Several SAMI commands will be executed for doing the shaking task. In the phase of transporting microplate to thermo shaker, four SAMI commands will be executed: 1) “Open”, remove the lid from thermo shaker; 2) “Move”, move microplate to thermo shaker; 3) “Lid”, cover the lid on microplate; 4) “Close”, cover the lid to thermo shaker. Then it is ready for SAMI to send the shaking command to thermo shaker. When shaking is finished, the microplate will be transported out from thermo shaker. In the phase of transferring microplate from thermo shaker to table, four SAMI commands will be executed too: 1) “Open”, remove the lid from thermo shaker; 2) “Delid”, remove the lid from microplate; 3) “Move”, move microplate to table; 4) “Close”, cover the lid to thermo shaker.
3) The parameters for indicating microplate’s position and the microplate lid’s position will be added to the “*ArrayList*”.
4) When the samples’ preparations are finished, they are ready to be transported to the LC auto sampler. For doing this, the microplate containing samples will be transported to the tray of LC auto sampler first. The “Move” can be used to transport microplate from the table to the tray or from the tray to the table. R-interface will judge the “move” directions according to the information encoded in the SAMI command. When the microplate is one the tray, the “Lid” and “Delid” commands are still available.
5) For each command, “Open”, “Close”, Move”, “Delid” or “Lid”, a call-file will be generated.

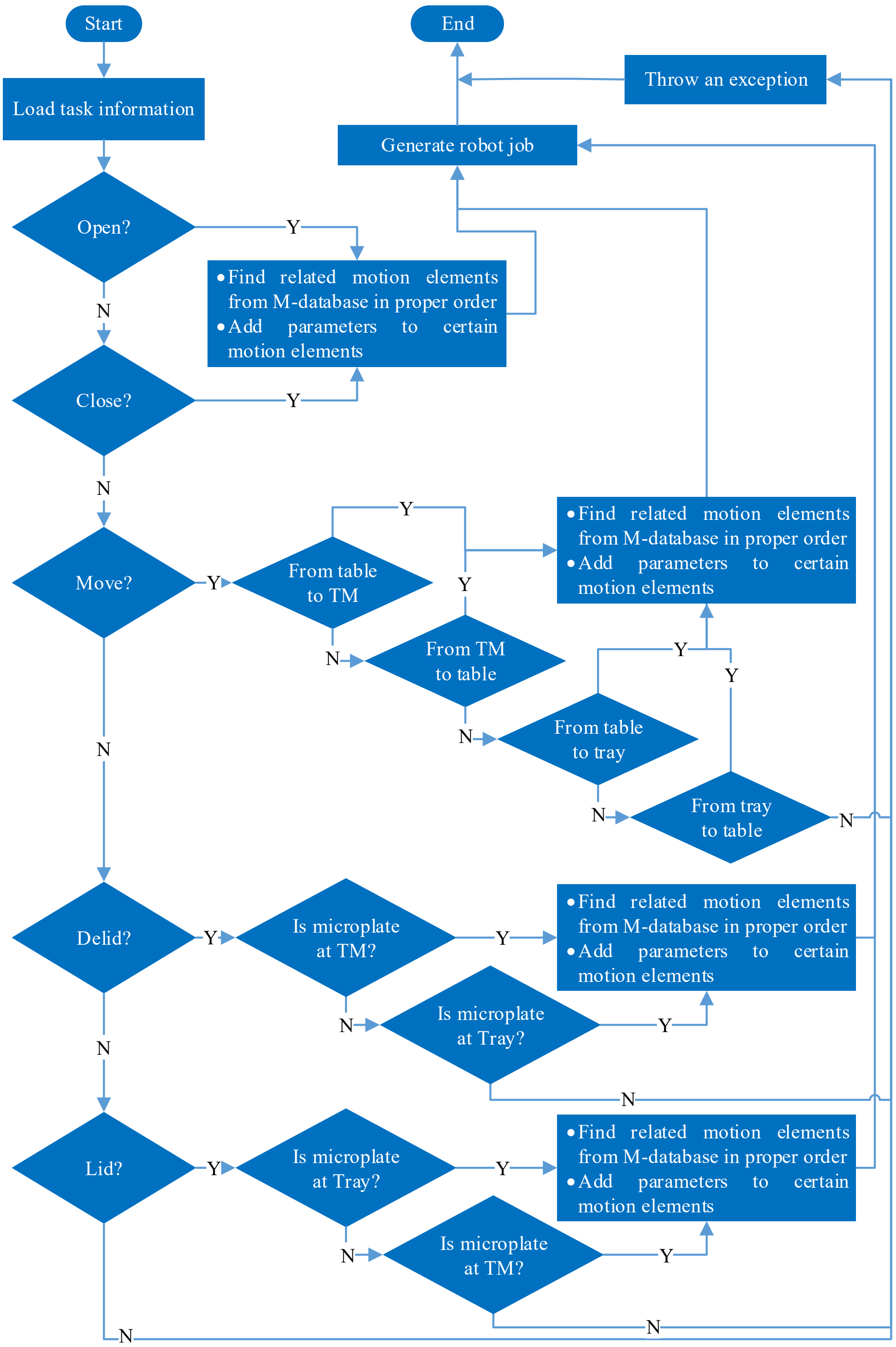

Start
Load task information
Open?
Y
N
Find related motion elements from M-database in proper order
Add parameters to certain motion elements
Close?
Y
N
End
Throw an exception
Generate robot job
Move?
Y
From table to TM
Y
N
From TM to table
Y
N
From table to tray
Y
N
From tray to table
Y
N
Find related motion elements from M-database in proper order
Add parameters to certain motion elements
N
Delid?
Y
Is microplate at TM?
Y
N
Is microplate at Tray?
Y
N
Find related motion elements from M-database in proper order
Add parameters to certain motion elements
N
Lid?
Y
Is microplate at Tray?
Y
N
Is microplate at TM?
Y
N
Find related motion elements from M-database in proper order
Add parameters to certain motion elements
N

Figure 96. Flow chart of integrating motion elements related with thermo shaker.

9.2.4.8 LC Auto Sampler and GC Auto Sampler Module

After samples are prepared in the microplate and then the microplate is transported to the tray LC auto sampler, it is ready to transfer the samples (on the tray) into the LC auto sampler. The motion elements of this module can be found in section 6.3.9. Figure 97 shows the flow chart of generating call-files for transferring samples into the LC auto sampler. There are two directions: transferring samples from the table into the auto sampler or transferring samples from the auto sampler to the table. When the samples for analyzing by GC-MS are prepared, they are contained in vials and will be transported to the auto sampler of GC-MS one by one.

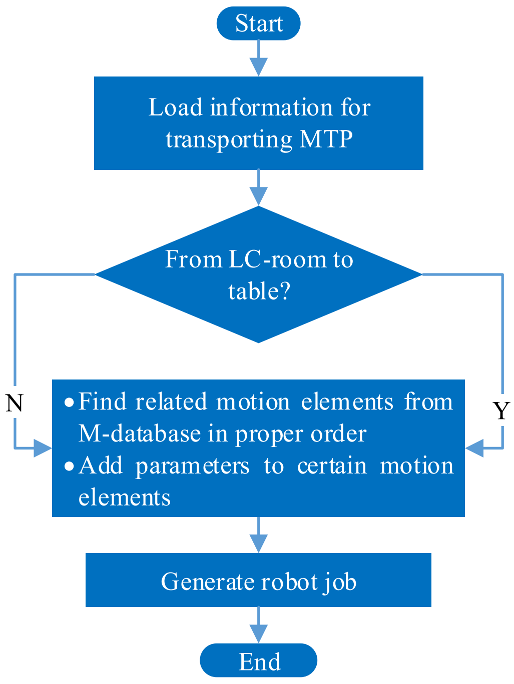

Start
Load information for transporting MTP
From LC-room to table?
N
Y
• Find related motion elements from M-database in proper order
• Add parameters to certain motion elements
Generate robot job
End

Figure 97. Flow chart of integrating motion elements to transport samples to LC auto sampler.

9.3 Communications with the Robot Controller

In order to control the robot, R-interface will communicate with the robot controller (FS100). Two ways has been used for performing the communications. One way is based on the Motocom32 APIs, with a dongle needed. The other way is to communicate with the robot controller through the High-speed Ethernet protocol (called HSE for short) and without a dongle. The communication module based on HSE protocol will be discussed in this section.

9.3.1 Packet Format

According to HSE protocol, a series of data packets will be processed by the program, including sending data packets to FS100 and parsing data packets sent from FS100. A data packet includes two parts: header part and data part. A header part includes 32bytes, and is mainly used to clarify:

- Identifier, which is fixed to “YERC”.
- Data size, including the size of the header part and data part.

- Processing division, which is used to indicate this packet is for robot control (when the value is set to 0x01) or file control (when the value is set to 0x02). While robot control is to do some control to the robot such as reading robot status, setting servo ON/OFF or reading robot positions, the file control is to handle robot job files, such downloading jobs, deleting jobs or listing jobs.
- ACK: if this is a request packet[6] sent to robot, this byte should be set to 0x00. Otherwise, it should be set to 0x01.
- ID: indicating the ID of the packet.
- Block No. (4 bytes). When it is a request packet, theses bytes will be set to 0. In data transmission, the Block No. will be more meaningful, e.g. for transferring a big job file. As the job file has a big size, it needs to be separated to several packets. After each packet is sent, the Block No. will be added 1. And it will add 0x80000000 in the last packet.
- Sub-header (request), indicating the command and parameters.
- Sub-header (answer), indicating the communication result.

The data part is only used when some data needs to been transferred, such the status data, job name, or job content. And the size of a data part is not fixed, but is no more than 479 bytes. By handling the data packets, several functions had been created, see Table 5.

Table 5. Functions of communication module based on HSE protocol.

No.	Functions
1	Read robot status
2	Read robot position
3	Get the list of jobs on FS100
4	Download job to FS100
5	Upload job to PC
6	Delete job from FS100
7	Set servo ON/OFF, set hold ON/OFF
8	Select a job on FS100
9	Start a job on FS100

9.3.2 General Process of Handling Data Packets

The HES is based on UDP (User Datagram Protocol). In the robot control, the UDP port in the FS100 side is fixed to 10040. And in the file control, the UDP port in the FS100 side is fixed to 10041. Figure 98 shows the general process of handling data packet. First, the UDP objects will be initialized. Then, the data packet according the packet format is generated and sent to FS100. The data packet will be different according the specific functions (e.g. reading robot status or reading robot positions). Next, the program will wait for the reply from FS100 in certain time. In this way, when the connection between the PC and FS100 is not correct, the program can be interrupted after waiting not a long time. If time is out and there is on reply, an exception will be thrown indicating that the connection between PC and FS100 is not good. If a reply is detected in certain time, the program will check the count of available

[6] A “request” packet is sent from PC to FS100 to command the FS100 to do something.

bytes replied from FS100. If the count is more than 0, the program will receive the data packet. If the count of available bytes equates to 0, an exception will be thrown. In the case of the count equating to 0, receiving data packet can block the program. After the data packet is received, the program will check the sub-header part of the packet to determine if the request is answered successfully. If not, the "answer" data packet will be ignored and an exception will be thrown. If yes and count of bytes of data part is not zero, the program will abstract the data from the data part of the packet. For some requests, the answer packet does not include the data part, such as "Select a job on FS100".

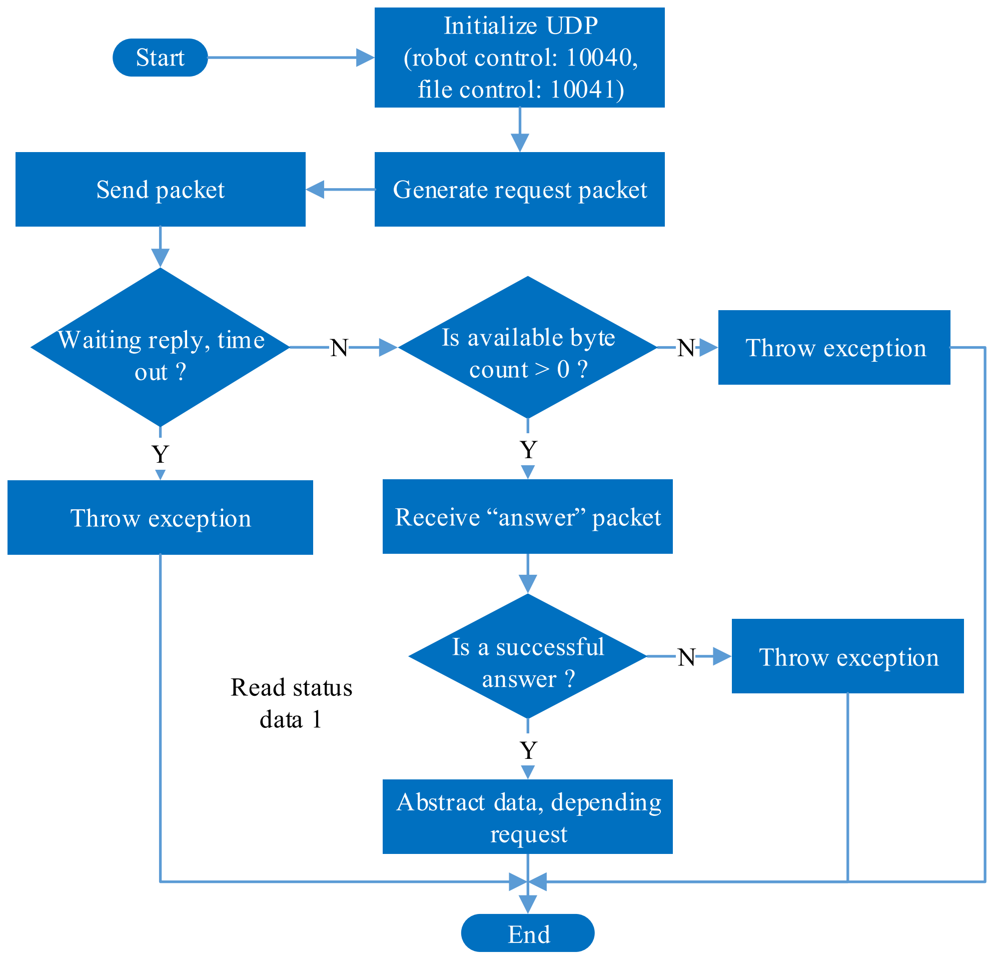


Figure 98. General flow chart of handling data packet.

9.3.3 Reading Robot Status

The flow chart of reading robot status has been given in Figure 99. The status data of robot includes there three parts: data 1, data 2 and IO data (80020). The data is to indicate the status of the robot, such as if the robot is running, if the robot is stopped by the light curtain or if the safe guard is open. More information about the description of status data can be seen in Table 23. The status data 1 and data 2 are obtained by using command "0x72", and the IO data with the logical number 80020 is read by using command "0x78". But the program flow chart of reading these status data is similar. First, the UDP objects will be initialized. Then, the request data packet for reading robot status is generated and sent to FS100. Next, the program will wait for the reply from FS100 in certain time. If time is out and there is on reply, an exception will be thrown indicating that the connection between PC and FS100 is not good. Otherwise, when the count of available bytes replied from FS100 is more than 0, the program will receive the data packet. If the count of available bytes equates 0, an exception will be thrown. After the data packet is received, the program will check the sub-header part of the packet to determine if the data packet is useful. If yes, the program will abstract the status data from the data part of the packet.

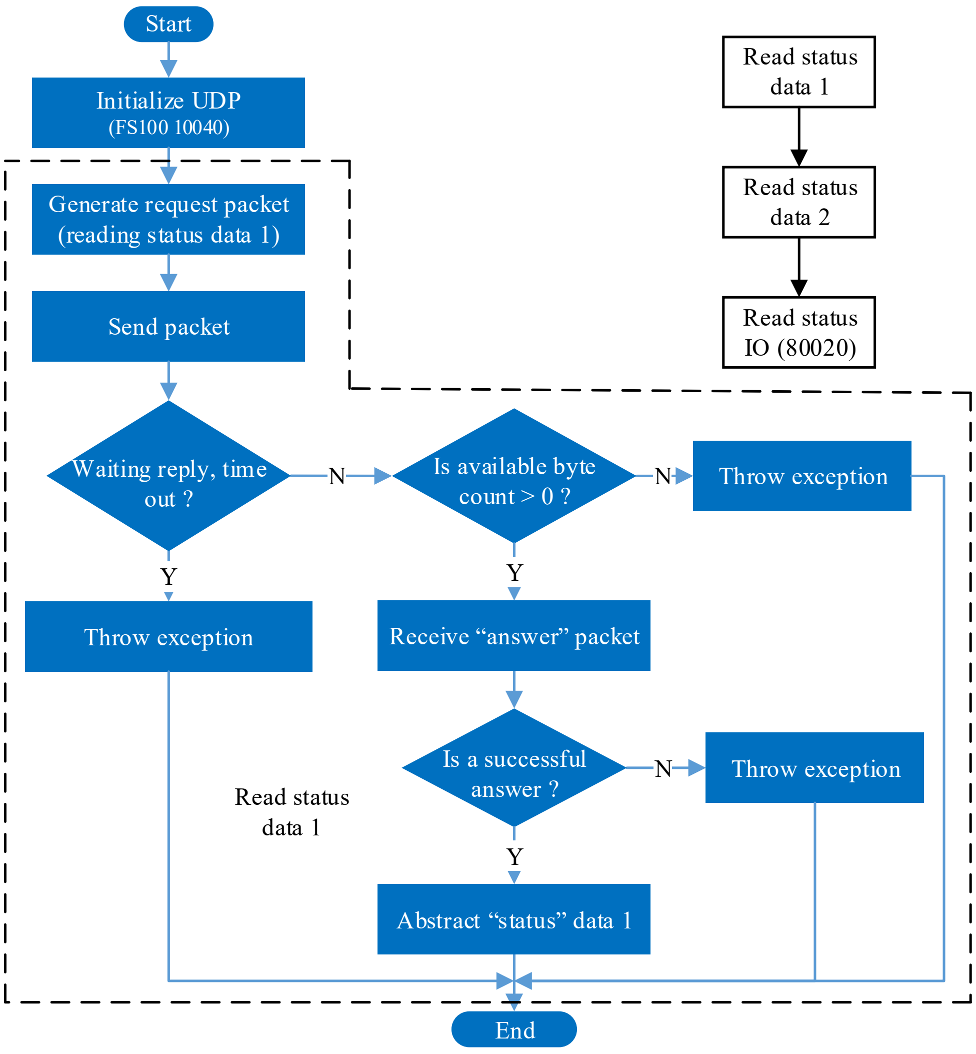


Figure 99. Flow chart of reading robot status.

9.3.4 Other Functions of Robot Control

The functions of reading robot positions, setting servo ON/OFF, setting hold ON/OFF, selecting a job on FS100 and starting a job on FS100 have the similar data packet handling process as shown in Figure 98. The request data packets are different. When reading robot positions, No. "0x75" command is used. The robot position data is included in the data part of the "answer" packet. When setting servo ON/OFF or setting hold ON/OFF, No. "0x83" command is used and the data part of the request packet indicates the specific operation ("ON" or "OFF"). The data part of "answer" packet is not used. Before running a certain job on the FS100, this job should be selected first. When selecting a job on FS100, No. "0x87" command is used with the name of the job included in the data part of the request packet. For starting (running) a job on FS100, No. "0x86" command is used with the data part of the request packet having a fixed value "0x00000001".

9.3.5 Downloading Jobs

According to the packet format, the data part of the packet can only include 479 bytes in maximum. Thus, when a job file is bigger than 479 bytes, it needs to be separate to several packets for sending. Figure 100 shows the flow chart of downloading a job from PC to FS100.

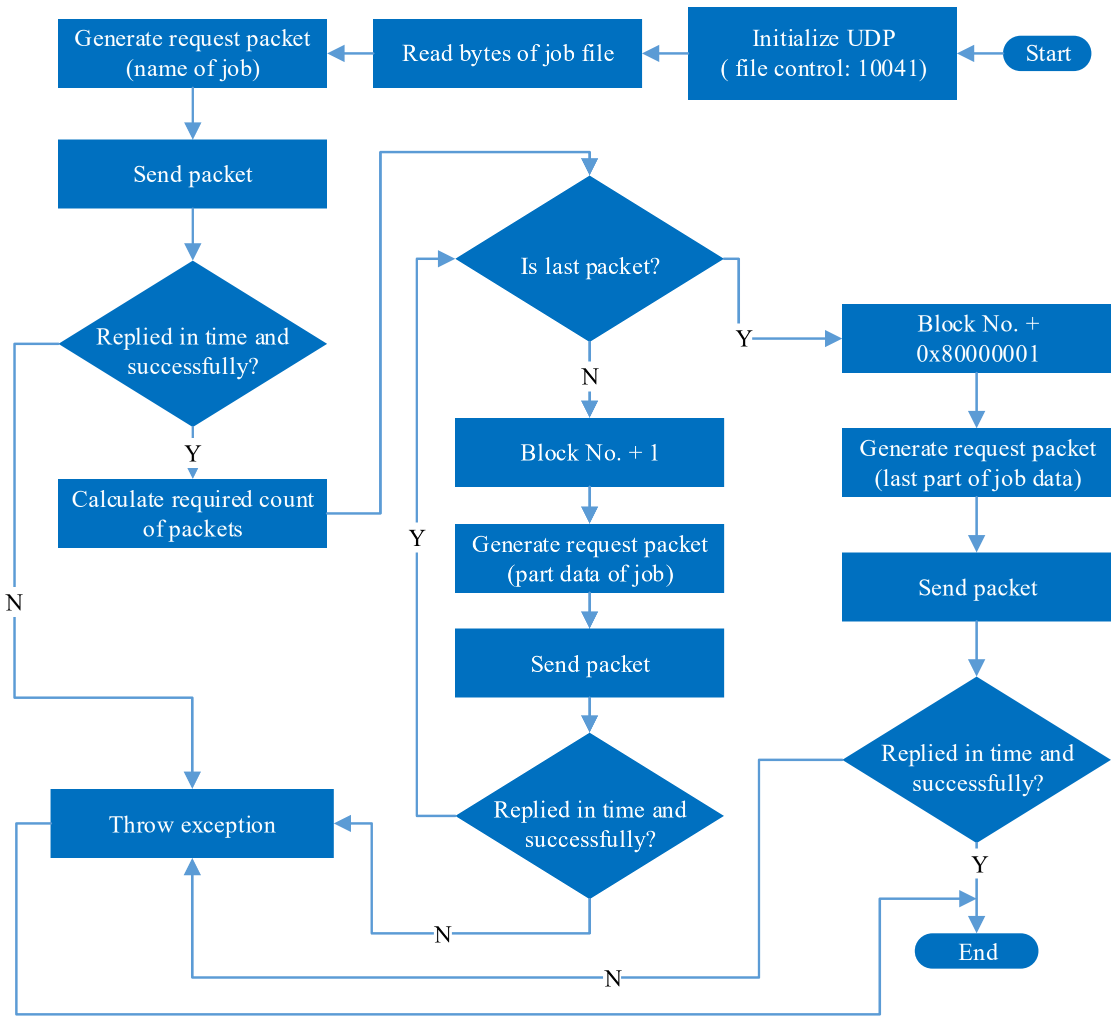

Start
Initialize UDP (file control: 10041)
Read bytes of job file
Generate request packet (name of job)
Send packet
Replied in time and successfully?
Y
N
Calculate required count of packets
Is last packet?
N
Y
Block No. + 1
Generate request packet (part data of job)
Send packet
Replied in time and successfully?
Y
N
Throw exception
Block No. + 0x80000001
Generate request packet (last part of job data)
Send packet
Replied in time and successfully?
Y
N
End

Figure 100. Flow chart of downloading a job from PC to FS100.

After the UDP objects are initialized, the program will read the data of the job in bytes. Then a request packet is generated with the name of job included in its data part, which is to inform FS100 that a job is going to be sent to it. After this request packet (including name of job) is sent to FS100, the program will check if it is received successfully (more information about the process of handling data packet can be seen in Figure 98). If it is received successfully, the program will calculate the required count of packets needed for sending the data of the job, using the number of bytes as the dividend and using “479” as the divisor. If the size of the job is no more than 479 bytes, only one packet is needed (this packet is also the last packet). Otherwise, at least two packets are needed. Figure 101 shows an example of separating job data. In this example, the job will be separated to four parts, with the previous three parts including 479 bytes separately and the last part including 363 bytes.

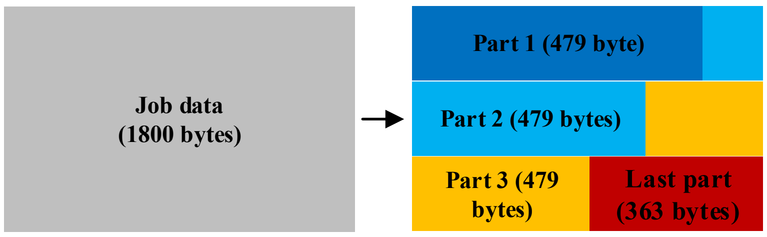


Figure 101. Example of separating job data.

The data of part 1 will be sent first, by included to the data part of a request packet. And the other parts will be sent successively. The Block No. will be added by one for sending each packet, indicating the number of packet being sent. For the last packet, the highest byte of Block No. will be set to 0x80 to indicate that this is the last packet of the job data.

9.3.6 Other Functions of File Control

Besides downloading job to FS100, a job can also be uploaded from FS100 to PC or be deleted from FS100. Also, the list of jobs on FS100 can be obtained by communicating with FS100 based on HSE protocol. The process of deleting a job from FS100 is simple, as it just needs to specify the name of the job to be deleted in the data packet. The processes of upload job and getting list of jobs are similar. Both of them are to request some kinds of data from FS100. While uploading job is to request the data of a certain job, getting list is to request the data about the names of all the jobs saved on FS100. Figure 102 gives the flow chart of uploading a job from FS100 to PC.

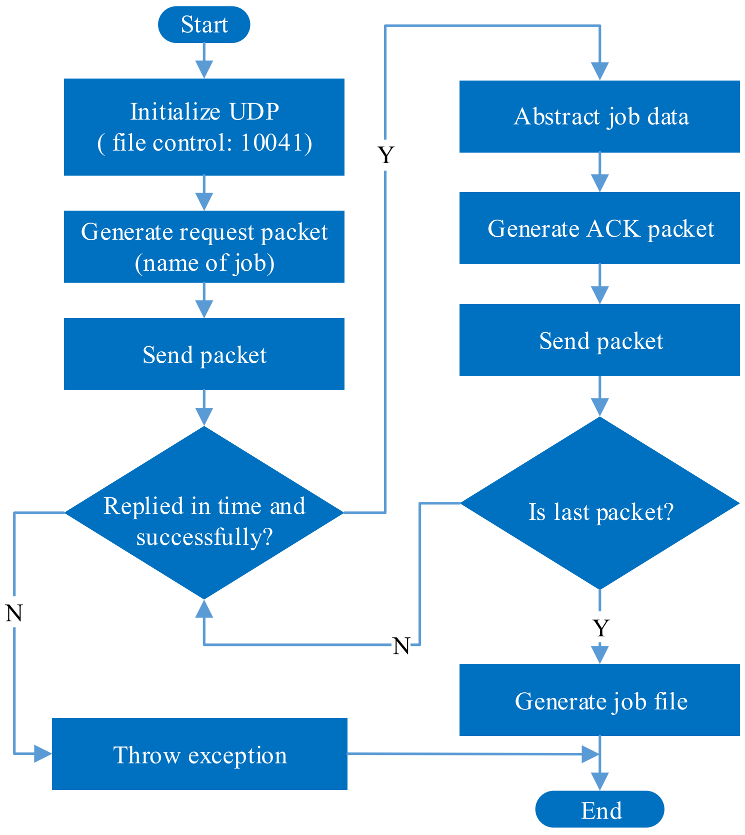


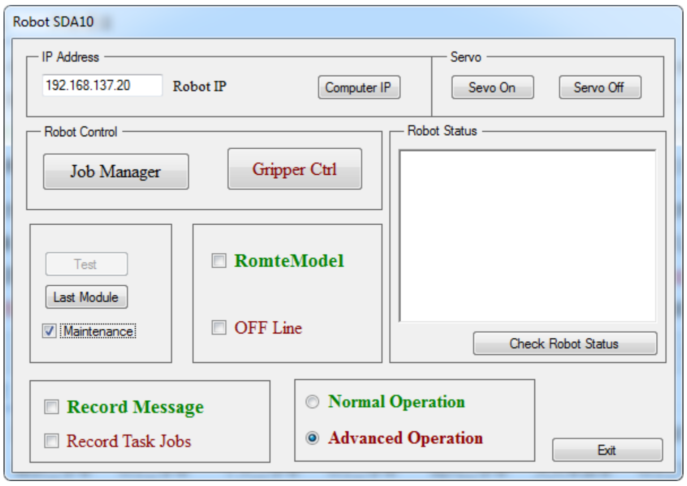
Figure 102. Flow chart of uploading a job from FS100 to PC.

In the request packet, the name of the job required to be transmitted is included. If the request packet is received successfully and the job is on FS100, FS100 will start to transmit the job data (no more than

479 bytes per transmission). Thus, for a job having much more than 479 bytes, it can be transmitted in several packets. If the reply from FS100 is correct, the program will abstract job data from the packet. Then, an ACK packet, which is used to inform FS100 that the packet if received successfully, will be generated and sent to FS100. FS100 will continue to transmit the remaining packets, until the last packet is transmitted. The program will determine if it is the last packet by checking the Block No. At last, all the received job data will be combined to generate a job file and saved on the PC.

9.4 Auxiliary Tools

Even through R-interface in used for integrating the robot system with SAMI system, a user interface has also been built for helping the user to making some testing or managing robot jobs. By using it, the user can set IP address of robot, open some child windows or do other things.


Robot SDA10

IP Address: 192.168.137.20 Robot IP

[✓] Maintenance

[ ] RomteModel

[ ] OFF Line

[ ] Record Message

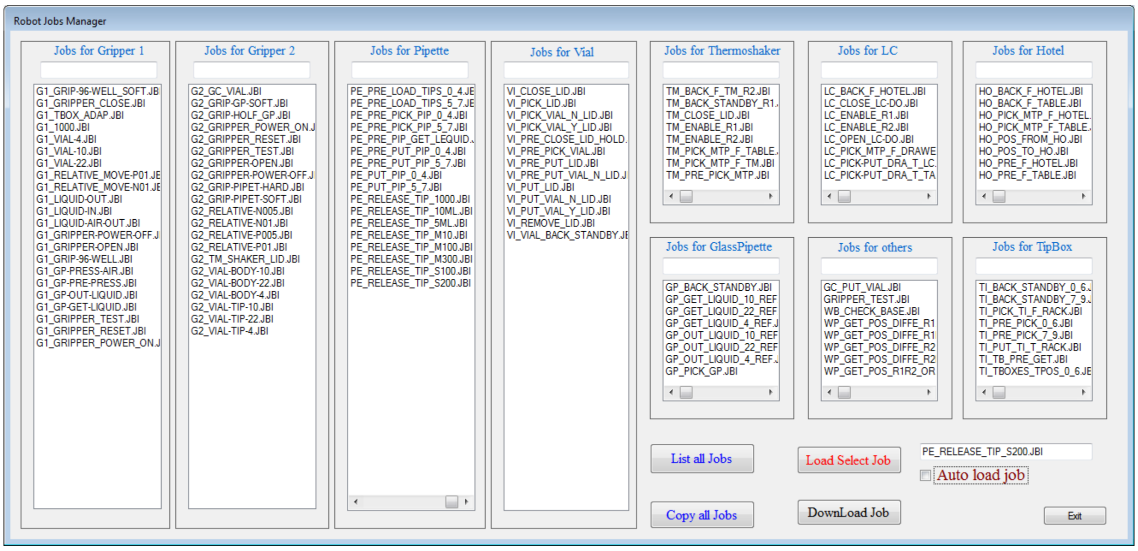
[ ] Record Task Jobs

( ) Normal Operation

(●) Advanced Operation

Figure 103. The main user interface of R-interface.

Figure 104 shows the tool for managing robot jobs. By using it, the user can get the list of robot jobs from the robot controller FS100, download jobs from FS100 to PC, upload jobs form PC to FS100 and delete jobs from FS100.


Robot Jobs Manager

Jobs for Gripper 1

G1_GRIP-96-WELL_SOFT.JB
G1_GRIPPER_CLOSE.JBI
G1_TBOX_ADAP.JBI
G1_1000.JBI
G1_VIAL-4.JBI
G1_VIAL-10.JBI
G1_VIAL-22.JBI
G1_RELATIVE_MOVE-P01.JE
G1_RELATIVE_MOVE-N01.JE
G1_LIQUID-OUT.JBI
G1_LIQUID-IN.JBI
G1_LIQUID-AIR-OUT.JBI
G1_GRIPPER-POWER-OFF.J
G1_GRIPPER-OPEN.JBI
G1_GRIP-96-WELL.JBI
G1_GP-PRESS-AIR.JBI
G1_GP-PRE-PRESS.JBI
G1_GP-OUT-LIQUID.JBI
G1_GP-GET-LIQUID.JBI
G1_GRIPPER_TEST.JBI
G1_GRIPPER_RESET.JBI
G1_GRIPPER_POWER_ON.J

Jobs for Gripper 2

G2_GC_VIAL.JBI
G2_GRIP-GP-SOFT.JBI
G2_GRIP-HOLF_GP.JBI
G2_GRIPPER_POWER_ON.J
G2_GRIPPER_RESET.JBI
G2_GRIPPER_TEST.JBI
G2_GRIPPER-OPEN.JBI
G2_GRIPPER-POWER-OFF.J
G2_GRIP-PIPET-HARD.JBI
G2_GRIP-PIPET-SOFT.JBI
G2_RELATIVE-N005.JBI
G2_RELATIVE-N01.JBI
G2_RELATIVE-P005.JBI
G2_RELATIVE-P01.JBI
G2_TM_SHAKER_LID.JBI
G2_VIAL-BODY-10.JBI
G2_VIAL-BODY-22.JBI
G2_VIAL-BODY-4.JBI
G2_VIAL-TIP-10.JBI
G2_VIAL-TIP-22.JBI
G2_VIAL-TIP-4.JBI

Jobs for Pipette

PE_PRE_LOAD_TIPS_0_4.JE
PE_PRE_LOAD_TIPS_5_7.JE
PE_PRE_PICK_PIP_0_4.JBI
PE_PRE_PICK_PIP_5_7.JBI
PE_PRE_PIP_GET_LEQUID.
PE_PRE_PUT_PIP_0_4.JBI
PE_PRE_PUT_PIP_5_7.JBI
PE_PUT_PIP_0_4.JBI
PE_PUT_PIP_5_7.JBI
PE_RELEASE_TIP_1000.JBI
PE_RELEASE_TIP_10ML.JBI
PE_RELEASE_TIP_5ML.JBI
PE_RELEASE_TIP_M10.JBI
PE_RELEASE_TIP_M100.JBI
PE_RELEASE_TIP_M300.JBI
PE_RELEASE_TIP_S100.JBI
PE_RELEASE_TIP_S200.JBI

Jobs for Vial

VI_CLOSE_LID.JBI
VI_PICK_LID.JBI
VI_PICK_VIAL_N_LID.JBI
VI_PICK_VIAL_Y_LID.JBI
VI_PRE_CLOSE_LID_HOLD.
VI_PRE_PICK_VIAL.JBI
VI_PRE_PUT_LID.JBI
VI_PRE_PUT_VIAL_N_LID.J
VI_PUT_LID.JBI
VI_PUT_VIAL_N_LID.JBI
VI_PUT_VIAL_Y_LID.JBI
VI_REMOVE_LID.JBI
VI_VIAL_BACK_STANDBY.JE

Jobs for Thermoshaker

TM_BACK_F_TM_R2.JBI
TM_BACK_STANDBY_R1.
TM_CLOSE_LID.JBI
TM_ENABLE_R1.JBI
TM_ENABLE_R2.JBI
TM_PICK_MTP_F_TABLE.
TM_PICK_MTP_F_TM.JBI
TM_PRE_PICK_MTP.JBI

Jobs for LC

LC_BACK_F_HOTEL.JBI
LC_CLOSE_LC-DO.JBI
LC_ENABLE_R1.JBI
LC_ENABLE_R2.JBI
LC_OPEN_LC-DO.JBI
LC_PICK_MTP_F_DRAWE
LC_PICK-PUT_DRA_T_LC.
LC_PICK-PUT_DRA_T_TA

Jobs for Hotel

HO_BACK_F_HOTEL.JBI
HO_BACK_F_TABLE.JBI
HO_PICK_MTP_F_HOTEL.
HO_PICK_MTP_F_TABLE.
HO_POS_FROM_HO.JBI
HO_POS_TO_HO.JBI
HO_PRE_F_HOTEL.JBI
HO_PRE_F_TABLE.JBI

Jobs for GlassPipette

GP_BACK_STANDBY.JBI
GP_GET_LIQUID_10_REF
GP_GET_LIQUID_22_REF
GP_GET_LIQUID_4_REF.J
GP_OUT_LIQUID_10_REF
GP_OUT_LIQUID_22_REF
GP_OUT_LIQUID_4_REF.J
GP_PICK_GP.JBI

Jobs for others

GC_PUT_VIAL.JBI
GRIPPER_TEST.JBI
WB_CHECK_BASE.JBI
WP_GET_POS_DIFFE_R1
WP_GET_POS_DIFFE_R1
WP_GET_POS_DIFFE_R2
WP_GET_POS_DIFFE_R2
WP_GET_POS_R1R2_OR

Jobs for TipBox

TI_BACK_STANDBY_0_6.J
TI_BACK_STANDBY_7_9.J
TI_PICK_TI_F_RACK.JBI
TI_PRE_PICK_0_6.JBI
TI_PRE_PICK_7_9.JBI
TI_PUT_TI_T_RACK.JBI
TI_TB_PRE_GET.JBI
TI_TBOXES_TPOS_0_6.JE

PE_RELEASE_TIP_S200.JBI

[ ] Auto load job

Figure 104. Tool for managing robot jobs.

When listing the jobs, they will be separate to group, so it is easy to find a job from the list. All of the robots can be copied to a PC by clicking the button “Copy all Jobs”. It is also able to just download one or several by selecting them from the lists. For deleting some jobs from FS100, select them from the list and then click the button “Delete job”.

Figure 105 shows the tool for reading the status of the robot: “IsPlay” will be checked, if the robot is moving; “Is Teach” will be checked, if the robot is in “Teach” model; and it is similar to the others. The tool is used to test if R-interface can read the robot status correctly.

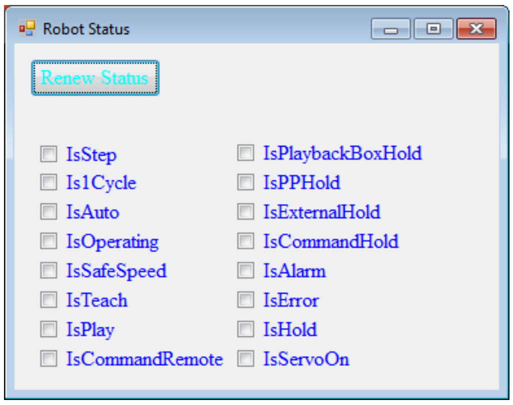
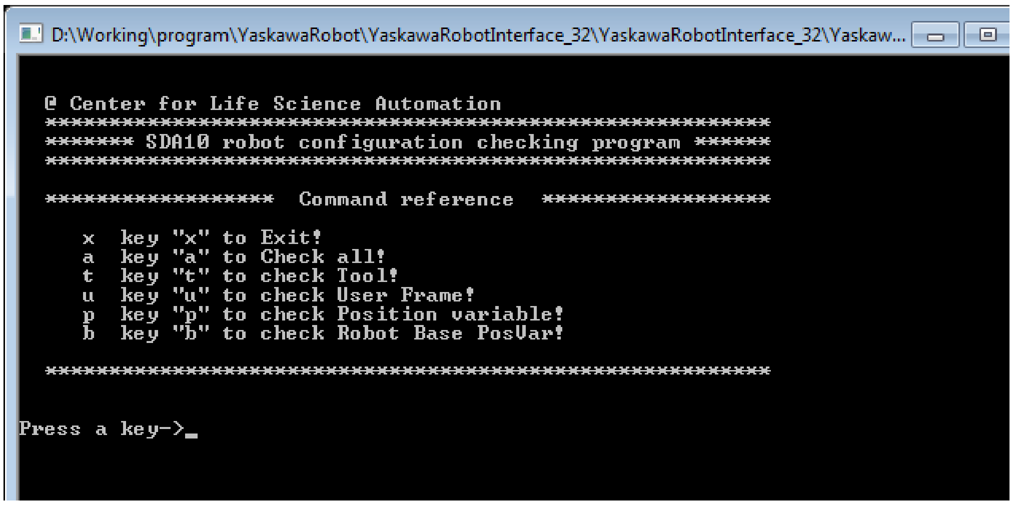


Figure 105. Tool for reading the status of the robot.

Figure 106 shows the tool for checking the configuration of the robot system, including the robot tools, user frames, the position variables and the base position variables. The user can input related keys from the keyboard to performing checking. The “x” key is used to exit the tool. The “a” key is used to perform a complete checking. The “t” key is used to perform a checking about the robot tool settings. The “u” key is used to check the defined user frames. The “p” key is used to check the position variables and the “b” key is used to check the base position variables. These configurations are critical. If they are changed unintentionally, it may cause failures to the robot. When this tool is loaded, the configuration files about robot tools, user frames and variables will be downloaded to the computer. Then, their contents will be compared with those of the previous copies. The differences will be shown to the user.



```
D:\Working\program\YaskawaRobot\YaskawaRobotInterface_32\YaskawaRobotInterface_32\Yaskaw...

  @ Center for Life Science Automation
  *********************************************************
  ******* SDA10 robot configuration checking program *******
  *********************************************************

  ******************  Command reference  ******************

     x  key "x" to Exit!
     a  key "a" to Check all!
     t  key "t" to check Tool!
     u  key "u" to check User Frame!
     p  key "p" to check Position variable!
     b  key "b" to check Robot Base PosVar!

  *********************************************************


Press a key->_
```

Figure 106. Tool for checking the configuration of the robot system.

Chapter 10 Integration with SAMI Software

In order to reduce the requirement of programming for the robot and also automatically start the sample analysis processes, the robot system, LC-MS system and GC-MS system have been integrated with SAMI EX workstation system. By using its friendly graphical interface, the user can design the sample preparation and analysis processes. Some examples about SAMI method can be seen in Appendix C. The software integrations between SAMI system and LC-MS system, between SAMI system and GC-MS system and between SAMI system and the robot system will be discussed in the following parts of this chapter.

10.1 System Configuration

The LC-MS is controlled by a LC-MS data acquisition software (LC-MS) software, which runs on a personal computer (PC). A GC-MS data acquisition software (GC-MS) software running on another PC controls the GC-MS. The FS100 robot controller controls the dual arm robot. SAMI software contacts with them through local network. The configuration can be seen in Figure 107.

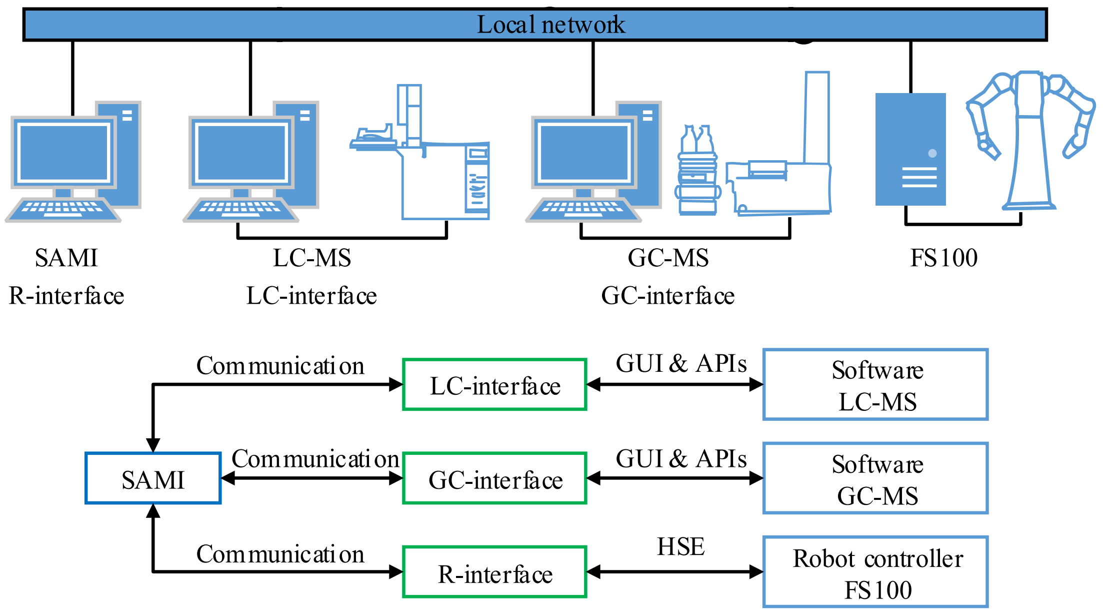


Figure 107. System configuration.

For integrating their software with the SAMI software, three program interfaces have been built separately:

1) LC-interface is used to integrate the LC-MS software with the SAMI and runs on the same PC with the LC-MS software. On one side, the LC-interface can interact with LC-MS software based on the graphical user interface and the APIs (more details can be seen in Chapter 8). On other side, the LC-interface can communicate with SAMI software through the local network.

2) GC-interface is used to integrate the GC-MS software with the SAMI and runs on the same PC with the GC-MS software. The GC-interface works in a similar way with the LC-interface:

interacting with GC-MS software based on the graphical user interface and the APIs (more details can be seen in Chapter 8) and interacting with SAMI software by communicating through the local network.

3) R-interface is used to integrating the dual-arm robot system with the SAMI system and runs on the same PC with the SAMI software. R-interface uses the HSE protocol to communicate with the FS100 controller through local network. The interactions between R-interface and the robot have been discussed seen in Chapter 9. Though the R-interface and the SAMI software are configured on the same computer, their interactions are also achieved through communications. If the R-interface and SAMI software are configured on different computers, they are able to communicate with each other through local network.

10.2 Commands Receiving

Two threads will be used to deal with SAMI commands: scanning-thread and processing-thread. The scanning-thread is used to scan the data buffer to check whether new data has been received. If received, the command will be read from data buffer and delivered to the processing-thread to process it. If not, the scanning thread will go on to scan the data buffer. The processing-thread is started by the scanning-thread when the first SAMI command is received. Once started, the processing-thread will work in loops. If new command is revived, the processing-thread will process it and then wait for the next new commands.

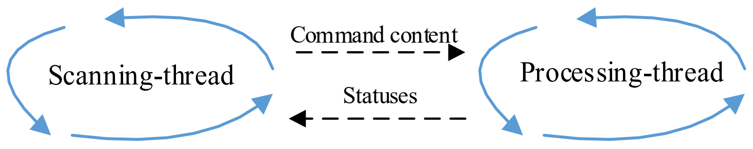


Figure 108. Threads for scanning and processing command.

10.2.1 Scanning Thread

SAMI commands will be sent through the local network to the program interfaces. The program will keep scanning the data buffer to detect whether new commands, which is done by using a thread. In this way, when waiting for receiving SAMI commands, the program interface is still able to perform the other work. This thread is called scanning-thread. The scanning-thread is mainly used to receive the commands sent from SAMI software to the program interface. At the beginning of running the thread, a no GC (Garbage Collection) region with 8.0 x 10^6 bytes will be set to disallow the automated garbage collection during the execution of SAMI commands. And for controlling the allocated memory, enforced garbage collections will be performed in the following process. Then the data buffer will be checked. If no new data is received, it will keep check the data buffer. If received, the command data will be read out and then the status of the processing-thread will be checked. If it is still busy, the scanning-thread will wait until it become idle. So, it can be seen that only one SAMI command can be executed at the same time. Next, the allocated memory will be read. When the allocated memory exceeds the value of 3.0 x 10^6, the program will force an immediate garbage collection to release the memory.

If it is the first SAMI command received by the scanning-thread, the processing-thread will be started. After this, the processing-thread will keep alive, until the program is closed. So, for the following commands, it does not need to start the processing-thread again. The content of the SAMI command will be delivered to the processing-thread and a mark will be made to inform the processing-thread that a new command is ready to be processed. Then, the scanning-thread will wait until the new received commanded is marked as under processing. If the socket is closed, the scanning-thread will be terminating. Otherwise, it will go back to check the data buffer to detect new commands.

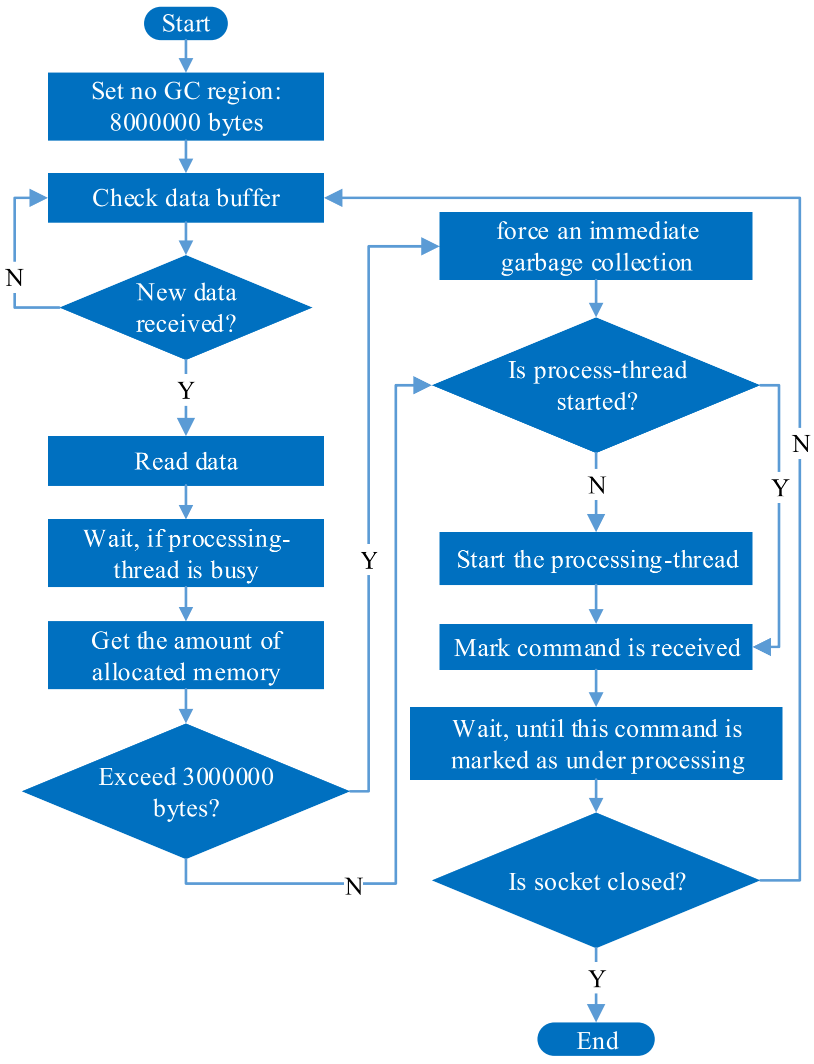

Start
Set no GC region: 8000000 bytes
Check data buffer
New data received?
N
Y
Read data
Wait, if processing-thread is busy
Get the amount of allocated memory
Exceed 3000000 bytes?
Y
N
force an immediate garbage collection
Is process-thread started?
N
Y
Start the processing-thread
Mark command is received
Wait, until this command is marked as under processing
Is socket closed?
N
Y
End

Figure 109. Flow chart of scanning-thread.

10.2.2 Processing Thread

The SAMI command will be processed by another thread, called processing-thread. The processing-thread is started by the scanning-process, when the first command is received. When a new command is received by the scanning-thread, it will check if the processing-thread is already started. If not, the scanning-thread will start the processing-thread to process the new received command. After the new command is delivered from the scanning-thread, the processing-thread will start to process it, beginning with marking the thread as busy and the command as under processing. These marks are used for the scanning-process to check the status of the processing-thread. Then, the program will check if the command is a valid one. If the command does not have a legal format, the command will be ignored. If it is a valid one, the program will call relate methods to execute it. After the command is executed completely, the processing-thread will be marked as idle, so it is ready to execute the next command. If the processing-thread is forced to be closed, then it will go to the end. Otherwise, it will go back to check for new command.

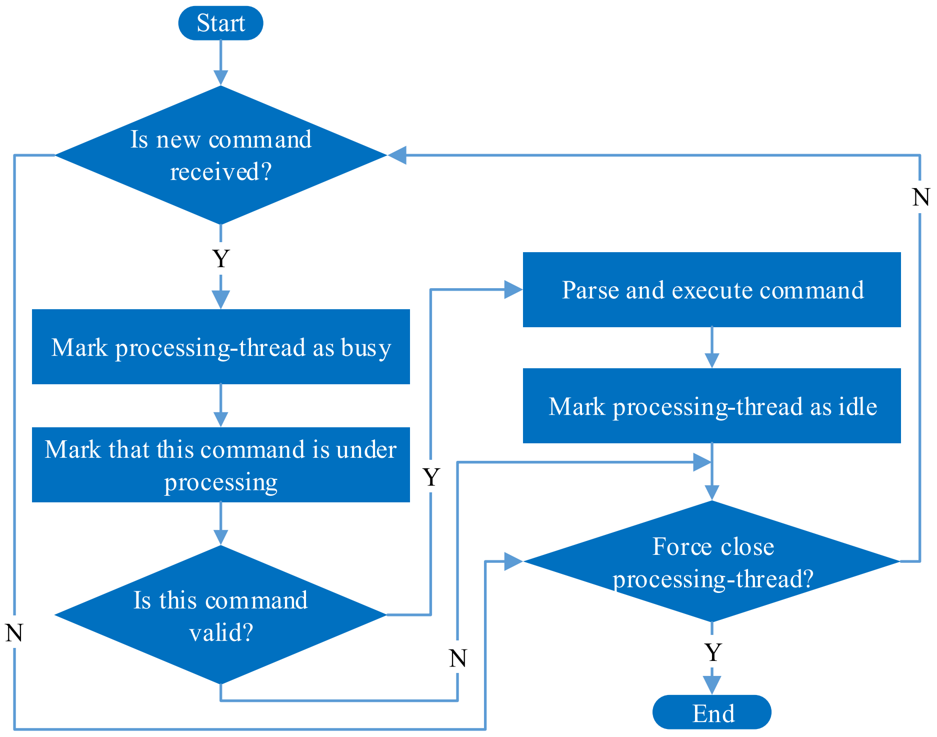

Start
Is new command received?
Y
N
Mark processing-thread as busy
Mark that this command is under processing
Is this command valid?
Y
N
Parse and execute command
Mark processing-thread as idle
Force close processing-thread?
N
Y
End

Figure 110. Flow chart of processing-thread.

10.3 Commands Processing

As the controls to the LC-MS and the GC-MS are similar, the LC-interface and the GC-interface are also work in the similar way to parse and execute SAMI commands. Table 6 shows the list of commands supported by LC-interface and GC-interface and the flow chart of executing commands can be seen in Figure 111. After the command is identified as valid, the name and the ID of the command will be extracted. Then, the command name will be used as the identifier to call related methods to fulfill the command (more information can be seen in Chapter 8). As the command parsing and executing of R-interface is much more complex, it will be mainly discussed in this section.

Table 6. Commands supported by LC-interface and GC-interface.

Command Name	**Realization**
“Version”	Get the version the LC-interface and GC-interface.
“List Methods”	Get the list of LC-MS methods or GC-MS methods.
“List Projects”	Get the list of created projects.
“Status”	Get the status of LC-MS instrument or GC-MS instrument: idle, busy or error.
“MethodRun”	This command is used to start a measurement. In the process of executing this command, worklist/sequence will be loaded, LC-MS

	method or GC-MS method will be loaded and then the measurement is started.
“Abort”	Abort the measurement.
“Stop”	Stop the measurement (not supported by GC-interface).
“Continue”	Resume the measurement (not supported by GC-interface).

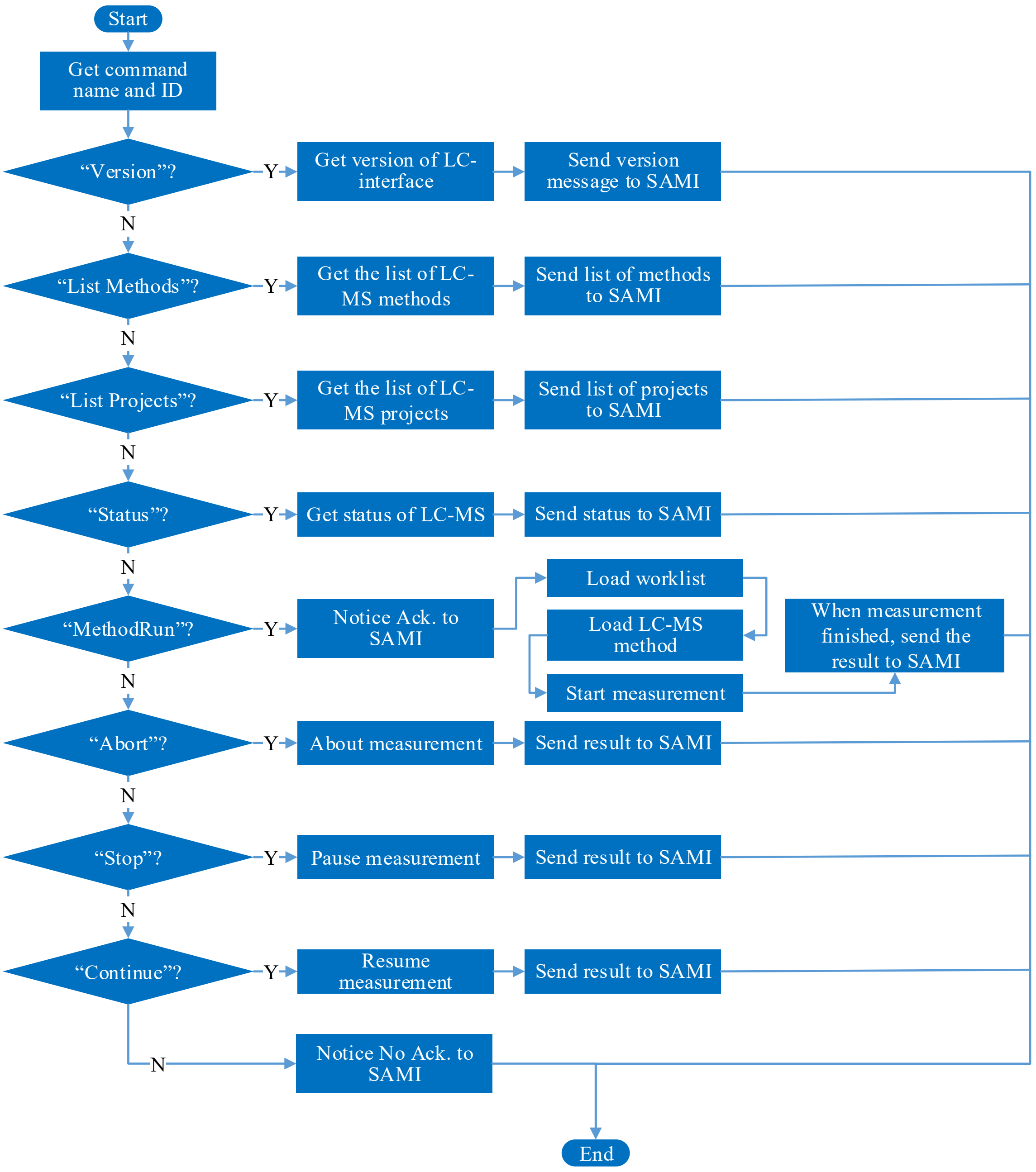


Figure 111. Flow chart of executing commands by LC-interface.

10.3.1 Command Protocol for Controlling Robot

This part mainly discusses the commands related with the processes of sample preparation and transport, including “MethodRun” command, “Move” command, “Delid” command and “Lid” command. The process of executing a SAMI command is shown in Figure 112.

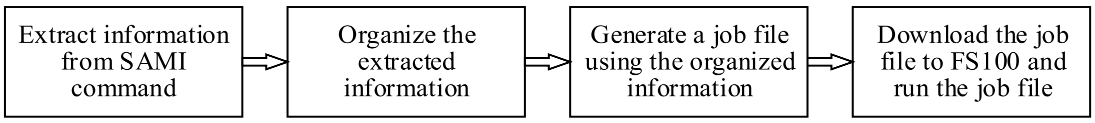

Extract information from SAMI command
Organize the extracted information
Generate a job file using the organized information
Download the job file to FS100 and run the job file

Figure 112. Process of executing commands.

First, the information included in the command will be extracted and then will be organized in certain format (XML data). Next, the organized information will be used to generate a call-file, which integrates robot motion elements. The generation of a call-file is discussed in section 9.2.3. After that, the call-file will be downloaded to the robot controller (FS100). When this call-file is executed on FS100, the robot will be driven to perform the related tasks.

10.3.1.1 "MethodRun" Command

The "MethodRun" command is used to require the robot to pipette solutions using pipettes or glass pipettes, and to filter solutions using syringes. Figure 113 gives the flow chart of executing a "MethodRun" command.

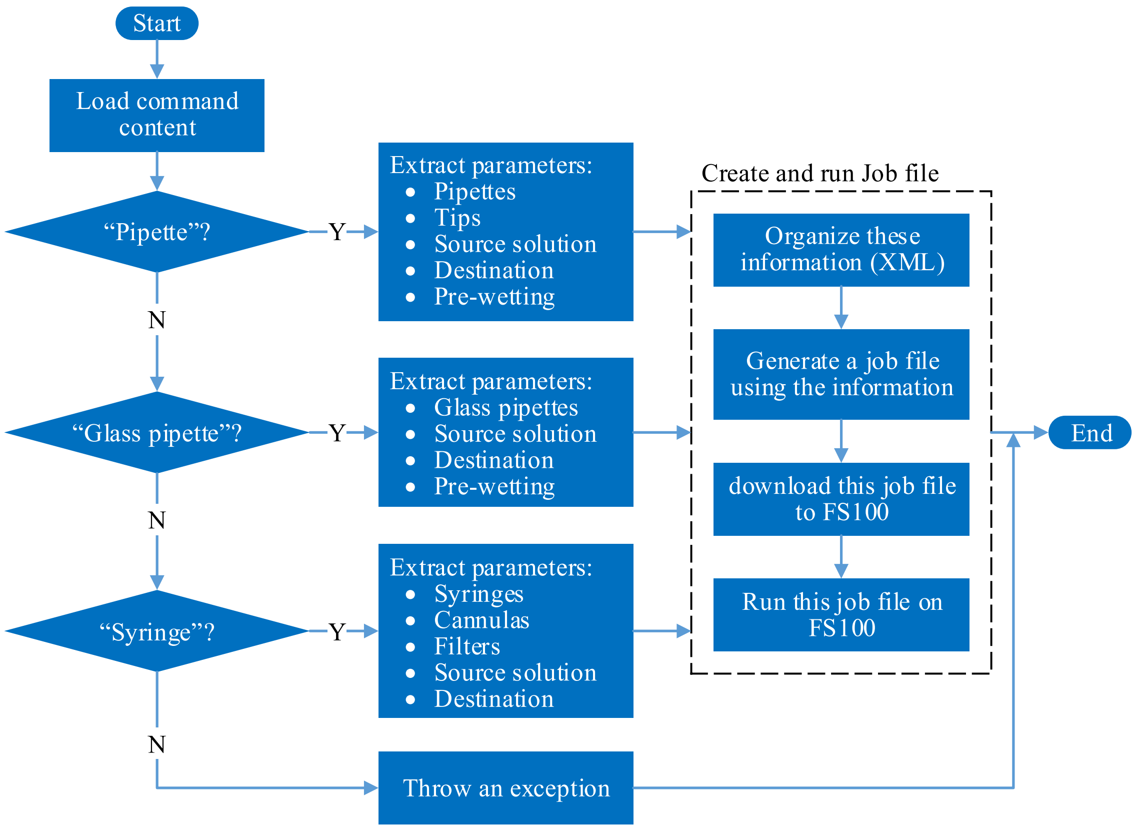


Figure 113. Flow chart of executing "MethodRun" command.

With different parameters, this command can be used to order the robot to perform pipetting by pipettes or glass pipettes, or to filter solutions using syringes. If the command is about "Pipette", the parameters will be extracted from SAMI command, including:

1) Pipettes. As different pipettes are supported (see Figure 28), the information about the pipette going to be used is necessary.
2) Tips. Which tips to be used (The positions of the tips).

3) Source solution. The source solution can be in vials, reservoirs or microplates. This information gives where to get the solutions.
4) Destination. The destination can also be vials, reservoirs or microplates, giving where to pipette the solutions to.
5) Pre-wetting. Sometime it needs to pre-wet tips. And this can be determined by SAMI commands.

The parameters for “Glass pipette” is some different, as the glass pipettes do not need tips and are used to transfer solutions between vials. Its parameters should include: which glass pipettes to use, where the solution is now (in which vial) and where to pipette the solution (to which vial). The command about “Syringe” is used to drive the robot to using syringes to filter solutions, which are contains will vials. The information about which syringes, which cannulas and which filters to used, where the source solutions are and where to filter the solutions, is required.

10.3.1.2 "Move" Command

The "Move" command is used to instruct the robot to transport vials, microplates, tip boxes, the lid of thermo shaker, auto sampler trays. The flow chart of executing a "Move" command is shown in Figure 114.

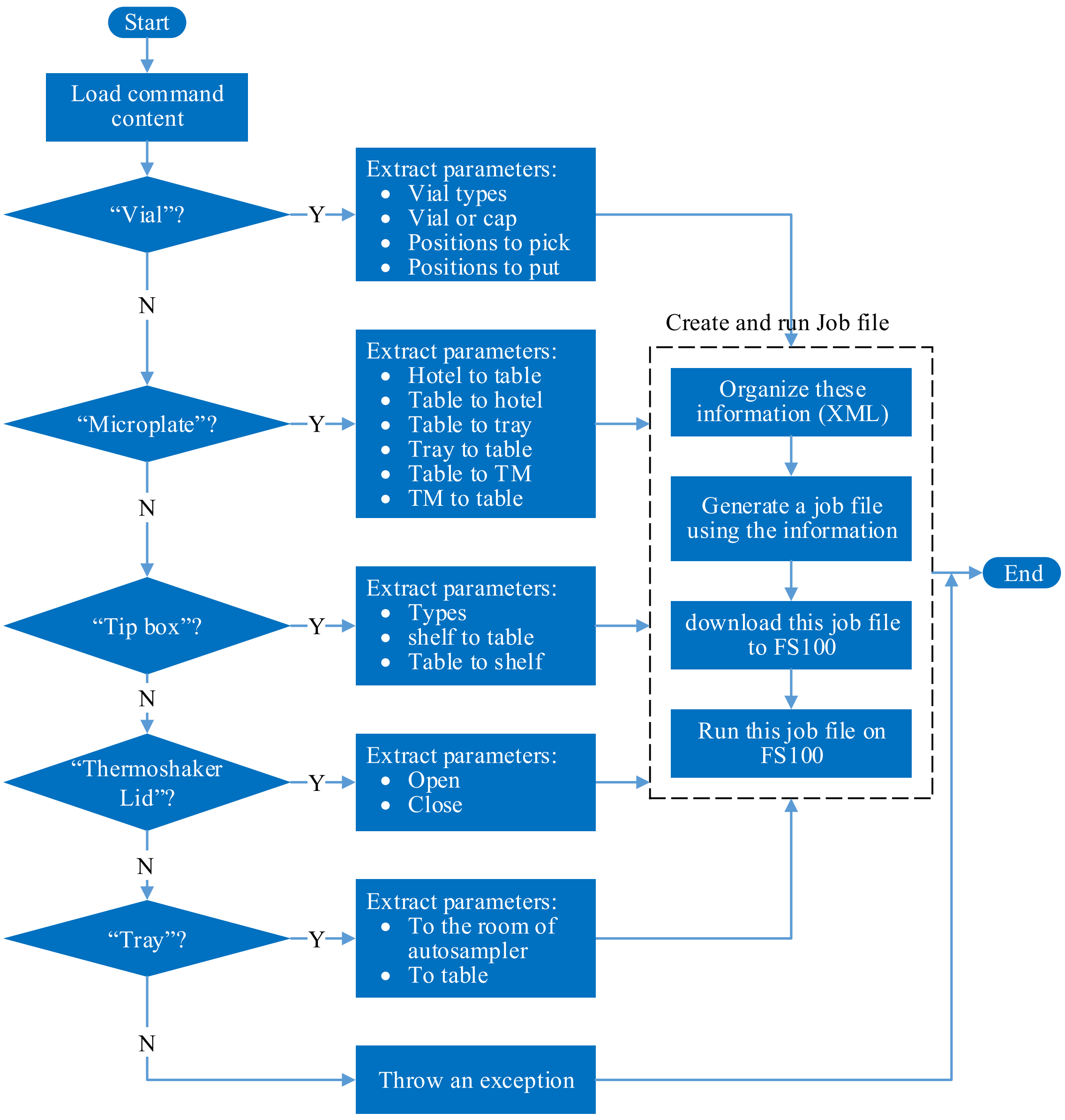


Figure 114. Flow chart of executing "Move" command.

With different parameters, the "Move" command can be used to transfer different lab ware, including the following cases:

1) In case of "Vial", the command is used to move vials and the information about the vial types, positions of vials and positions to put vials will be needed. Also, the "Move" command can be used to transfer the caps of vials if required.
2) In the case of "Microplate", the "Move" command can be used to transfer microplates between hotel and table, between table and the tray of auto sampler and between table and the thermo shaker.

3) In the case of “Tip box”, the command is used to transfer tip boxes between the shelf and the table.
4) In the “Thermo shaker Lid” case, the command is used to move the lid away to open the thermo shaker, so that the microplate can be transferred to the thermo shaker. After the microplate is transferred to the thermo shaker, the “Move” command can also be used to move the lid back to close the thermo shaker.
5) In the case of “Tray”, the command is used to transfer the tray between the table and the LC auto sampler. The prepared samples are placed in the microplate, which will be transferred to the tray of LC auto sampler by using the “Move” command. And also by using the “Move” command, the tray (with samples) can be transferred into the auto sampler to analyze and move out from the auto sampler after analyzed.

10.3.1.3 “Delid” and “Lid” Command

The “Delid” command is used to unscrew cap from vial or to remove lid from microplate and the “Lid” command is used to screw cap on vial and to cover lid to microplate. For “Delid” or “Lid” vials, the information about the positions of vials and caps, and the vial types should be given. For “Delid” or “Lid” microplate, the information about the position of microplate (on table, on the tray or in the thermo shaker) and the position of the microplate lid should be given.

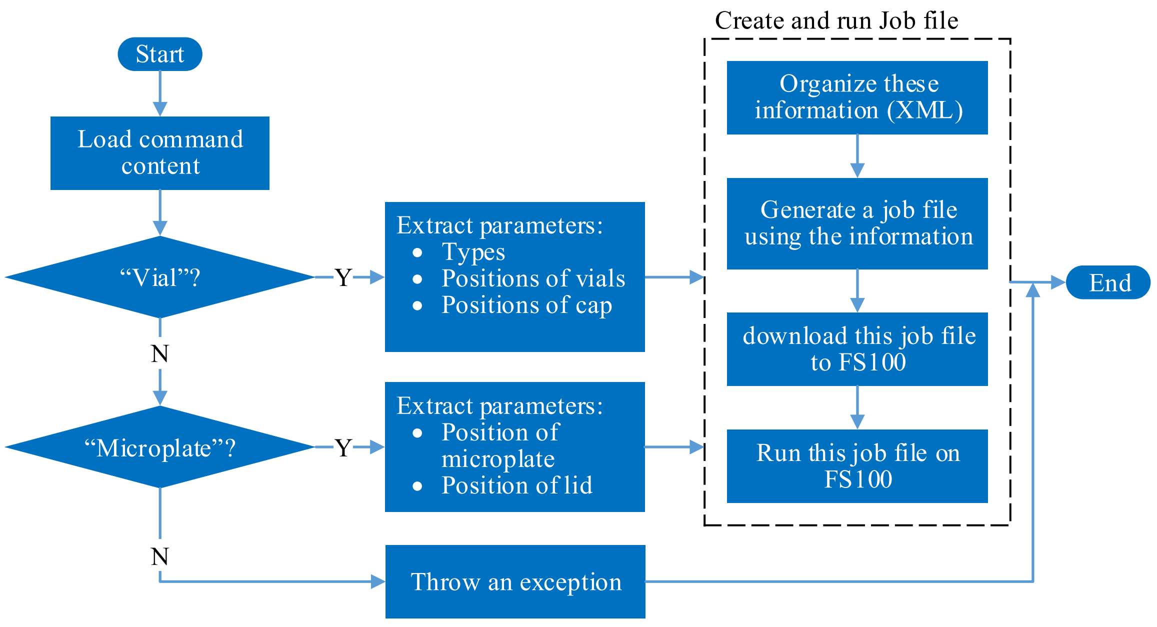


Figure 115. Flow chart of executing “Delid”/ “Lid” command.

10.4 Evaluation of Performance of Garbage Collector

In the .Net Framework, a garbage collector (GC) is used to allocate and release the memory. The garbage collector will work to free some unused memory at an unknown time, which is determined by the garbage collector itself. However, it seems that the garbage collector can cause fatal errors to the application program when releasing the memory. This fault will affect reliabilities of the program interface. Several tests about garbage collection will be discussed in this section.

10.4.1 Argument 1: Causing Errors Probabilistically

It seems that some errors can occur probabilistically when garbage collection is performing. As it is said that, the garbage collector manages the memory automatically. And this can be seen in Figure 116. The horizontal axis represents the count of executed commands and the vertical axis represents the used memory with byte as unit. From Figure 116, it can be seen that, when new command is executed, more memory will be allocated. So, the allocated memory will increase. However, when the allocated memory reaches certain level, it will decease obviously. This is because that the garbage collector has released the waste memory. From the trend of memory changing in Figure 116, it seems that the garbage collector will work to release the waste memory, when the allocated memory reaches certain levels. A fatal error happened, when the program was executing the 392th command. When compared with the front “saw teeth”, it was possible that the garbage collector was executing the release of memory when the error happened. The situation happened in Figure 116 is not an individual case. The similar results can be seen in Figure 117, Figure 118 and Figure 119. The trends of memory changing in these figures are similar. A fatal error happened when executing the 412th command in Figure 117, the 220th command in Figure 118 and the 207th command in Figure 119. But it seems that the reasons for the errors are similar. In order to find more proofs, more experiments have been shown in the following parts.

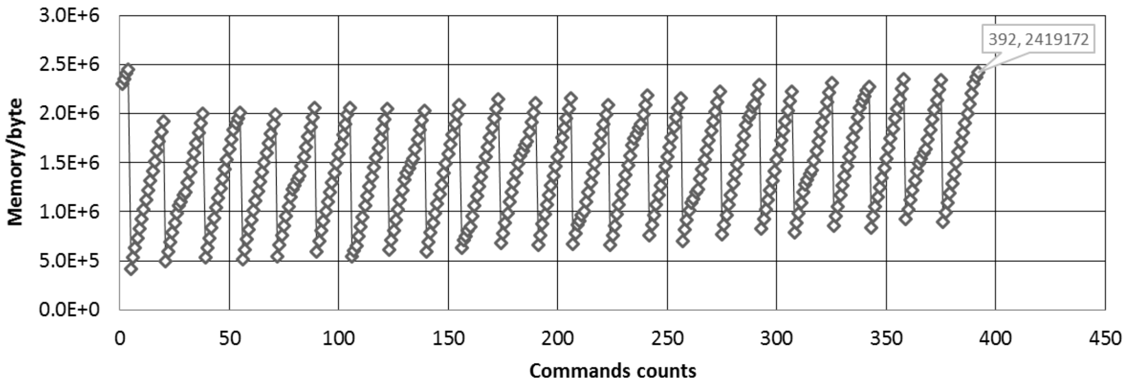


Figure 116. Allocated memory is changing when executing commands. An error happened when executing 392th command.

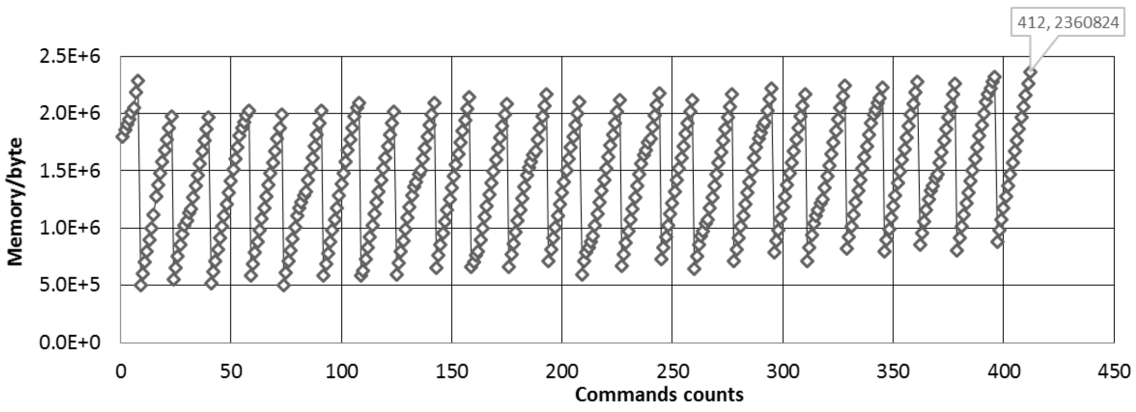


Figure 117. Allocated memory is changing when executing commands. An error happened when executing 412th command.

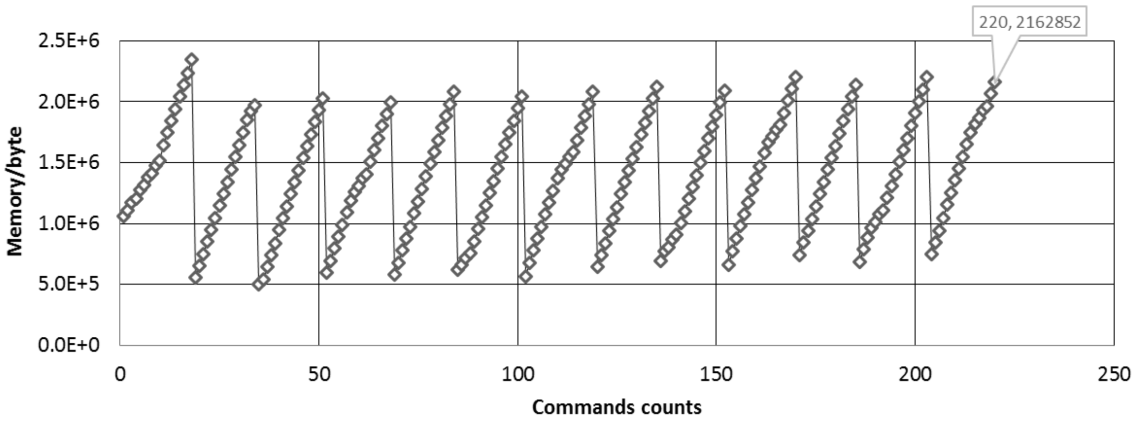


Figure 118. Allocated memory is changing when executing commands. An error happened when executing 220th command.

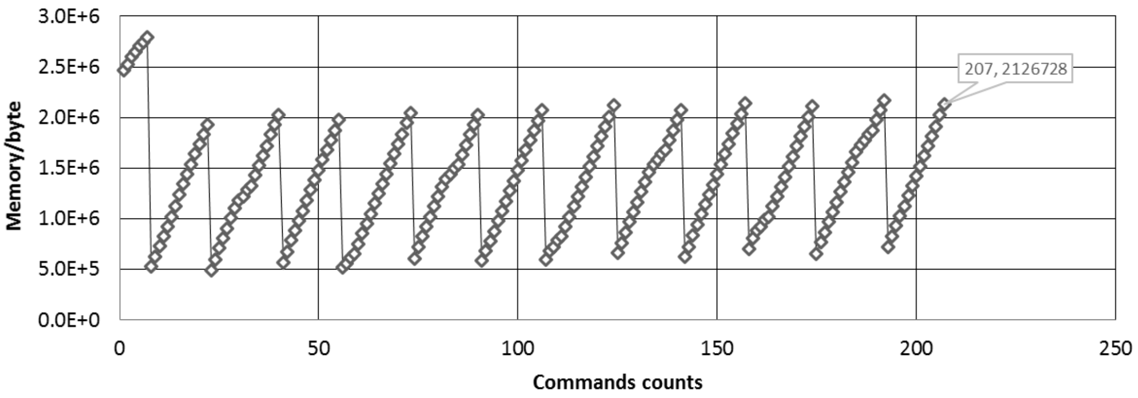


Figure 119. Allocated memory is changing when executing commands. An error happened when executing 207th command.

10.4.2 Argument 2: Forced Garbage Collection

In this experiment, a forced garbage collection will be called immediately after received each six commands. Figure 120 gives the trend of memory changing. Each “saw tooth” had six diamonds, representing six commands (with the first “saw tooth” ignored).

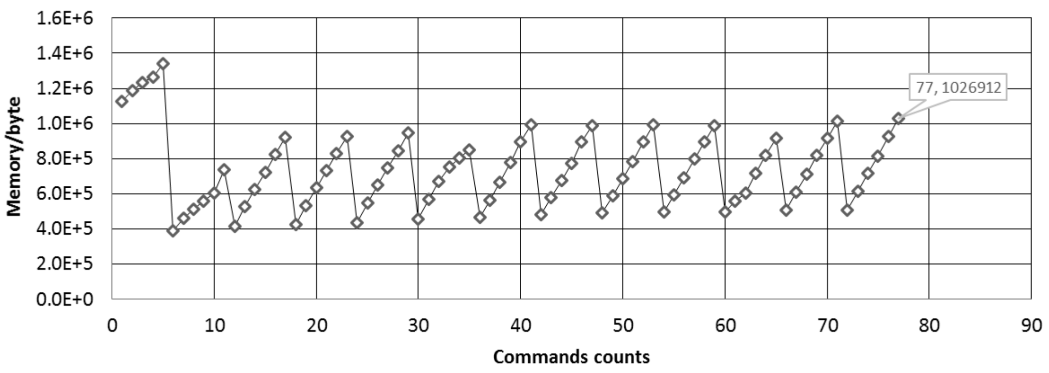


Figure 120. A forced garbage collection had been called immediately when received each six commands. An error happened when executing 77th command.

With the executions of commands, the allocated memory increased first and then decreased after the "6th" command (the top one at each "saw tooth"). A fatal error happened at the 77th command. As this was one of the "6th" commands, which means that a forced garbage collection had called when executing this command, so it can be seen that garbage collections can cause errors to the application program probably, but not always.

10.4.3 Argument 3: No Garbage Collection Region

This experiment can give more proofs, by monitoring the memory changing in a no garbage collection region. The no garbage collection region is used to attempt to disallow garbage collection within a specified amount of memory. In this experiment, the specified amount of memory is 8.0E+6. From Figure 121, it can be seen that the allocated memory was keeping increasing approximately and no garbage collection was performed before the allocated memory reached 8.0E+6 bytes (with the changing of memory at the beginning commands ignored). When the allocated memory exceeded 8.0E+6 bytes, garbage collections were allowed. And a fatal error happened when executing the 91th command. From argument 2, it can be known that a garbage collection can be forced to perform. And from argument 3, it can be known that no garbage collection regions can be set to attempt to disallow garbage collections. So, what will happen when forcing garbage collections in no garbage collection regions? The result will be show in the following part.

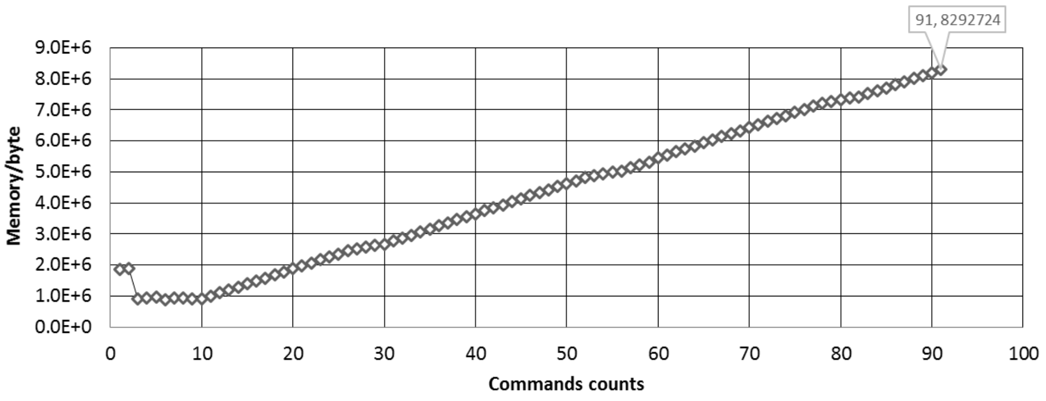


Figure 121. The trend of memory changing in no garbage collection. When the allocated memory reaches 8.0E+6 bytes, garbage collections will be allowed. An error happened when executing 91th command. At that moment, the allocated memory exceeded 8.0E+6 bytes.

10.4.4 Argument 4: Forced Garbage Collection in NoGCRegion

In this experiment, a NoGCRegion (no garbage collection region) was set with 8.0E+6 bytes in order to disallow automated garbage collections. And when it is detected that the allocated memory exceeds 3.0E+6 bytes, a forced garbage collection will be called before executing critical codes. From Figure 122, it can be seen that the allocated memory was controlled under 3.0E+6 bytes roughly. The more important is that no error had happened with 10400 commands having been executed successfully.

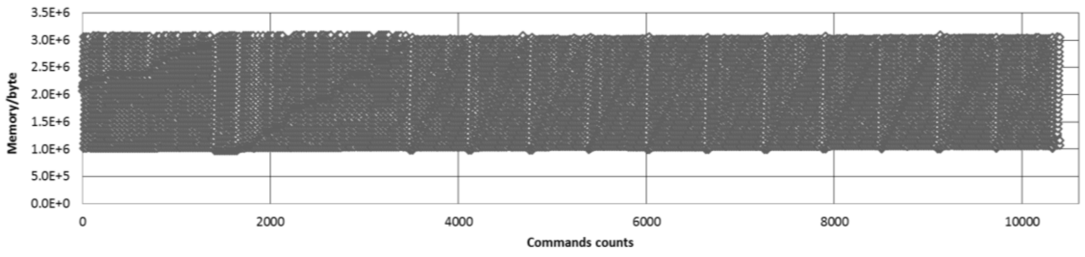


Figure 122. Force a garbage collection when allocated memory exceeded 3.0E+6 bytes. No error had happened with 10400 SAMI commands having been executed successfully.

10.4.5 Summary

Argument 1, 2 and 3 indicate that garbage collection is possible to cause fatal errors to the application program. Argument 2 shows that garbage collections can be performed by forced. And from argument 3, it can be seen that no garbage collection regions can be set to disallow automated garbage collections. An automated garbage collection will be performed at an unknown time. That could be the reason of causing errors. So, the solution is performing garbage collections at a known and safe time: first, set a no garbage collection region with enough memory; second, force garbage collections before executing critical codes. Argument 4 proves that this is an available way.

Chapter 11 Applications

11.1 General Testing

11.1.1 Testing Results and Improvements

In order to evaluate the performance of the dual-arm robot platform in real applications two different sample preparation processes have been automated and tested. In the process for the determination of chiral compound more than 1,000 robot subtasks will be executed, while more than 700 subtasks have to be executed in a sample preparation process for the determination of cholesterol. Some subtasks will be carried out for several times in a process. For example, in a process about chiral compound, the subtask – "unscrew cap from vial" will be conducted 10 times, to open ten different vials. The results of executing key subtasks have been recorded in appendix (section "D Testing Records").

The tests about sample preparation process of chiral compound had been conducted 19 times. The summary of failures in the tests has been given in Table 7 (more information can be seen in appendix D.1).

Table 7. Summary of failures in 19 tests about chiral compound.

No.	Failed subtasks	Description
1	Load tip to pipet	Eight tips should be loaded. In failures, at least one tip was not loaded to the pipet.
2	Release tip from pipet	In failures, at least one tip was not released from the pipet.
3	Pipet liquids	Liquid kept dropping when moving pipet, as tip was not fixed very well on the pipet.
4	Pick vial from rack	Rack was dragged a little up.
5	Release glass pipet	Glass pipet fell down from rack.
6	Transfer microplate from LC-tray to table	Failed to pick a microplate from the tray.
7	Transfer lid of microplate	Failed to grip lid of microplate.

The tests about sample preparation process of cholesterol had been conducted 36 times. The summary of failures in tests has been shown in Table 8 (more details can be seen in appendix D.2).

Table 8. Summary of failures in 36 tests about cholesterol.

No.	Failed subtasks	Description
1	Put cap on rack	Cap was not put on the rack correctly.

2	Pipet liquids	Liquid kept dropping when moving pipet, as tip was not fixed very well on the pipet.
3	Pick vial from rack	Rack was dragged from the ALP completely.
4	Screw cap on vial	Failed to screw cap on vial, with cap not screwed correctly or fell down from vial.
5	Filter liquid by syringe	Collision happened between cannula and vial, when moving cannula into a vial to draw liquid. Collision happened between filter and vial, when filtering liquids.

According the failures shown in Table 7 and Table 8, improvements has been carried out in four parts to make the robot platform more reliable:

1. **Adjust access positions.** The reasons of the first five kinds of failures in Table 7 and the first three kinds of failures in Table 8 are related with the access position of robot arms. For example, the robot moves a pipet onto the top of a tip and then insert the pipet into the tip. The position of the robot arm, which is ready to insert the pipet into the tip, is termed as an access position. However, if this position is not very appropriate, it will fail to load the tip. The solution is to adjust the access position by moving or rotating the related robot arm in X, Y or Z direction. Normally, a slight adjustment, e.g. smaller than 1 mm (Euclidean distance), can bring a significant improvement. The reason of failing to release a tip (tips) from a pipet is due to the insufficient pressing of a button of the pipet, which can be solved by adjusting the robot position to perform a further pressing. When pipetting liquids, if a tip is not loaded on a pipet tightly, the liquid will keep dropping from the tip. The solution is adjusting the related access position to insert the pipet deeper into a tip when loading the tip. Similarly, the problems of picking a vial, putting a cap and releasing a glass pipet can be improved by adjusting access positions.

2. **Adjust robot motions**. If the designed moving path of the robot arm is not appropriate, the robot may fail to do something. The solution of picking a microplate from the LC tray has been discussed in section 11.1.2.2 and the problems about using syringes to filter liquids can also be solved by adjusting robot motions (see section 11.1.2.5).

3. **Adjust gripper finger positions**. In the failure of picking the lid of a microplate, one reason is that gripper fingers were not opened enough, which can be solved simply by adjusting gripper finger positions. When picking something, if the gripper fingers are not closed enough, it may also fail.

4. **Lab ware reasons**. The failures of screwing caps on vials are closely related to the relative position of vanish threads between cap and vial screw threads. The problem can be avoided by this way: when putting vials and caps on the rack, make sure that the relative position of their vanish threads is about 180 degrees.

After improvements, the robot platform became more reliable. In last 9 tests about chiral compounds, there were only 3 failures happened, which was much lower than the previous 10 tests with 139 failures. One failure about unscrewing caps from vials, as the vial was closed too tightly. The other two failures were about putting glass pipets to the rack. In the last 23 tests about cholesterol, there were 6 failures, which was much lower than the previous 13 tests with 53 failures.

11.1.2 Improvements about Robot Motions

Robot motions are critical and will affect the results of executing tasks. Several real examples are given in this part, to discuss special issues about robot motions.

11.1.2.1 Loading Tips

When loading tips to a 300uL 8-channel pipette, the pipette will be leant before going inside the tips, as shown in Figure 123. First, the pipette will be moved to the top of tips, and then the pipette will be leant 10 degrees. Next, the pipette will be moved inside the tips and then leant back to verticality. In this way, it is much more reliable to load tips to 300uL 8-channel pipette. The reason of loading tips in this way is that the space for the pipette going into a tip is small. It can be taken as inserting a cylinder into a ring, but the diameter of the ring is not much bigger than that of the cylinder. So, the relative position between the cylinder and the ring should be very accurate to permit a successful vertically inserting. However, in the real application such an "accurate position" is hard to be permitted. The pipette has eight channels, which means eight tips should be loaded to the pipette at the same time. But the tips in a row may not stand in a very straight line, as some of them are a little sloping, which makes it more challenging to load tips. In a vertically inserting, it is a surface going into the ring. If the cylinder is leaned first and then inserted, it will be a curve line going into the ring and so the space will be increased relatively. It is possible to load tip in a simple way – moving pipette up on tips and then going inside tips vertically. But this simple way is very possible to fail loading tips.

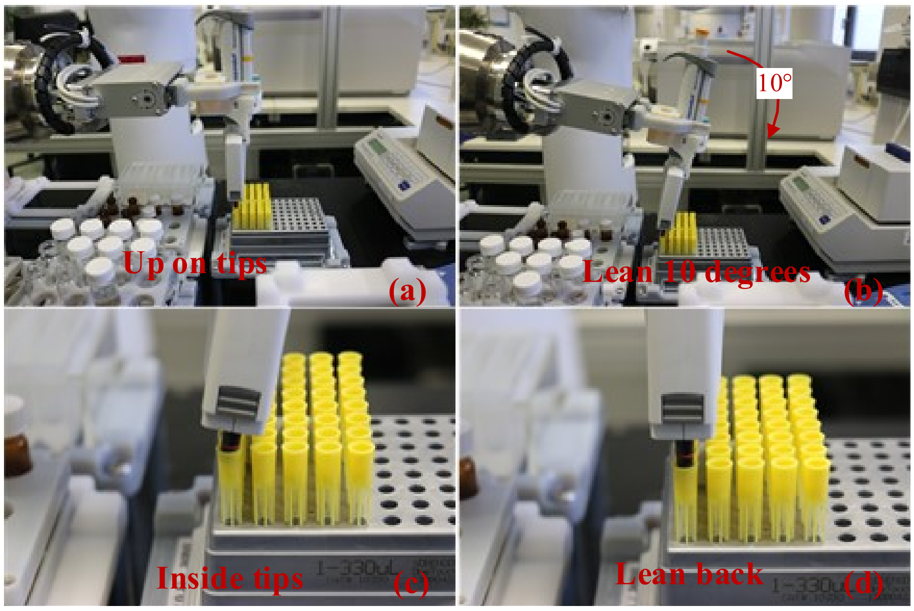


Figure 123. Loading tips to 300μL 8-channel pipette.

When loading tips to a 10μL 8-channel pipette, the pipette will be turn to left side and then right side for 2 degrees separately, in order to make the tips loaded to the pipette securely, as shown in Figure 124. Loading tips in this way is helpful to permit all tips are loaded securely on the pipette. If a tip is not loaded securely enough, liquid drawn into this tip will not reach the target volume.

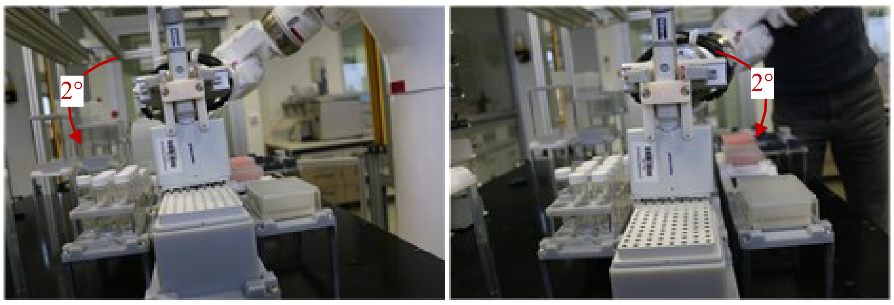


Figure 124. Loading tips to 10μL 8-channel pipette.

11.1.2.2 Transporting Microplates

After samples are prepared in a microplate, the microplate will be transported onto the tray of the LC auto sampler. The tray with the microplate will then be transported into the LC auto sampler, in order to analyze samples. From Figure 125 it can be seen that there are elastic pieces for fixing microplate on the tray, which makes it challenging to transport a microplate onto the tray or moving a microplate out from the tray.

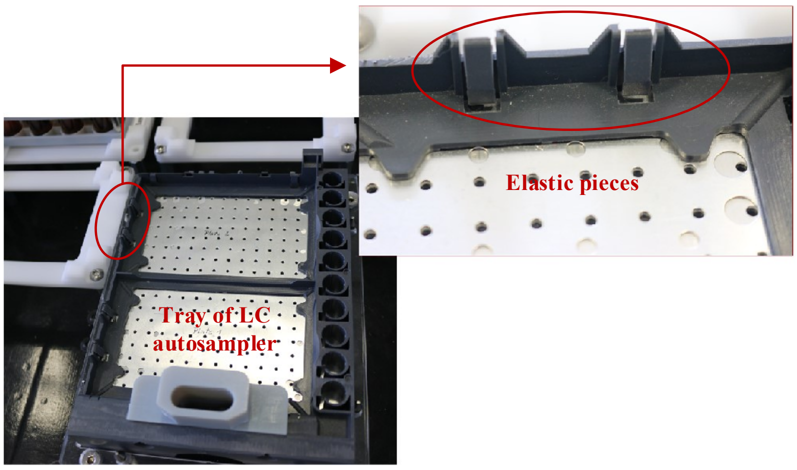


Figure 125. Elastic pieces on the tray of LC auto sampler.

Figure 126 shows the way of transporting a microplate to the tray. First, one side of the microplate will be placed onto the tray. Next, the gripper finger will press the other side into the tray and confirm that the microplate is placed onto the tray completely.

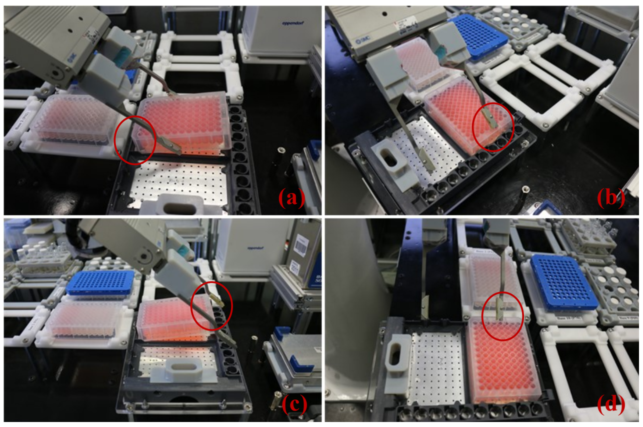


Figure 126. Transferring microplate to the tray.

Figure 127 shows the way of moving a microplate out from the tray. The microplate is clamped by the elastic pieces. First, one side of the microplate will be gripped tightly and then dragged to the other side. In this way, one side will be free. Then that side will be dragged out from the tray. In this case, the microplate can be able to be picked up. Then the robot will grip the middle of the microplate to pick it out from the tray.

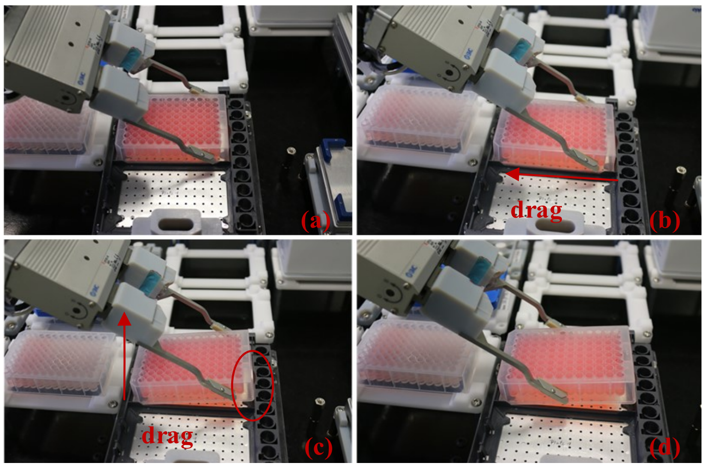


Figure 127. Transferring microplate out from the tray.

11.1.2.3 Removing Lids from Microplates

The thermo shaker has a tray to hold a microplate. When mixing liquids in the microplate, the tray will keep shaking. The shaking time can be long, e.g. 1 hour. In order to prevent liquids evaporating away from microplates during the shaking process, a lid (the blue one) is placed on the microplate. However, when the shaking is stopped, the tray may stop at different positions, which can cause failures of removing the lid from the microplate. In order to make it reliable, when removing the lid from thermo shaker, one finger of the gripper will move close to the lid, touch it and then push it in a small range. Then, the gripper will be turned to grip the lid using two fingers and pick it up.

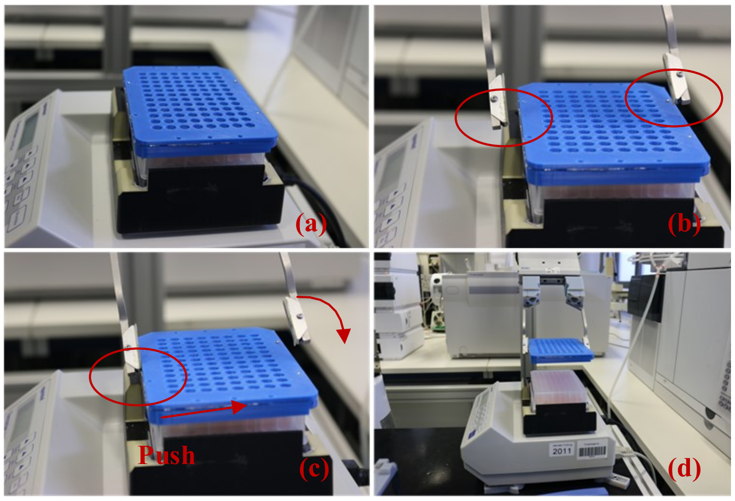


Figure 128. Removing microplate lid from thermo shaker.

11.1.2.4 Picking Microplates from Hotel

The hotel is used to store microplates. In Figure 129 (a), the microplate is not placed in the hotel very well, with some part exceeding the edge. This will cause the failure of transferring the microplate to an ALP on the table by the robot, as the microplate is not gripped at an expected position. An ALP is used to hold and position a microplate. In this case, when picking a microplate, the robot will push the microplate to the hotel in a small range. When pushing, the gripper will grip the microplate loosely, thus the microplate can slip between the gripper fingers. When the microplate is pushed to the end, it will stop moving. If the robot continues to push the microplate, the microplate will slip between the gripper fingers.

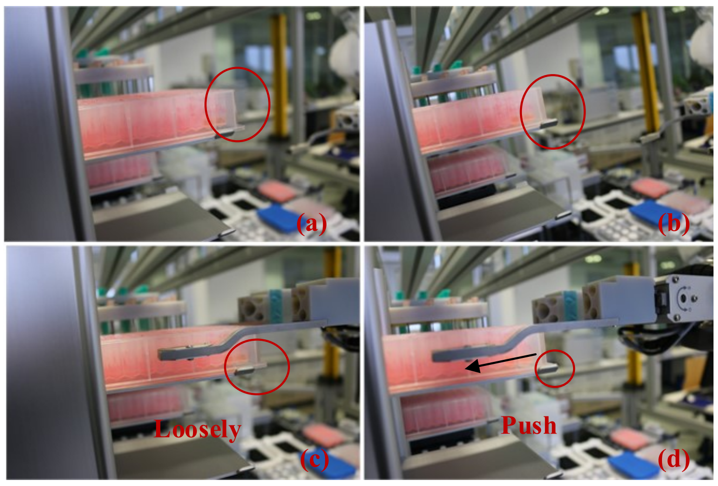


Figure 129. Adjusting the position of microplate.

11.1.2.5 Gripping Syringes

Syringes are used to filter liquids. As shown in Figure 130, the syringes used on the robot platform do not have a uniform diameter (the rod of the syringe is ignored in Figure 130). In this case, if the syringe is gripped at “Position 1”, collisions can happen normally when moving the cannula or filter into a vial, as the syringe will slope in some degrees. In a process of sample preparation, several syringes will be used. For different syringes, the degree and direction of slope are possibly different. It is a correct way to grip a syringe at “Position 2” when moving the cannula (or filter) into a vial. As in this way the syringe will not sloped, collisions between cannulas and vials will be avoided. However, “Position 2” is not a suitable position for gripping when dragging the rod to draw liquid into the syringe. The syringes are plastic products and “Position 2” is soft and deformable. If the gripper grips the syringe firmly at “Position 2”, the movement of the rod will be affected. If not, the syringe can be dragged out from the syringe together with the rod. Conversely, when drawing liquid (the cannula is already in a vial), it will be changed to grip the syringe at “Position 1”, as that part is hard and will not block the moving of rod.

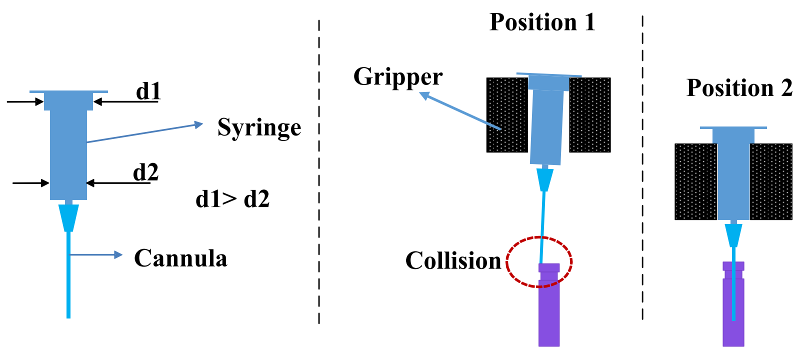


Figure 130. Gripping syringes.

After the handling of the syringes was changed in this way, tests had been made to evaluate the improvements. In the test, 180 different syringes had been used. Using a syringe to filter liquid includes 7 key subtasks, e.g. “Load cannula to syringe”. Each subtask had been testes 180 times. No failure happened in these tests. In the previous tests before changing in this way, the failure rate of handling syringes could be 15%.

11.1.2.6 Screwing Vials

More tests had been made to evaluate the reason of causing failures of screwing cap on vial by the robot, the results indicated that it is related with the vanish thread of the vial. As shown in Figure 131, depending on the relative position between the two vanish threads, the cap can be sloped when screwing it on the vial. Four cases about the relative positions had been tested: (1) around 0 degree, (2) around 90, (3) around 180 degrees and (4) around -90 degrees.

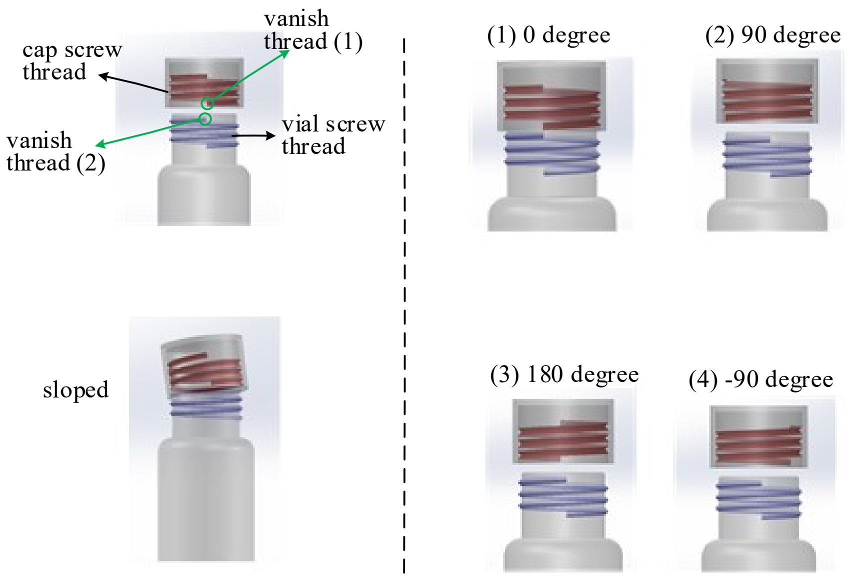


Figure 131. Vanish thread of a vial.

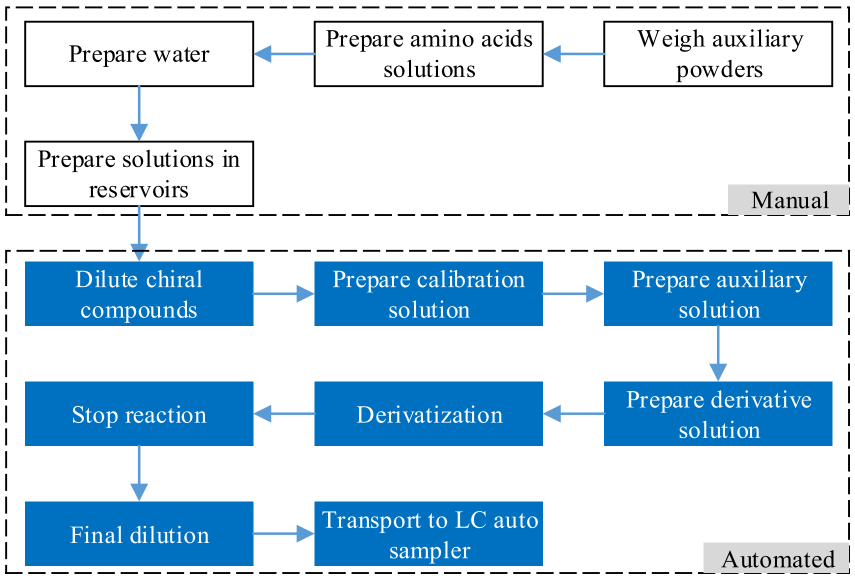
For each of the four cases, the task of closing vials had been tested with the 12 different vials and caps. These vials and caps were used in the other cases. In the case 1 (around 0 degree), the robot failed to close 6 of these 12 vials, with the caps sloped on the vials. In the other cases, the robot closed all of these 12 vials successfully. In order to avoid the failures of screwing caps on vials, one effective solution is: when putting vials and caps on the rack, make sure that the relative position of the vanish threads is around 180 degrees. In this way, when putting a cap on a vial by the robot, it will be the case (3).

11.2 Analysis of Samples

11.2.1 Analysis of Chiral Compounds

11.2.1.1 Automated Procedure

The process of sample preparation about chiral compounds is shown as Figure 132. The part of preparing source powders and source solutions is done manually. The manual part includes: weighing N_α-(2,4-Dinitro-5-fluorophenyl)-L-valinamide and N_α-(5-Fluoro-2,4-dinitrophenyl)-D-leucinamide powders to glass vials, pipetting amino acids solutions to glass vials, pipetting water to glass vials and transferring the other source liquids to reservoirs. In reservoirs, there are eight containers: the first container is clean which will be used to contain prepared auxiliary solution later, the second to the fifth containers are to contain methanol, the sixth container is to contain formic acid solution, the seventh container is to contain NH_4HCO_3 solution, and the last container is to contain acetone. After the source powders and source liquids are provides, the process of using these source materials to prepare sample solutions are automated.


Prepare water
Prepare amino acids solutions
Weigh auxiliary powders
Prepare solutions in reservoirs
Manual
Dilute chiral compounds
Prepare calibration solution
Prepare auxiliary solution
Stop reaction
Derivatization
Prepare derivative solution
Final dilution
Transport to LC auto sampler
Automated
Figure 132. Sample preparation process about chiral compounds.

The automated process includes (more details about the flowchart can be seen in Figure 133):

a) **Diluting chiral compounds.** This process is to dilute the amino acids (proline). L-proline solution and D-proline solution are contained in two glass vials, and both of them have a concentration of 50 mmol/L. In order to dilute the amino acids to a concentration of 1 mmol/L, 5,880 µL water and 120 µL amino acid will be pipetted separately to two glass vials. Finally, one glass vial will contain 120 µL L-proline solution and 5,880 µL water, while the other one contains 120 µL D-proline solution and 5,880 µL water.

b) **Preparing calibration solution.** The diluted L-proline (L) solution and D-proline (D) solution will be distributed to five empty glass vials with different volume ratios (L:D): 2,000 µL:0 µL (100ee%), 1,500 µL:500 µL (50ee%), 1,000 µL:1,000 µL (0ee%), 500 µL:1,500 µL (-50ee%) and 0 µL:2,000 µL (-100ee%). At the end, each vial contains 2,000µL solution in total. The five solutions are used to produce the calibration curve for calculating the enantiomeric excess of the samples.

c) Preparing auxiliary solution. Auxiliary solution is used to react with the amino acid solutions to produce derivatives which are used to determine the chiral compounds. For generating the derivatives of L-proline and D-proline, N_{α}-(2,4-Dinitro-5-fluorophenyl)-L-valinamide (called L-FDVA for short) and N_{α}-(5-Fluoro-2,4-dinitrophenyl)-D-leucinamide (called D-FDLA for short) are used. Before preparing auxiliary solution, 7.50 mg L-FDVA powder and 7.85 mg D-FDLA powder are weighed and distributed in two glass vials separately (this step is done manually). Beside this step of weighing L-FDVA and D-FDLA powders, the steps of dissolving the powders and distributing the resulting solutions to a microplate are done by the robot automatically. Acetone is used to dissolve the powders. First, 2.5 mL acetone is pipetted to the vial containing the L-FDVA; then the resulting L-FDVA solution is pipetted to the vial containing D-FDLA. This procedure of pipetting acetone will be repeated for four times in total. At the end, the final 10 mL auxiliary solution mixing with L-FDVA and D-FDLA will be pipetted to a reservoir, in order to make it feasible to use an 8-channel pipet to distribute the auxiliary solution to a microplate.

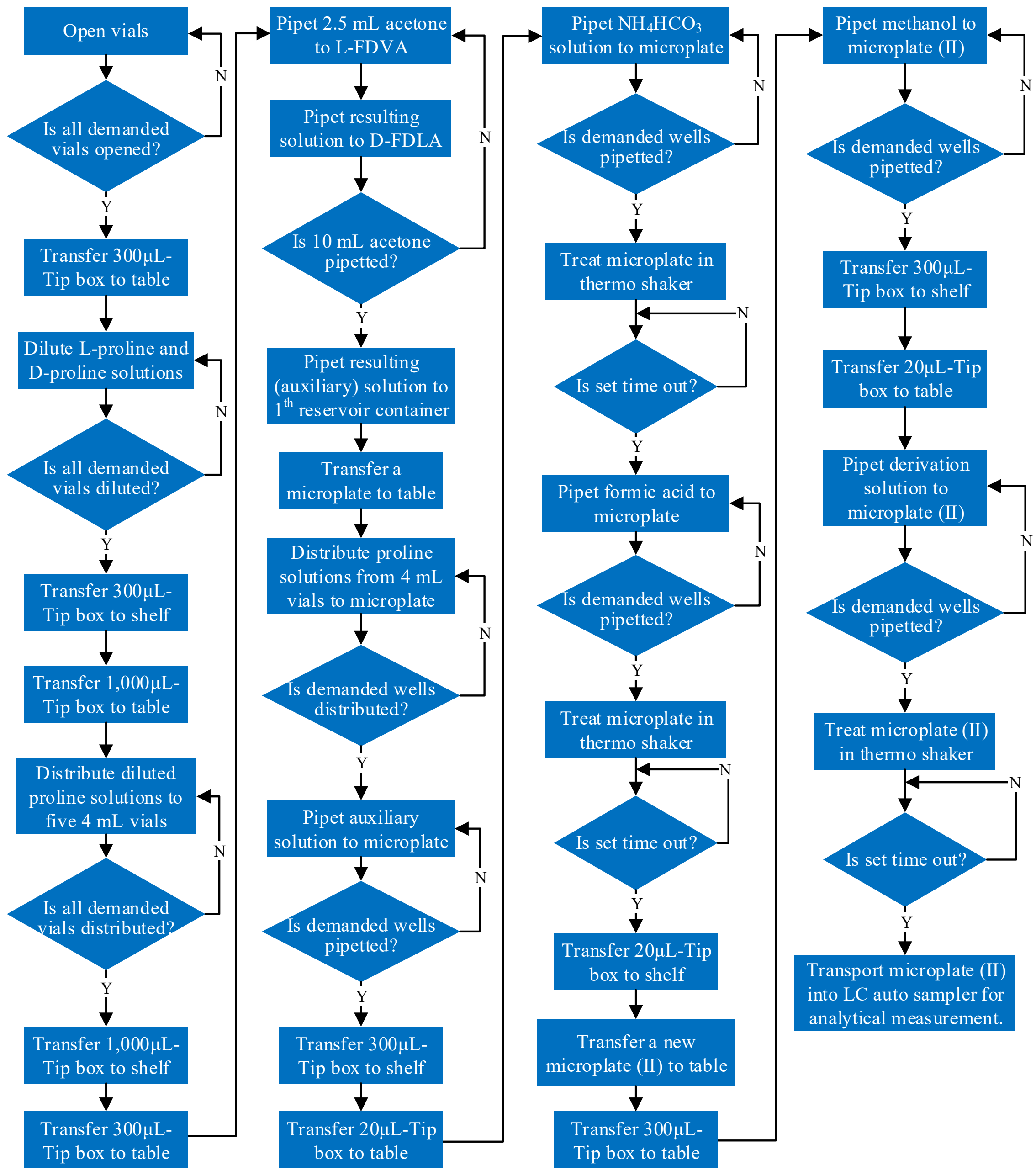
d) Preparing derivation solution. After the solutions of the chiral compounds and auxiliary solutions are prepared, they will be distributed to a microplate for reactions to produce the derivatives. First, 50 µL amino acid solution is pipetted to each well of the microplate. The first 15 wells, from “A1” to “G2”, are used for calibration. The “A1”, “B1” and “C1” wells are pipetted with the 100 ee% calibration solutions; the “D1”, “E1” and “F1” wells are pipetted with the 50 ee% calibration solution; the “G1”, “H1” and “A2” wells are pipetted with the 0 ee% calibration solution; the “B2”, “C2” and “D2” wells are pipetted with the -50 ee% calibration solution; the “E2”, “F2” and “G2” wells are pipetted with the -100 ee% calibration solution. The “H2” well is blank, pipetted with water replacing the amino acid solution. The remaining wells will be pipetted with the solution of chiral compounds, which are treated as “unknown samples”. Second, 100 µL auxiliary solution will be pipetted into each well of the microplate. At last, 20 µL NH_4HCO_3 solution is pipetted to each well of the microplate, which is used to promote the reaction between amino acid solution and auxiliary solution.

e) Derivatization. After the solutions are pipetted into the microplate, it will be transferred into a thermo shaker. The derivatization is conducted in the thermo shaker, with room temperature, 750 rpm and a 1 hour shaking time.

f) Quenching reaction. After procedure of derivatization, the microplate will be transferred out from the thermo shaker and put to an ALP on the table. And 10 µL formic acid solution will be pipetted into each well of the microplate, which is used to stop the reaction between the amino acid solution and the auxiliary solution. In order to stop the reaction effectively, the microplate will be transferred again to the thermo shaker for 3 minutes shaking.

g) Final dilution. Finally, the sample solutions in the microplate (I) containing the derivation solutions will be diluted to a suitable concentration for analysis by the LC-MS instrument. For doing this, another empty microplate (II) will be used. Methanol is used as the solvent to dilute the sample solution. First, 490 µL methanol is pipetted into each well of microplate II. Then the 10 µL sample solution is pipetted from the each well of microplate (I) into the corresponding well of microplate (II). The volume ratio between methanol and sample solution can be changed for obtaining final solutions with different concentrations. Then the microplate (II) is transferred into the thermo shaker again for 6 minutes evenly shaking.

h) **Transport.** Finally, the microplate with the diluted sample solutions will be placed on a tray, and then the tray will be transported into the LC auto sampler for analysis.


Open vials
Is all demanded vials opened?
Y
N
Transfer 300µL-Tip box to table
Dilute L-proline and D-proline solutions
Is all demanded vials diluted?
Transfer 300µL-Tip box to shelf
Transfer 1,000µL-Tip box to table
Distribute diluted proline solutions to five 4 mL vials
Is all demanded vials distributed?
Transfer 1,000µL-Tip box to shelf
Transfer 300µL-Tip box to table
Pipet 2.5 mL acetone to L-FDVA
Pipet resulting solution to D-FDLA
Is 10 mL acetone pipetted?
Pipet resulting (auxiliary) solution to 1th reservoir container
Transfer a microplate to table
Distribute proline solutions from 4 mL vials to microplate
Is demanded wells distributed?
Pipet auxiliary solution to microplate
Is demanded wells pipetted?
Transfer 300µL-Tip box to shelf
Transfer 20µL-Tip box to table
Pipet NH₄HCO₃ solution to microplate
Is demanded wells pipetted?
Treat microplate in thermo shaker
Is set time out?
Pipet formic acid to microplate
Is demanded wells pipetted?
Treat microplate in thermo shaker
Is set time out?
Transfer 20µL-Tip box to shelf
Transfer a new microplate (II) to table
Transfer 300µL-Tip box to table
Pipet methanol to microplate (II)
Is demanded wells pipetted?
Transfer 300µL-Tip box to shelf
Transfer 20µL-Tip box to table
Pipet derivation solution to microplate (II)
Is demanded wells pipetted?
Treat microplate (II) in thermo shaker
Is set time out?
Transport microplate (II) into LC auto sampler for analytical measurement.
Figure 133. Automated sample preparation process about chiral compounds by the dual-arm robot.

The automated part of sample preparation by the dual-arm robot is quite similar with the manual one, besides several different points. Because there are only two fingers on a robot's hand, the robot uses the tools in a different way as that by a human user. E.g., while a human can use one hand to handle a pipette to transfer liquid, the robot needs to use two hands to do this (one hand holds the pipette and the other one presses the button). Second, the part of covering a lid on a microplate is changed. A lid is used to prevent liquid evaporation, when shaking the microplate in the thermo shaker. In the manual process, a soft lid will be pressed onto the microplate by a specific mechanical mechanism. In the automated process, a printed lid is used to replace the soft lid, which can be easily put onto or take away from a microplate. Besides, some racks have been made to assist the automation. An adapter has been fixed on the pipette, in order to make the robot able to hold it tightly. Racks have been printed for holding and positioning vials, glass pipettes, and tips.

11.2.1.2 Analysis Results

The robot prepared the samples following the process shown in Figure 133. Then the final solutions had then been analyzed by the LC-MS instrument, which includes a G1379B vacuum degasser, a G1312B binary pump, a G1367C high-performance automated liquid sampler, and a G1969A time-of-flight mass spectrometer (TOF-MS) with electrospray ionization (ESI). Data acquisition and processing were performed using MassHunter Data Acquisition and MassHunter Qualification software (Agilent Technologies). Figure 134 shows the m/z values of derivatives of proline in the mass spectra. From the value, it can be known that the target derivatives were generated. Thus, the robot had prepared the samples successfully.

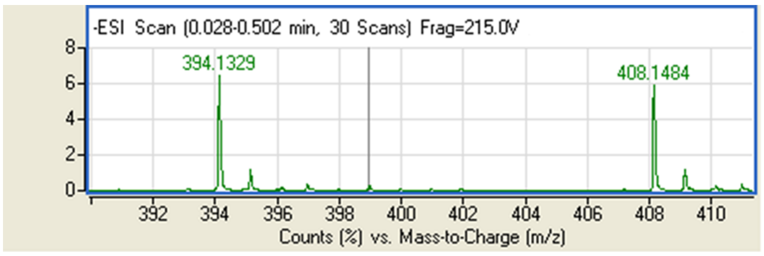


Figure 134. Mass spectrum of the [M-H]– ions of the derivatives of proline. The sample was prepared using the CSDA10F dual-arm robot.

The analysis results had been compared with that of samples which were prepared manfully using the same labware and following the same preparation process. Figure 135 gives the calibration lines from the sample analysis data: (a) was from the samples prepared by the robot and (b) was from the samples prepared by the skilled laboratory assistant. At each measurement points, three samples had been prepared. The average of the measured data from the three samples was used to generate the calibration line.

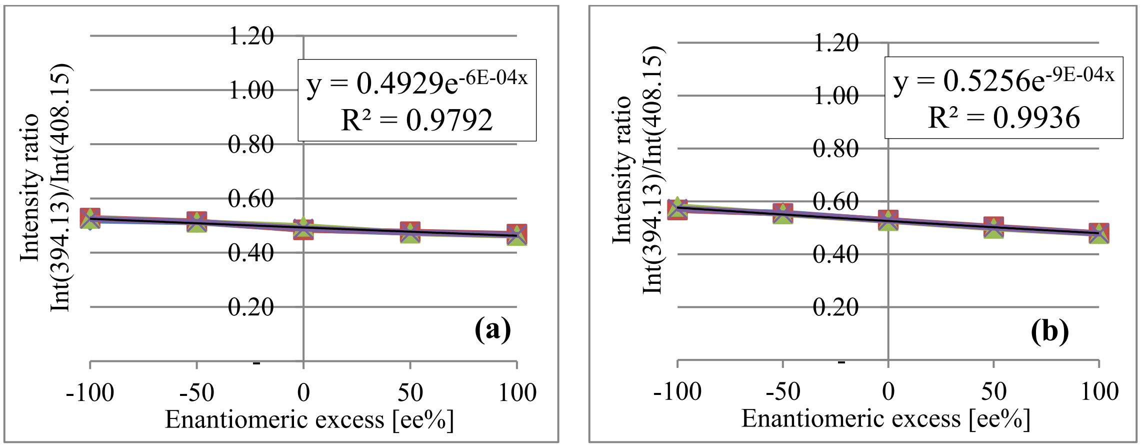


Figure 135. The calibration line created from the analysis data of the samples: prepared by the robot (a), prepared by skilled laboratory assistant (b).

Table 9 gives the measured data at each measure point. From Table 9 and Figure 135, it can the seen the samples prepared by the robot were similar with those by manual.

Table 9. Measure data of calibration samples.

	Samples by robot		**Samples by manual**	
ee%	**Int(394.13)/Int(408.15)**[7]	**Average**	**Int(394.13)/Int(408.15)**	**Average**
100	0.465	0.466	0.478	0.478
100	0.469		0.479	
100	0.463		0.477	
50	0.474	0.474	0.502	0.501
50	0.477		0.504	
50	0.472		0.498	
0	0.486	0.489	0.528	0.528
0	0.485		0.529	
0	0.497		0.526	
-50	0.512	0.513	0.557	0.554
-50	0.515		0.554	
-50	0.512		0.552	
-100	0.519	0.524	0.567	0.572
-100	0.528		0.568	
-100	0.526		0.581	

The repeatability of the robot platform in preparing samples had been evaluated, by repeatedly preparing 16 samples at each enantiomeric excess level. These samples were prepared with the known solution, with L-proline and D-Proline solutions at certain concentrations. Table 10 shows the analyzed results. The average of the intensity ratios of derivatives in 16 samples at -50 ee% was 0.510, with the sample standard deviation being 0.0046 and the coefficient of variation being 0.9%. The analysis results about the other samples can also be seen in Table 10.

Table 10. Sample repeatability.

ee%	**Average**	**Sample standard deviation**	**Coefficient of variation**
100	0.455	0.0056	1.2%
50	0.473	0.0050	1.1%
0	0.489	0.0059	1.2%
-50	0.510	0.0046	0.9%
-100	0.523	0.0066	1.3%

Table 11 summarizes the processing times by the robot platform for each step. The time used in the manual way is also given. The total time used for the robot is around 81 minutes, while it costed around 31 minutes by manual. Figure 136 shows the time percentage costed in handling different lab ware by the robot in an automated process. The biggest part was about using pipettes to transport liquid.

Table 11. Comparison of the time used for preparing samples.

Step	**Time by robot**	**Time by manual**
Diluting chiral compounds	12 min 59 s	1 min 20 s
Preparing calibration solutions	5 min 57 s	3 min 10 s

[7] The intensity of ions of derivative with 394.13 m/z divided by the intensity of ions of derivative with 408.13 m/z.

Preparing auxiliary solution	15 min 5 s	2 min 40 s
Preparing derivation solutions[8]	24 min 1 s	9 min 30 s
Final Dilution[9]	23 min 3 s	14 min 0 s
Total	81 min 5 s	30 min 40 s

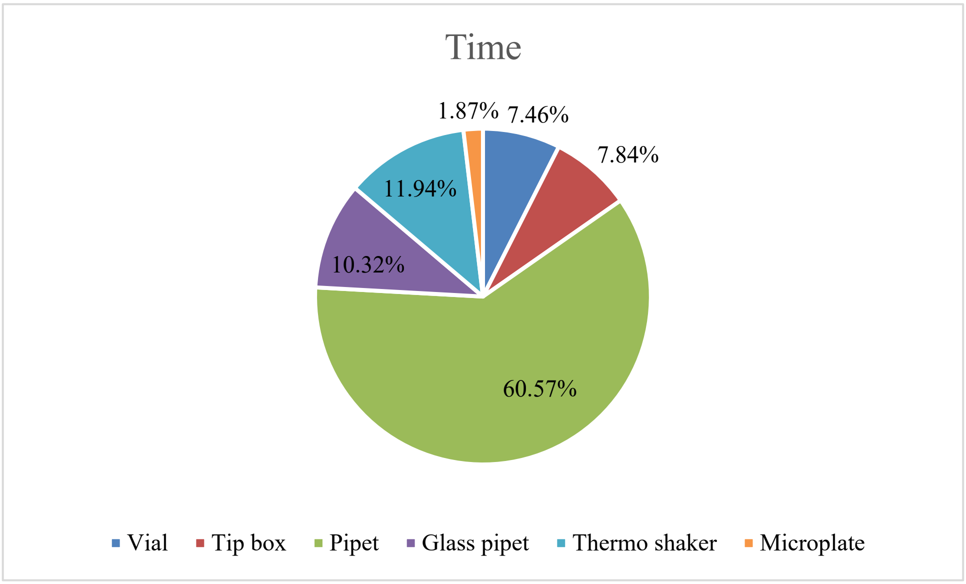


Figure 136. Time costed at handling different lab ware.

In Table 11, it can be seen the robot platform takes more time in each step. In the step of diluting chiral compounds, it took the robot platform 12 min 59 s, which was much longer than that in the manual way (1 min 20 s). There are three potential ways to reduce the operating time. As a gripper has only two fingers, it is not as flexible as a human hand. In this case, the robot has to turn its gripper several times to open a vial (while a human only need to slide his/her fingers), has to cooperate its two grippers to pipet liquids (while a human can do it by only one hand), has to put a tip box on a re-gripping ALP first and then put it on the table (while a human can put it directly on the table) and so on. Firstly, one potential way is to replace the grippers by flexible robotic hands, which should be as flexible as human hands and also include a flexible wrist at the end of the robotic hand. When pipetting liquids, the robot has to pick a pipet from the shelf behind the robot and put it back after finishing pipetting (a human pick a pipet from the front table, which has a much shorter moving distance). Another solution is possibly optimizing the layout of the platform to reduce the moving distance. For handling each lab ware, several motion elements have been created. Thirdly, it is also possible to reduce the operating time by improving the motion elements, e.g. optimizing the motion steps of a motion elements to make it more effective. The step of preparing derivation solutions costed about 24 minutes, which took much longer time than the other steps. This is because this step included a lot of pipetting (132 times of pipetting) and the transportations of the microplate between table and the thermo shaker for the derivatization. The step of final dilution costed about 23 minutes, which included the time used for quenching reaction and transporting microplates between the thermo shaker and the table. In summary, there are several reasons limiting the robot platform speed about handling labware:

[8] It includes the step of derivatization. The shaking time in thermo shaker is not included.
[9] It includes the step of quenching reactions. The shaking time in thermo shaker is not included.

1. Firstly, the robot cannot pick an object (e.g. a vial) from a rack at a high speed (e.g. limited to 30 mm/s). A rack is placed on an ALP and is moveable. There is a friction generated between the object and the labware, when picking the object from the rack. If the speed is high, the fiction will be big enough causing the rack picked together with the object.
2. Secondly, when pipetting liquid, the tip/tips should not leave from the liquid at a high speed (e.g. limited to 40 mm/s), otherwise there will be drops of liquid adhered to the outer surface of tip/tips.
3. Thirdly, the robot has only two fingers for each hand, which are not flexible and reliable as human fingers. For example, for the robot to open a vial, it has mainly to do five things: 1) pick the vial from the rack using the right hand; 2) put it to the left hand; 3) turn the right hand to unscrew the cap; 4) put the cap to the rack; 5) and put the vial back to the rack. The right hand can only turn no more than 360 degrees. When it has to unscrew a cap for more than 360 degrees, the right hand will repeat the clockwise turns and counterclockwise turns. In this way, it takes around 40 seconds for the robot to open a vial. Benefiting to our hands with five fingers for each one, it can take no more than 4 seconds to open a vial.
4. Fourthly, the robot works in a narrow environment, with instruments and the shelf around it. In this case, the robot has to move a longer path to reach the destination position, in order to avoid collisions. Moreover, some parts of the motions are not smooth, with some frictional noises heard when the moving speed was high. As a high jerk may expedite the damage to an actuator, the speed of such motions is limited to 50% of the total speed.

11.2.2 Analysis of Cholesterol

11.2.2.1 Automated Procedure

For preparing sample solutions and calibration solutions, some raw materials needed to be prepared previously, including sample powder obtained from biliary endoprosthesis, cholesterol solution with specified concentration (e.g. 10 mg/L) which is used to make calibrations, 20 mg/L 5α-cholestane solution used as internal standard and hexane for extracting cholesterol from the sample and diluting sample solutions. 1 mg sample powder is weighed manually and filled to the GC vials (of which the volume is around 2 mL) previously. Besides these parts, the rest parts will be done by the dual-arm robot automatically. The process of preparing cholesterol samples is shown in Figure 137.

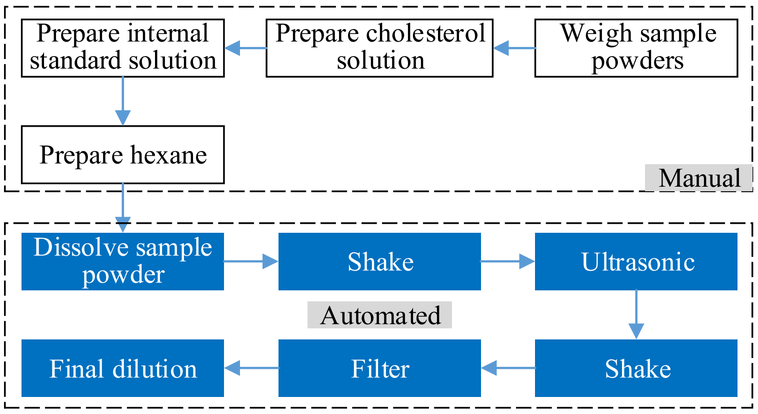


Figure 137. Sample preparation process about cholesterol.

The automated sample preparation process mainly includes six parts (more information can be seen in Figure 139):

a) **Dissolving sample powder.** In this step, 1,000 µL hexane will be added to each of the GC vials which contains the sample powder. Two types of pipettes are used for pipetting the 1,000 µL hexane, using 1mL pipette to transfer 850 µL hexane and 200 µL pipette to transfer 150 µL. The volume of a pipette is set previously. Currently, the dual-arm robot platform is not able to adjust the volume of pipettes. The hexane is used to extract the cholesterol from the sample powder.

b) **Mixing.** A small shaker is used to promote the mixing of the components in the solutions. After hexane is pipetted to the GC vials, they will be transported to the rack on the small shaker and shaken for 1 minute.

c) **Ultrasonic.** The extraction of cholesterol is promoted in the ultrasonic bath. GC vials will be transported to the rack in the ultrasonic bath. After processed in the ultrasonic bath for 5 minutes, the GC vials will be transported again to the small shaker and shaken for 15 minutes.

d) **Filer.** After the ultrasonic treatment and the shakings, the resulting solution will be filtered using classical syringes. The filtered solution will be stored in new GC vials. First, the robot will pick a syringe and load a cannula to it. After the solution is drawn into the syringe, the cannula will be released and a filter will be loaded to the syringe. Then the solution will be pressed into a clean GC vial. For the next vial, a new syringe will be used.

e) **Final sample solution.** Then, the filtered solution will be diluted by hexane. To a new GC vial, 100 µL filtered solution, 50 µL internal standard solution and 850 µL hexane will be pipetted, to prepare a final sample.

f) **Calibration solution.** Calibration solutions will also be prepared by the robot with the concentration of cholesterol ranging from 500 µg/L to 2500 µg/L, in order to calculate the concentration of cholesterol in final samples. But it is not necessary to prepare calibration solutions for every measurement.

Finally, the prepared solutions will be stored in 12 GC vials. Figure 138 gives a plan about the compounds in the 12 GC vials. The first five vials contain the calibration solutions. The remaining seven vials are the sample solutions.

(10) 100mg/L sample -- 50µL 5α-cholestane -- 100µL sample -- 850µL hexane	(7) 100mg/L sample -- 50µL 5α-cholestane -- 100µL sample -- 850µL hexane	(4) 2000µg/L Calibration -- 200µL cholesterol -- 50µL 5α-cholestane -- 750µL hexane	(1) 500µg/L Calibration -- 50µL cholesterol -- 50µL 5α-cholestane -- 900µL hexane
(11) 100mg/L sample -- 50µL 5α-cholestane -- 100µL sample -- 850µL hexane	(8) 100mg/L sample -- 50µL 5α-cholestane -- 100µL sample -- 850µL hexane	(5) 2500µg/L Calibration -- 250µL cholesterol -- 50µL 5α-cholestane -- 700µL hexane	(2) 1000µg/L Calibration -- 100µL cholesterol -- 50µL 5α-cholestane -- 850µL hexane
(12) 100mg/L sample -- 50µL 5α-cholestane -- 100µL sample -- 850µL hexane	(9) 100mg/L sample -- 50µL 5α-cholestane -- 100µL sample -- 850µL hexane	(6) 100mg/L sample -- 50µL 5α-cholestane -- 100µL sample -- 850µL hexane	(3) 1500µg/L Calibration -- 150µL cholesterol -- 50µL 5α-cholestane -- 800µL hexane

Figure 138. Final solutions in the 12 GC vials.

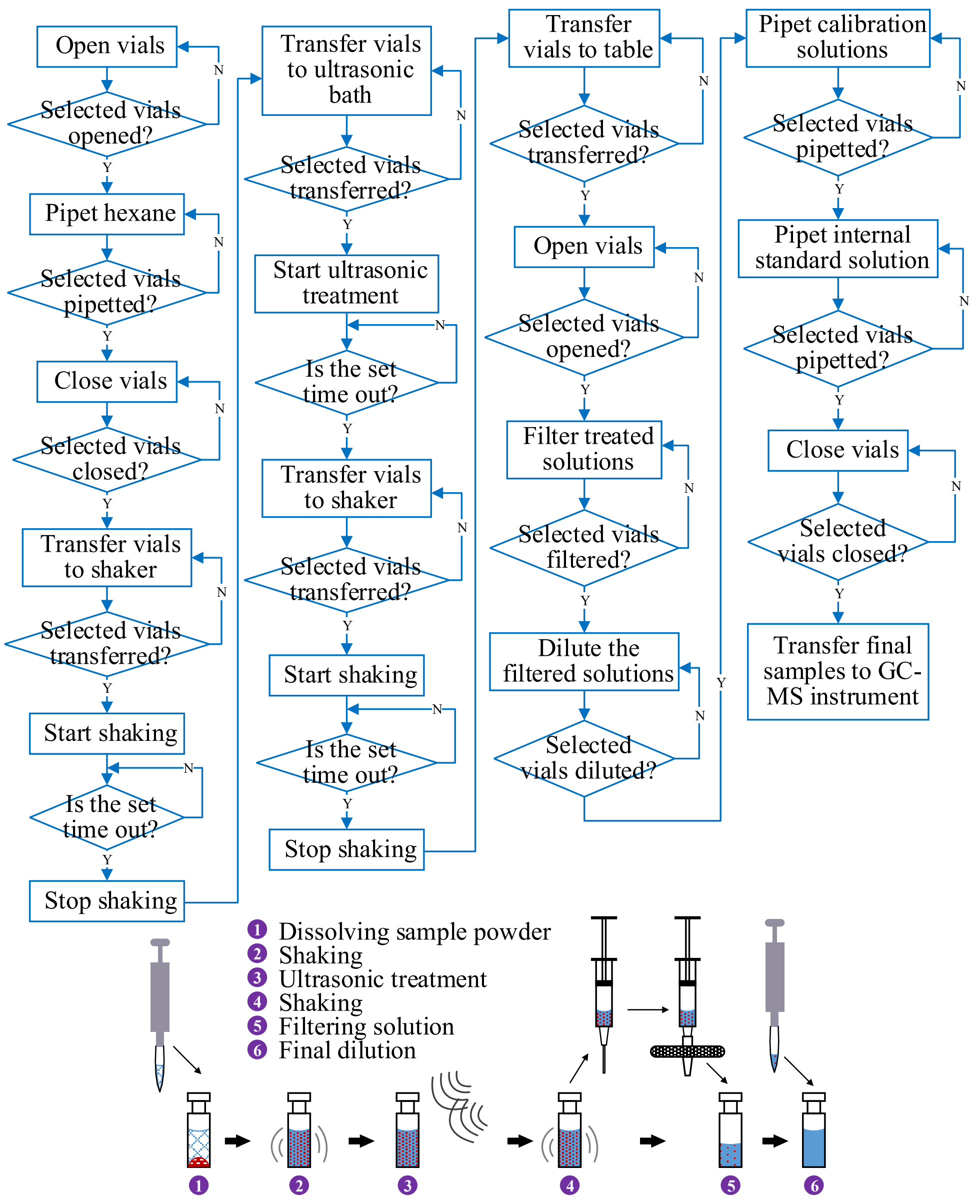


Figure 139. Automated process of preparing cholesterol samples by the dual-arm robot.

11.2.2.2 Analysis Results

The prepared samples by the dual-arm robot were analyzed by the Agilent GC-MS instrument which includes a 7890A gas chromatograph and a 7010-triple quadrupole mass analyzer. Data acquisition and data analyzing are performed using the following Agilent software: MassHunter GC-MS Acquisition B.07.02., MassHunter Qualification B.07.00 and MassHunter Quantification for QQQ B.07.00. Figure 140 shows the chromatogram of a cholesterol sample prepared by the robot. The sample was prepared with the concentration of 5α-cholestane solution being 0.25 mg/L and the concentration of cholesterol solution being 4 mg/L.

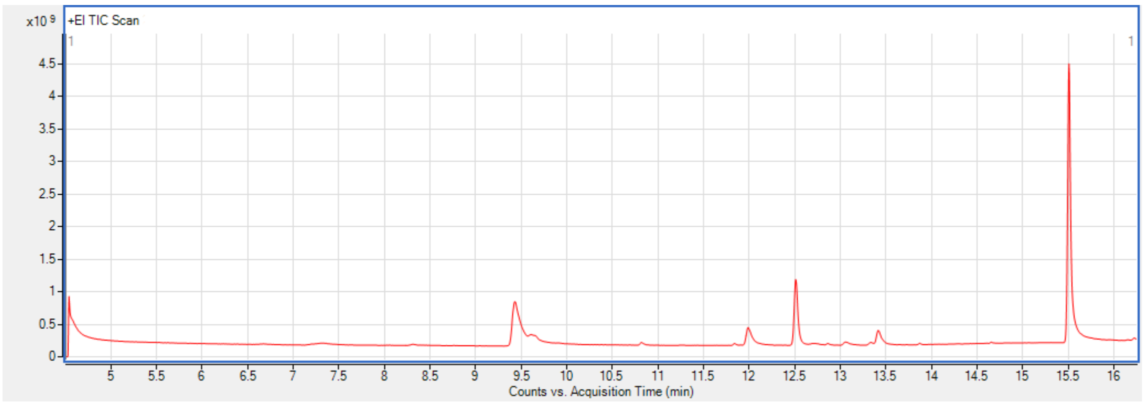


Figure 140. Chromatogram of a cholesterol sample prepared by the dual-arm robot, with 4 mg/L cholesterol and 0.25 mg/L 5α-cholestane.

The responses of the calibration solutions are shown in Figure 141. The calibration solutions were prepared by the dual-arm robot, with the concentrations of cholesterol ranging from 0.5 mg/L to 2.5 mg/L. In the calibration solutions, 5α-cholestane is also used as internal standard.

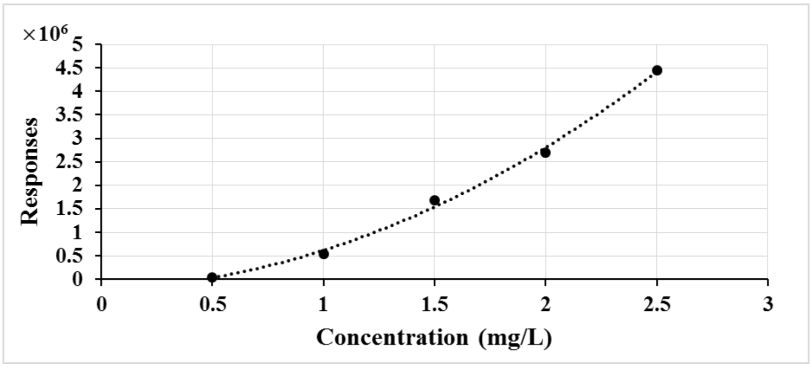


Figure 141. Responses of calibration solutions prepared by the dual-arm robot.

In Table 12, nine groups of samples had been prepared by the dual-arm robot and then analyzed by the GC-MS instrument, with each group including 12 samples. As the concentration of cholesterol in a sample was 4,000 μg/L when prepared, the average of calculated concentration of the samples in a group should also approximate to 4,000 μg/L. However, it can be seen from Table 12 that the average of calculated concentration ranged from 3,227.3μg/L to 4,761.5μg/L. Further study related with the sample preparation process has to be carried out in order to make improvements to prepare more uniform samples. The coefficient of variation of samples in a group was below to 10%. While it tasks around 24 minutes to prepare 12 samples, robot used about 80 minutes to prepare the same amount samples.

Table 12. Sample repeatability.

No.	Date	Ave. Con.[10]	Sample standard deviation	Coefficient of variation
1	Aug. 18, 2016	4,761.5μg/L	383.8μg/L	8.1 %
2	Aug. 23, 2016	4,298.7μg/L	365.0μg/L	8.5 %
3	Aug. 30, 2016	4,352.3μg/L	306.8μg/L	7.0 %
4	Aug. 30, 2016	3,616.1μg/L	354.5μg/L	9.8 %
5	Aug. 31, 2016	4,214.6μg/L	205.4μg/L	4.9 %
6	Sep. 01, 2016	4,469.0μg/L	329.8μg/L	7.4 %
7	Sep. 02, 2016	3,227.3μg/L	196.7μg/L	6.1 %
8	Sep. 05, 2016	4,787.4μg/L	359.8μg/L	7.5 %
9	Sep. 06, 2016	4,107.8μg/L	346.5μg/L	8.4 %

[10] Average of calculated concentration of 12 samples.

Chapter 12 Conclusions and Outlook

12.1 Conclusions

In this dissertation, a dual-arm robot platform has been presented for automating the sample preparation and analysis in life science fields. In this platform a series of technical and scientific issues has been solved as following:

(1) Definition of motion element. As a lot of robot motions are needed for handling different experimental tools, e.g. pipettes, syringes or vials, and also for avoiding collisions in the narrow space, motions are separated into many small units. Each unit is called as a motion element. Different motion elements can be combined through position nodes. There are two types of position nodes: front node, which is the position of the first step of a motion element, and end node, which is the position of last step of a motion element. Motion elements sharing the same position node can be combined together. As each motion element is relatively independent with each other, they can be modified or deleted separately. And new built motion elements can be added by "connecting" to the existing position nodes. Motion elements are separated to two types: path motion element with fixed moving path and relative motion element with changeable moving path. By setting parameters, the relative motion element can be adapted in certain range.

(2) Call-file for integrating motion elements. One single motion element is useless. For certain tasks, it needs to integrate several motion elements together. And for the relative motion elements, it needs to set parameters to them. The motion elements can be integrated by a call-file, which is a special robot job and can run on the robot controller. There are three ways to create such call-files: create them online by using the teach pendant, create them offline on a personal computer similarly as creating a text file or generate by a program automatically.

(3) Database for recording motion elements. A database, named as M-database, is defined for recording the information of motion elements and is to support other programs for dealing with the motion elements. The M-database is generated by analyzing the contents of robot job files. Key information will be abstracted from the file contents and unavailable job files will be filed. In the database, every motion element has nigh properties, including its names, its position nodes and the list of related motion elements. The M-database can be renewed by a program. Thus, when motion elements are modified, deleted or adding new elements, the M-database can be updated in time.

(4) Handling experimental tools. Many motion elements had been built. By using them, the robot can be driven to handle a series of experimental tools, including pipettes, vials, microplates, tip boxes, glass pipettes, syringes, thermo shaker, ultrasonic machine, LC auto sampler and GC auto sampler.

(5) Integrating LC-MS instrument and GC-MS instrument. The LC-MS instrument and GC-MS instrument have been integrated into this platform. In the hardware side, the robot will transport samples to the instruments for analysis. The challenge of integrating LC-MS instrument is that

the “room” of the auto sampler is closed, protecting by a door. For transferring samples to the “room” of auto sampler, the robot needs to open the door first and then close the door. It is similar for transferring analyzed samples out from the “room” of auto sampler. In the software side, they are integrated with SAMI software and thus SAMI software can control LC-MS instrument and GC-MS instrument to start a measurement automatically. The challenge is that there are not ample APIs to integrating the software of LC-MS and GC-MS with SAMI software. Two program interfaces, wrote in C#, have been built separately for integrating the software of LC-MS and GC-MS with SAMI software. And the GUIs of the software of LC-MS and GC-MS have been used for realizing the integrations. The program interface can simulate mouse inputs and keyboard inputs to the GUIs and also can capture the text and color information showing on the GUIs.

(6) Integrating the robot system with SAMI software. Finally, the robot system has been integrated with the SAMI system, which provides a friendly graphical interface. By integrating with SAMI system, the user having no knowledge about programming for the robot can also operate the robot to prepare sample. A program interface (called R-interface) has been built for integrating the robot system and the SAMI system. On one side, R-interface will interactive with SAMI by receiving and sending XML data. SAMI will send commands to R-interface. And R-interface will execute them and feedback the results to SAMI. On the other side, R-interface will create call-files according SAMI commands and download them to the robot controller. These call-files are used to integrate the motion elements. R-interface will also generate the M-database to record the information of motion elements. When executing the SAMI commands, the related motion elements will be indexed from the M-database. Then, together with parameters, the names of these related motion elements will be integrated into a call-file.

(7) Extendibility. The dual-arm robot platform is flexible to be extended. This is discussed at section 12.2.

(8) Automating the processes of sample preparation and analysis. The dual-arm robot has automated the process of sample preparation and analysis about chiral compounds and the process of sample preparation and analysis about cholesterol. The samples of chiral compounds prepared by the robot had been analyzed by the LC-MS instrument and the samples of cholesterol prepared by the robot had been analyzed by the GC-MS instrument. The analysis results were similar with the results of the samples prepared by manual.

12.2 Extendibility of the Robot Platform

As shown in Figure 25, there are several ALPs mounted on the deck for placing lab ware. A lab ware fits with the ALP can be easily placed or moved away from an ALP. Thus, it is flexible to layout new lab ware on the deck for new applications. New instruments can also be mounted on the deck. As shown in Figure 142, an SPE (Solid-Phase Extraction) instrument was mounted on the deck afterwards, which is used to separate compounds in a liquid mixture according their physical and chemical properties. This is the way for adding new lab ware to the platform. For the robot to handle the new lab ware, related robot motions have to be created, such as motions for transferring lab ware to the SPE instrument, motions for pressing the button on the SPE instrument or motions for pipetting liquid to the SPE instruments. After separating these motions to motion elements, they can be connected with the previous motion elements. Thus, the newly built motions elements and the previous motion elements can be used together to perform tasks.

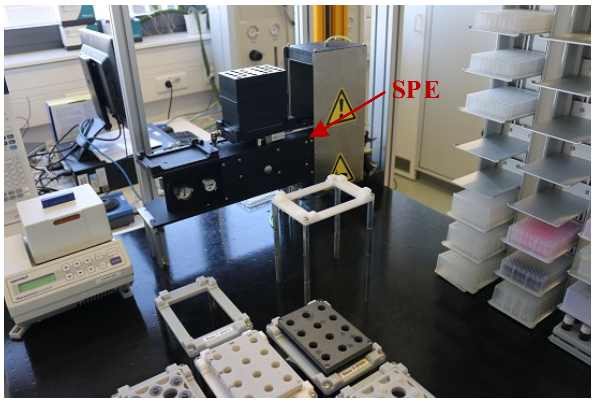


Figure 142. Integrating SPE to the platform.

Besides, a shuttle has also been mounted to the platform later, which is used as the interface for interacting with mobile robots. The shuttle is shown in Figure 143, which can move following the guide. After a mobile robot putting a lab ware on the shuttle, it will be moved to the dual-arm robot. Then the robot can pick it from the shuttle and put it on an ALP. Similarly, the dual-arm robot can also put a labware on the shuttle to transport it to the mobile robot at the other end of the guide. Robot motion elements for picking microplates from the shuttle and putting microplates to the shuttle has been created and connected with previous motion elements.

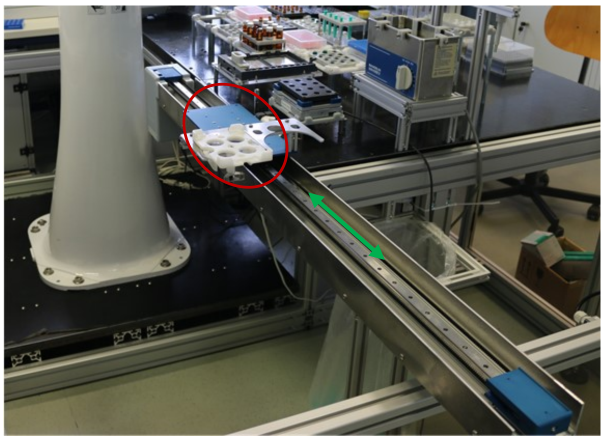
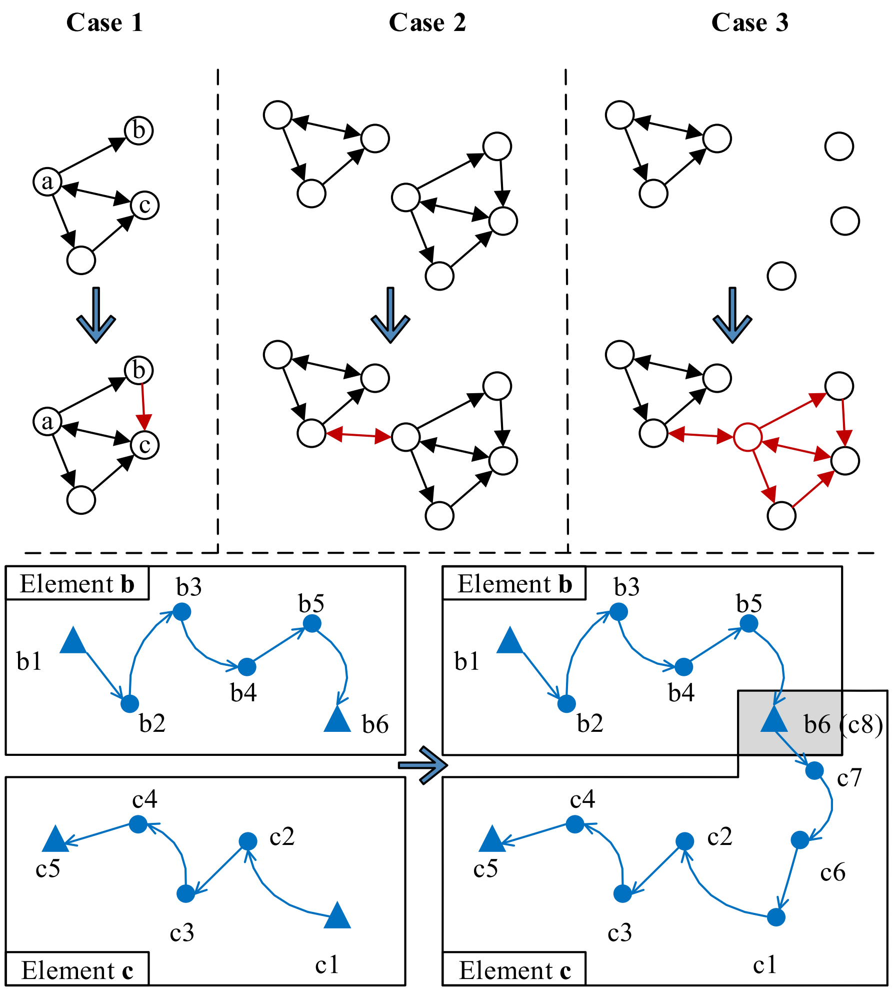


Figure 143. Interface for interacting with mobile robots.

There are three bad cases when adding new motion elements, as shown in Figure 144. In case 1, motion element (a) is connected to motion element (b) in one-way. After motion element (b) is used, there will be no way to go. The solution is to connect motion element (b) to the others, e.g. motion element (c). In case 2, the newly created group of motion elements is isolated with the previous group. The solution is connecting at least two motion elements of the two groups in two-way. In case 3, the newly created motion elements are isolated with each other. More work has to be done for connecting them to the previous motion elements, even some new motion elements has to be created.



Connect element **b** and **c**

Figure 144. Problems and solutions of extending new motion elements.

For connecting two motion elements, some path has to be created, by extending the path of a motion element or creating a new motion element to realize the extend path. In the example shown in Figure 144, the path of motion element (c) is extended, in order to connect with motion element (b). Assuming motion element (b) starts from position (b1), passes position (b2), (b3), (b4), (b5) and ends at position (b6). Position (b1) is the front node of motion element (b), while position (b6) is its end node. Position (c1) was the front node of motion element (c) before extending. The extended path starts from position (b6) and ends at position (c1). After extended, the path of motion element (c) will passes position (c6) and (c7) and starts from position (b6). Position (b6) is the shared position. It is the end node of motion element (b) and is also the front node of motion element (c) after extended. In this way, motion element (b) is connected to motion element (c).

12.3 Outlook

(1) Interactive with Mobile robot. A mobile robot system has been built at celisca, which is used for long-distance transports between workstation. In the future, the mobile robots are expected to

transfer consumables to the dual-arm robot platform, such as source solutions, microplates and tip boxes, and to moving the waste away from the platform.

(2) Failure detecting. The dual-arm robot uses shock sensors to detect the collisions. However, the sensors are not sensitive enough to detect small forces. This means, when some light collisions happen, the robot may not detect them and continues to run. In this case, the following parts of a process will not be performed correctly. Thus, some methods should be used to detect the failures of the robot.

(3) The robot platform has been developed using LC-MS instrument to analyze chiral compounds, using GC-MS instrument to analyze cholesterol samples and using SPE instrument to separate compounds in a liquid mixture. But the robot platform can be extended to do more tasks, e.g. preparing samples for determining calcium in stents.

Appendix

A. Specification of Hardware

A.1 SDA10 Dual-arm Robot

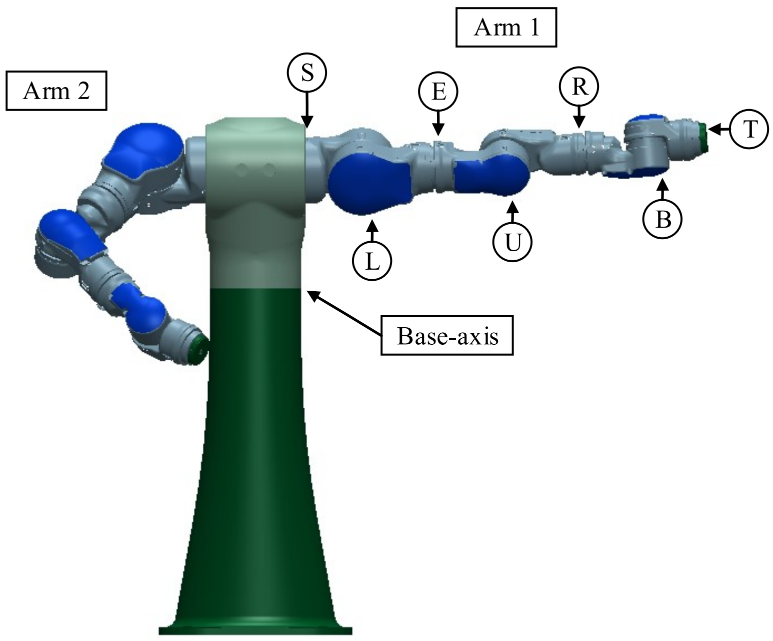


Figure 145. SDA10 dual-arm robot.

Table 13. Basic specifications of SDA10.

Degree of freedom	10
Payload	10kg per arm
Repetitive Positioning Accuracy	±0.1 mm
Power Capacity	1.5 kVA
Approx. Mass	220 kg

Table 14. Parameters of each axis.

	Range of motion	**Maximum speed**
Base-axis	-170° ~ +170°	2.97 rad/s (170°/s)
S-axis	-180° ~ +180°	2.97 rad/s (170°/s)
L-axis	-110° ~ +110°	2.97 rad/s (170°/s)
E-axis	-170° ~ +170°	2.97 rad/s (170°/s)
U-axis	-135° ~ +135°	2.97 rad/s (170°/s)
R-axis	-180° ~ +180°	3.49 rad/s (200°/s)
B-axis	-110° ~ +110°	3.49 rad/s (200°/s)

T-axis	-180° ~ +180°	6.98 rad/s (400°/s)

Table 15. Parameters of robot controller FS100.

Positioning system	Absolute encoder (serial interface)
Programming language	INFORM III
Robot motion control	Joint motion, linear, circular, spline interpolation
Dual-channel safety system	Emergency stop, safety interlock
Collision avoidance	Collision avoidance zones and radial interference zones
Collision detection	Monitors robot axes' torque levels
Interface	Profibus (Slave), Ethernet IP (Master/Slave), DeviceNet (Master/Slave), CC-Link (Slave), Profinet (Master/Slave)
software solutions	MOTOMANSync, MOTOPlus, MOTOGSI

A.2 LEHF Gripper

Table 16. Parameters of gripper.

Model number	LEHF20K2-48-R86P5
Gripping force 40 to 100% (N)	11 to 28
Opening/closing speed (mm/s)	5 to 100
Drive method	Sliding screw and belt bending
Repeatability (mm)	± 0.05
Finger backlash: both sides (mm)	≤ 1.0
Max. operating frequency (c.p.m)	60
Weight (g)	750
Motor	Step motor (Servo 24VDC)
Encoder (Angular displacement sensor)	Increment A/B phase (800 pulse/rotation)
Power consumption (W)	28

Table 17. Parameters of gripper controller.

Model number	LECP6
Power supply	24VDC ± 10%
Parallel input	11 inputs (photo coupler isolation)
Parallel output	13 outputs (photo coupler isolation)
Serial communication	RS485

A.3 I/O Connections between Robot Controller and Gripper Controllers

Table 18. Definitions of I/O pins.

	CN2 (Robot controller side)			Gripper controller side	Job side
	Logical No.	Connector No.	Pin No.	No./Function	Signal No.
Input	20070	2	DI_00	1_B1/OUT0	#00030
	20071	20	DI_01	1_B2/OUT1	#00031
	20072	3	DI_02	1_B3/OUT2	#00032
	20073	21	DI_03	1_B4/OUT3	#00033
	20074	4	DI_04	1_B5/OUT4	#00034
	20075	22	DI_05	1_B6/OUT5	#00035
	20076	6	DI_06	1_B7/Busy	#00036
	20077	24	DI_07	1_B8/Area	#00037
	20080	7	DI_08	1_B9/SetOn	#00060
	20081	25	DI_09	1_B10/INP	#00061
	20082	8	DI_10	1_B11/SVRE	#00062
	20083	26	DI_11	1_B12/ESTOP	#00063
Output	30070	10	DO_00	1_A3/IN0	#10030
	30071	28	DO_01	1_A4/IN1	#10031
	30072	11	DO_02	1_A5/IN2	#10032
	30073	29	DO_03	1_A6/IN3	#10033
	30074	14	DO_04	1_A7/IN4	#10034
	30075	32	DO_05	1_A8/IN5	#10035
	30076	15	DO_06	1_A9/Setup	#10036
	30077	33	DO_07	1_A10/Hold	#10037
	30080	17	DO_08	1_A11/Drive	#10060
	30081	35	DO_09	1_A12/Reset	#10061
	30082	18	DO_10	1_A13/SVOn	#10062
	30083	36	DO_11		#10063
	CN1 (Robot controller side)			Gripper controller side	Job side
	Logical No.	Connector No.	Pin No.	Name	
Input	20090	2	DI_16	1_B13/ALARM	#00040
	20091	27	DI_17	2_B1/OUT0	#00041
	20092	3	DI_18	2_B2/OUT1	#00042
	20093	28	DI_19	2_B3/OUT2	#00043
	20094	4	DI_20	2_B4/OUT3	#00044
	20095	29	DI_21	2_B5/OUT4	#00045
	20096	5	DI_22	2_B6/OUT5	#00046
	20097	30	DI_23	2_B7/Busy	#00047
	20100	7	DI_24	2_B8/Area	#00050
	20101	32	DI_25	2_B9/SetOn	#00051
	20102	8	DI_26	2_B10/INP	#00052
	20103	33	DI_27	2_B11/SVRE	#00053

	20104	9	DI_28	2_B12/ESTOP	#00054
	20105	34	DI_29	2_B13/ALARM	#00055
	20106	10	DI_30		#00056
	20107	35	DI_31		#00057
Output	30090	12	DO_16	2_A3/IN0	#10040
	30091	37	DO_17	2_A4/IN1	#10041
	30092	13	DO_18	2_A5/IN2	#10042
	30093	38	DO_19	2_A6/IN3	#10043
	30094	16	DO_20	2_A7/IN4	#10044
	30095	41	DO_21	2_A8/IN5	#10045
	30096	17	DO_22	2_A9/Setup	#10046
	30097	42	DO_23	2_A10/Hold	#10047
	30100	19	DO_24	2_A11/Drive	#10050
	30101	44	DO_25	2_A12/Reset	#10051
	30102	20	DO_26	2_A13/SVOn	#10052
	30103	45	DO_27		#10053
	30104	23	DO_28		#10054
	30105	48	DO_29		#10055
	30106	24	DO_30		#10056
	30107	49	DO_31		#10057

B. Specification of Software

B.1 Robot Jobs

Table 19. List of robot jobs for realizing motion elements.

No.	Job name	Translated name
1	GP_AFT_PUT_GP_T_RACK.JBI	After Put Glass Pipette To Rack
2	GP_GET_L_F_VIAL.JBI	Get Liquid From Vial
3	GP_OUT_L_T_VIAL.JBI	Out Liquid To Vial
4	GP_PICK_GP_F_RACK.JBI	Pick Glass Pipette From Rack
5	GP_PRE_GET_L_F_VIAL.JBI	Prepare Get Liquid From Vial
6	GP_PRE_PICK_GP_F_RACK.JBI	Prepare Pick Glass Pipette From Rack
7	GP_PRE_PUT_GP_T_RACK.JBI	Prepare Put Glass Pipette To Rack
8	GP_PUT_GP_T_RACK.JBI	Put Glass Pipette To Rack
9	HO_AFT_PUT_MTP_T_HOTEL.JBI	After Put Microplate To Hotel
10	HO_AFT_PUT_MTP_T_TABLE.JBI	After Put Microplate To Table
11	HO_PICK_MTP_F_HOTEL.JBI	Pick Microplate From Hotel
12	HO_PICK_MTP_F_TABLE.JBI	Pick Microplate From Table
13	HO_PICK_RACK_F_SK.JBI	Pick Rack From SK
14	HO_PRE_PICK_MTP_F_HOTEL.JBI	Prepare Pick Microplate From Hotel
15	HO_PRE_PICK_MTP_F_TABLE.JBI	Prepare Pick Microplate From Table
16	HO_PRE_PUT_MTP_T_HOTEL.JBI	Prepare Put Microplate To Hotel
17	HO_PRE_PUT_MTP_T_TABLE.JBI	Prepare Put Microplate To Table

18	HO_PUT_MTP_T_HOTEL.JBI	Put Microplate To Hotel
19	HO_PUT_MTP_T_TABLE.JBI	Put Microplate To Table
20	HO_PUT_RACK_T_SK.JBI	Put Rack To SK
21	LC_AFT_ENABLE_R1_T_OBJ.JBI	After Enable R1 To Object
22	LC_AFT_ENABLE_R2_T_OBJ.JBI	After Enable R2 To Object
23	LC_AFT_PUT_MTP_T_HO.JBI	After Put Microplate To Hotel
24	LC_CLOSE_DOOR_T_LC.JBI	Close Door To LC
25	LC_ENABLE_R1_T_OBJ.JBI	Enable R1 To Object
26	LC_ENABLE_R2_T_OBJ.JBI	Enable R2 To Object
27	LC_OPEN_DOOR_F_LC.JBI	Open Door From LC
28	LC_PICK_MTP_F_DRAWER.JBI	Pick Microplate From Drawer
29	LC_PRE_PICK_MTP_F_DRAWER.JBI	Prepare Pick Microplate From Drawer
30	LC_PRE_PUT_MTP_T_HO.JBI	Prepare Put Microplate To Hotel
31	LC_PUT_DRAWER_T_LC.JBI	Put Drawer To LC
32	LC_PUT_DRAWER_T_TABLE.JBI	Put Drawer To Table
33	LC_PUT_MTP_T_HOTEL.JBI	Put Microplate To Hotel
34	PE_BACK_STANDBY_0_4.JBI	Back Standby 0 4
35	PE_BACK_STANDBY_5_7.JBI	Back Standby 5 7
36	PE_GET_MTP_M10_REF.JBI	Get Microplate 10uLMulti REF
37	PE_GET_TA_M100_REF.JBI	Get Tank 100uLMulti REF
38	PE_GET_TA_M10_REF.JBI	Get Tank 10uLMulti REF
39	PE_GET_TA_M300_REF.JBI	Get Tank 300uLMulti REF
40	PE_GET_TA_S1000_REF.JBI	Get Tank 1000uLSingle REF
41	PE_GET_TA_S10M_REF.JBI	Get Tank 10mLSingle REF
42	PE_GET_TA_S200_REF.JBI	Get Tank 200uLSingle REF
43	PE_GET_TA_S5M_REF.JBI	Get Tank 5mLSingle REF
44	PE_GET_VI10_1000-T2_REF.JBI	Get 10mLVial 1000-T2 REF
45	PE_GET_VI10_1000_REF.JBI	Get 10mLVial 1000uLSingle REF
46	PE_GET_VI10_S200_REF.JBI	Get 10mLVial 200uLSingle REF
47	PE_GET_VI22_1000_REF.JBI	Get 22mLVial 1000uLSingle REF
48	PE_GET_VI22_10ML_REF.JBI	Get 22mLVial 10mLSingle REF
49	PE_GET_VI4_S100_REF.JBI	Get 4mLVial 100uLSingle REF
50	PE_LOAD_1_300UL_M100.JBI	Load 1 300UL 100uLMulti
51	PE_LOAD_1_300UL_M300.JBI	Load 1 300UL 300uLMulti
52	PE_LOAD_1_300UL_S100.JBI	Load 1 300UL 100uLSingle
53	PE_LOAD_1_300UL_S200.JBI	Load 1 300UL 200uLSingle
54	PE_LOAD_TIP_1000-T2.JBI	Load Tip 1000-T2
55	PE_LOAD_TIP_1000.JBI	Load Tip 1000uLSingle
56	PE_LOAD_TIP_10ML.JBI	Load Tip 10mLSingle
57	PE_LOAD_TIP_5ML.JBI	Load Tip 5mLSingle
58	PE_LOAD_TIP_M10.JBI	Load Tip 10uLMulti
59	PE_OUT_MTP_M100_REF.JBI	Out Microplate 100uLMulti REF
60	PE_OUT_MTP_M10_REF.JBI	Out Microplate 10uLMulti REF
61	PE_OUT_MTP_M300_REF.JBI	Out Microplate 300uLMulti REF
62	PE_OUT_MTP_S100_REF.JBI	Out Microplate 100uLSingle REF
63	PE_OUT_MTP_S200_REF.JBI	Out Microplate 200uLSingle REF
64	PE_OUT_TA_S10M_REF.JBI	Out Tank 10mLSingle REF
65	PE_OUT_VI10_1000_REF.JBI	Out 10mLVial 1000uLSingle REF
66	PE_OUT_VI10_5M_REF.JBI	Out 10mLVial 5mLSingle REF

67	PE_OUT_VI10_S10ML_REF.JBI	Out 10mLVial 10mLSingle REF
68	PE_OUT_VI10_S200_REF.JBI	Out 10mLVial 200uLSingle REF
69	PE_OUT_VI22_1000_REF.JBI	Out 22mLVial 1000uLSingle REF
70	PE_OUT_VI2_1000-T2_REF.JBI	Out VI2 1000-T2 REF
71	PE_OUT_VI2_S100_REF.JBI	Out VI2 100uLSingle REF
72	PE_OUT_VI4_1000_REF.JBI	Out 4mLVial 1000uLSingle REF
73	PE_PICK_PE_F_TEMP.JBI	Pick Pipette From Temporary Position
74	PE_PICK_PIP_0_4.JBI	Pick Pipette 0 4
75	PE_PICK_PIP_5_7.JBI	Pick Pipette 5 7
76	PE_PIP_THIRD_POS.JBI	Pipette Third Position
77	PE_PRE_LOAD_TIPS_0_4.JBI	Prepare Load TIPS 0 4
78	PE_PRE_LOAD_TIPS_5_7.JBI	Prepare Load TIPS 5 7
79	PE_PRE_PICK_PIP_0_4.JBI	Prepare Pick Pipette 0 4
80	PE_PRE_PICK_PIP_5_7.JBI	Prepare Pick Pipette 5 7
81	PE_PRE_PIP_GET_LEQUID.JBI	Prepare Pipette Get Liquid
82	PE_PRE_PUT_PIP_0_4.JBI	Prepare Put Pipette 0 4
83	PE_PRE_PUT_PIP_5_7.JBI	Prepare Put Pipette 5 7
84	PE_PUT_PE_T_TEMP.JBI	Put Pipette To Temporary Position
85	PE_PUT_PIP_0_4.JBI	Put Pipette 0 4
86	PE_PUT_PIP_5_7.JBI	Put Pipette 5 7
87	PE_RELEASE_TIP_1000-T2.JBI	Release Tip 1000-T2
88	PE_RELEASE_TIP_1000.JBI	Release Tip 1000uLSingle
89	PE_RELEASE_TIP_10ML.JBI	Release Tip 10mLSingle
90	PE_RELEASE_TIP_5ML.JBI	Release Tip 5mLSingle
91	PE_RELEASE_TIP_M10.JBI	Release Tip 10uLMulti
92	PE_RELEASE_TIP_M100.JBI	Release Tip 100uLMulti
93	PE_RELEASE_TIP_M300.JBI	Release Tip 300uLMulti
94	PE_RELEASE_TIP_S100.JBI	Release Tip 100uLSingle
95	PE_RELEASE_TIP_S200.JBI	Release Tip 200uLSingle
96	SY_GET_LIQUID_CANULA_2.JBI	Get LIQUID Cannula 2
97	SY_OUT_LIQUID_FILTER.JBI	Out LIQUID Filter
98	SY_PICK_CANULA_F_RACK.JBI	Pick Cannula From Rack
99	SY_PICK_FILTER_F_RACK.JBI	Pick Filter From Rack
100	SY_PICK_FILTER_F_RACK_2.JBI	Pick Filter From Rack 2
101	SY_PICK_SYR_F_RACK.JBI	Pick Syringe From Rack
102	SY_PUT_SYR_T_G1.JBI	Put Syringe To G1
103	SY_WASTE_CANULA.JBI	Waste Cannula
104	SY_WASTE_SYRINGE_FILTER.JBI	Waste Syringe Filter
105	TI_AFT_PUT_TI_T_SHELF_0-6.JBI	After Put Tip Box To Shelf 0-6
106	TI_AFT_PUT_TI_T_TABLE.JBI	After Put Tip Box To Table
107	TI_PICK_TI_F_SHELF_0-6.JBI	Pick Tip Box From Shelf 0-6
108	TI_PICK_TI_F_TABLE.JBI	Pick Tip Box From Table
109	TI_PRE_PICK_TI_F_SHELF_0-6.JBI	Prepare Pick Tip Box From Shelf 0-6
110	TI_PRE_PICK_TI_F_TABLE.JBI	Prepare Pick Tip Box From Table
111	TI_PRE_PUT_TI_TO_SHELF_0-6.JBI	Prepare Put Tip Box To Shelf 0-6
112	TI_PRE_PUT_TI_TO_TABLE_0-6.JBI	Prepare Put Tip Box To Table 0-6
113	TI_PUT_TI_T_SHELF_0-6.JBI	Put Tip Box To Shelf 0-6
114	TI_PUT_TI_T_TABLE.JBI	Put Tip Box To Table
115	TM_ACCESS_R2_T_TM.JBI	Access R2 To Thermo shaker

116	TM_AFT_ACCESS_R2_T_TM.JBI	After Access R2 To Thermo shaker
117	TM_AFT_ENABLE_R1_T_OBJ.JBI	After Enable R1 To Object
118	TM_AFT_ENABLE_R2_T_OBJ.JBI	After Enable R2 To Object
119	TM_AFT_PUT_MTP_T_LAB.JBI	After Put Microplate To Lab ware
120	TM_CLOSE_LID_T_TM.JBI	Close Lid To Thermo shaker
121	TM_ENABLE_R1_T_OBJ.JBI	Enable R1 To Object
122	TM_ENABLE_R2_T_OBJ.JBI	Enable R2 To Object
123	TM_OPEN_LID_F_TM.JBI	Open Lid From Thermo shaker
124	TM_PICK_MTP-LID_F_DRAWER.JBI	Pick Microplate Lid From Drawer
125	TM_PICK_MTP-LID_F_TABLE.JBI	Pick Microplate Lid From Table
126	TM_PICK_MTP-LID_F_TM.JBI	Pick Microplate Lid From Thermo shaker
127	TM_PICK_MTP_F_TABLE.JBI	Pick Microplate From Table
128	TM_PICK_MTP_F_TM-H.JBI	Pick Microplate From Thermo shaker
129	TM_PRESS_BUTTON_T_TM.JBI	PRESS Button To Thermo shaker
130	TM_PRE_PICK_MTP_F_LAB.JBI	Prepare Pick Microplate From Lab ware
131	TM_PUT_MTP-LID_T_DRAWER.JBI	Put Microplate Lid To Drawer
132	TM_PUT_MTP-LID_T_TABLE.JBI	Put Microplate Lid To Table
133	TM_PUT_MTP-LID_T_TM.JBI	Put Microplate Lid To Thermo shaker
134	TM_PUT_MTP_T_DRAWER.JBI	Put Microplate To Drawer
135	TM_PUT_MTP_T_TABLE.JBI	Put Microplate To Table
136	TM_PUT_MTP_T_TM-H.JBI	Put Microplate To Thermo shaker High
137	VI_AFT_PUT_GC.JBI	After Put GC
138	VI_AFT_PUT_VIAL_T_RACK.JBI	After Put Vial To Rack
139	VI_AFT_SWITCH_US_SHAKER.JBI	After Switch Ultrasonic Shaker
140	VI_CLOSE_LID_T_VIAL.JBI	Close Lid To Vial
141	VI_OPEN_LID_F_VIAL.JBI	Open Lid From Vial
142	VI_PICK_LID_F_RACK.JBI	Pick Lid From Rack
143	VI_PICK_VIAL_F_SHAKER.JBI	Pick Vial From Shaker
144	VI_PICK_VIAL_F_US.JBI	Pick Vial From Ultrasonic
145	VI_PICK_VIAL_N-LID.JBI	Pick Vial N-LID
146	VI_PICK_VIAL_Y-LID.JBI	Pick Vial Y-LID
147	VI_PRE_CLOSE_LID_T_VIAL.JBI	Prepare Close Lid To Vial
148	VI_PRE_PICK_LID_F_RACK.JBI	Prepare Pick Lid From Rack
149	VI_PRE_PICK_VIAL_F_RACK.JBI	Prepare Pick Vial From Rack
150	VI_PRE_PUT_GC.JBI	Prepare Put GC
151	VI_PRE_PUT_LID_T_RACK.JBI	Prepare Put Lid To Rack
152	VI_PRE_PUT_LID_T_RACK_2.JBI	Prepare Put Lid To Rack 2
153	VI_PRE_PUT_VIAL_T_RACK.JBI	Prepare Put Vial To Rack
154	VI_PRE_SWITCH_US_SHAKER.JBI	Prepare Switch Ultrasonic Shaker
155	VI_PUT_LID_T_RACK.JBI	Put Lid To Rack
156	VI_PUT_VIAL_N-LID.JBI	Put Vial N-LID
157	VI_PUT_VIAL_T_SHAKER.JBI	Put Vial To Shaker
158	VI_PUT_VIAL_T_US.JBI	Put Vial To Ultrasonic
159	VI_PUT_VIAL_Y-LID.JBI	Put Vial Y-LID

Table 20. List of parameters of relative motion elements[11].

Variable Name	Usage
B000	Calculate lab ware base NO.
B010	Hotel row (Which room)
B011	Hotel column (Which hotel)
B018	Base No.
B019	Vial type (0: 22mL, 1: 10mL, 2: 4mL, 3: 2mL)
B020	Vial/Cap rack: X direction
B021	Vial/Cap rack: Y direction
B030	GP rack: X direction
B031	GP rack: Y direction
B032	Tip box bases on shelf
B033	Hotel: 0: 96-well 1: 96-well-soft Shelf: 0: ADAP 1: 96-well-soft
B035	Rack type: 0: MTP or 2mL vial rack 1: 10mL vial rack 2: reserved 3: reserved 4: syringe rack 6: reserved 7: tip box
B038	Repeat time in pipette mode
B039	Pipette mode: 0: normal 1: pre-wetting 2: final pipetting (pipette) 3: final pipetting (glass pipette)
B040	Pipette position on shelf
B043	Tip box: X direction
B044	Tip box: Y direction
B050	Reservoirs: X direction
B051	Microplate: X direction
B052	Microplate: Y direction

[11] The X, Y direction can be found in Figure 48.

Table 21. List of robot jobs for driving gripper 1[12].

Job Name	Usage
G1_1000.JBI	Move gripper fingers to 1 inch position
G1_GP-GET-LIQUID.JBI	Squeeze the rubber head of glass pipette for getting liquid
G1_GP-OUT-LIQUID.JBI	Squeeze the rubber head of glass pipette for releasing liquid
G1_GP-PRE-PRESS.JBI	Move gripper fingers close to the rubber head of glass pipette
G1_GP-PRESS-AIR.JBI	Squeeze the rubber head of glass pipette to press the air out
G1_GRIP-96-WELL.JBI	Grip microplate tightly
G1_GRIP-96-WELL_SOFT.JBI	Grip microplate loosely
G1_GRIPPER-OPEN.JBI	Open gripper figures maximally
G1_GRIPPER-POWER-OFF.JBI	Switch off powder supply to motors of gripper
G1_GRIPPER_CLOSE.JBI	Close gripper figures maximally
G1_GRIPPER_POWER_ON.JBI	Switch on powder supply to motors of gripper
G1_GRIPPER_RESET.JBI	Reset gripper
G1_GRIPPER_TEST.JBI	Initialize gripper
G1_LIQUID-IN.JBI	Release the rubber head of glass pipette to suck liquid
G1_LIQUID-OUT.JBI	Squeeze the rubber head of glass pipette for releasing liquid out
G1_MTP-LID.JBI	Grip microplate lid
G1_RELATIVE_MOVE-N01.JBI	Relative movement: close 0.01 inch / run
G1_RELATIVE_MOVE-P01.JBI	Relative movement: open 0.01 inch / run
G1_SPRITZE.JBI	Grip syringe
G1_TBOX_ADAP.JBI	Grip the adapter of tip box
G1_VIAL-10.JBI	Grip 10-mL vial
G1_VIAL-2.JBI	Grip 2-mL vial
G1_VIAL-22.JBI	Grip 22-mL vial
G1_VIAL-4.JBI	Grip 4-mL vial

Table 22. List of robot jobs for driving gripper 2.

Job Name	Usage
G2_GRIP-GP-SOFT.JBI	Grip the body of glass pipette loosely
G2_GRIP-HOLF_GP.JBI	Grip the body of glass pipette tightly
G2_GRIP-PIPET-HARD.JBI	Grip adapter of pipette tightly
G2_GRIP-PIPET-SOFT.JBI	Grip adapter of pipette loosely
G2_GRIPPER-CLOSE.JBI	Close gripper figures maximally
G2_GRIPPER-OPEN.JBI	Open gripper figures maximally
G2_GRIPPER-POWER-OFF.JBI	Switch off powder supply to motors of gripper
G2_GRIPPER_POWER_ON.JBI	Switch on powder supply to motors of gripper
G2_GRIPPER_RESET.JBI	Reset gripper
G2_GRIPPER_TEST.JBI	Initialize gripper

[12] Gripper 1 on left arm, gripper 2 on right arm.

G2_RELATIVE-N005.JBI	Relative movement: close 0.005 inch / run
G2_RELATIVE-N01.JBI	Relative movement: close 0.01 inch / run
G2_RELATIVE-P005.JBI	Relative movement: open 0.005 inch / run
G2_RELATIVE-P01.JBI	Relative movement: open 0.01 inch / run
G2_SPRITZE_TOP.JBI	Grip head of syringe piston
G2_TM_SHAKER_LID.JBI	Grip the handle on the lid of thermo shaker
G2_VIAL-BODY-10.JBI	Grip body of 10-mL vial
G2_VIAL-BODY-2.JBI	Grip body of 2-mL vial
G2_VIAL-BODY-22.JBI	Grip body of 22-mL vial
G2_VIAL-BODY-4.JBI	Grip body of 2-mL vial
G2_VIAL-TIP-10.JBI	Grip cap of 10-mL vial
G2_VIAL-TIP-2.JBI	Grip cap of 2-mL vial
G2_VIAL-TIP-22.JBI	Grip cap of 22-mL vial
G2_VIAL-TIP-4.JBI	Grip cap of 4-mL vial

B.2 Examples of Call-files

```
NOP
SET B018 4
SET B020 3
SET B021 0
SET B019 2
CALL JOB:VI_PICK_VIAL_Y-LID
CALL JOB:VI_OPEN_LID_F_VIAL
CALL JOB:VI_PRE_PUT_LID_T_RACK
SET B018 9
SET B020 1
SET B021 2
SET B019 2
CALL JOB:VI_PUT_LID_T_RACK
CALL JOB:VI_PRE_PUT_VIAL_T_RACK
SET B018 4
SET B020 3
SET B021 0
SET B019 2
CALL JOB:VI_PUT_VIAL_N-LID
END
```

Figure 146. Call-file of opening vial.

```
NOP
CALL JOB:LC_ENABLE_R2_T_OBJ
CALL JOB:LC_OPEN_DOOR_F_LC
CALL JOB:LC_PUT_DRAWER_T_LC
CALL JOB:LC_CLOSE_DOOR_T_LC
CALL JOB:LC_AFT_ENABLE_R2_T_OBJ
END
```

Figure 147. Call-file of transferring samples to LC auto sampler.

```
NOP
CALL JOB:TI_PRE_PICK_TI_F_TABLE
SET B018 7
SET B035 7
SET B033 1
CALL JOB:TI_PICK_TI_F_TABLE
CALL JOB:TI_PRE_PUT_TI_TO_SHELF_0-6
SET B032 2
SET B035 7
SET B033 1
CALL JOB:TI_PUT_TI_T_SHELF_0-6
CALL JOB:TI_AFT_PUT_TI_T_SHELF_0-6
END
```

Figure 148. Call-file of transferring tip box from table to shelf.

```
NOP
CALL JOB:PE_PRE_PICK_PIP_0_4
SET B040 0
CALL JOB:PE_PICK_PIP_0_4
CALL JOB:PE_PRE_LOAD_TIPS_0_4
SET B043 5
SET B044 0
CALL JOB:PE_LOAD_TIP_10ML
CALL JOB:PE_PRE_PIP_GET_LEQUID
SET B018 3
SET B020 0
SET B021 1
SET B039 1
SET B038 3
CALL JOB:PE_GET_VI22_10ML_REF
SET B018 0
SET B020 0
SET B021 0
SET B039 0
CALL JOB:PE_OUT_VI10_S10ML_REF
SET B018 3
SET B020 0
SET B021 1
SET B039 0
CALL JOB:PE_GET_VI22_10ML_REF
SET B018 0
SET B020 0
SET B021 1
SET B039 0
CALL JOB:PE_OUT_VI10_S10ML_REF
CALL JOB:PE_RELEASE_TIP_10ML
CALL JOB:PE_PRE_PUT_PIP_0_4
SET B040 0
CALL JOB:PE_PUT_PIP_0_4
CALL JOB:PE_BACK_STANDBY_0_4
END
```

Figure 149. Call-file of pipetting liquid using 10mL pipette

B.3 Robot Status Data

Table 23. Specification of robot status data.

	Bit	Description	Value
Data 1	Bit 0	Step	1: yes, 0: no
	Bit 1	1 cycle	
	Bit 2	Automatic and continuous	
	Bit 3	Running	
	Bit 4	In-guard safe operation	
	Bit 5	Teach	
	Bit 6	Play	
	Bit 7	Command remote	
Data 2	Bit 0	-	1: yes, 0: no
	Bit 1	Hold by program pendant	
	Bit 2	Hold externally	
	Bit 3	Hold by command	
	Bit 4	Alarming	
	Bit 5	Error occurring	
	Bit 6	Servo ON	
	Bit 7	-	
IO (80020)	Bit 3	Stop by safety guard	1: yes, 0: no
	Bit 4	Main CPU error	
	Bit 5	Stop by light curtain	
	Bit 6	Stop by program pendant	

C. SAMI Methods

C.1 Dealing with Vial

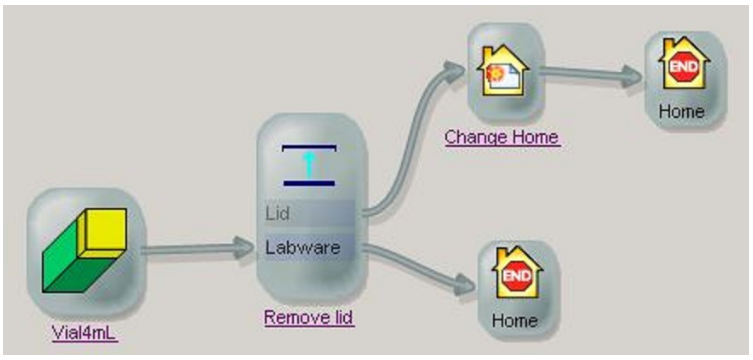


Figure 150. SAMI method of opening vials.

C.2 Pipetting Solution

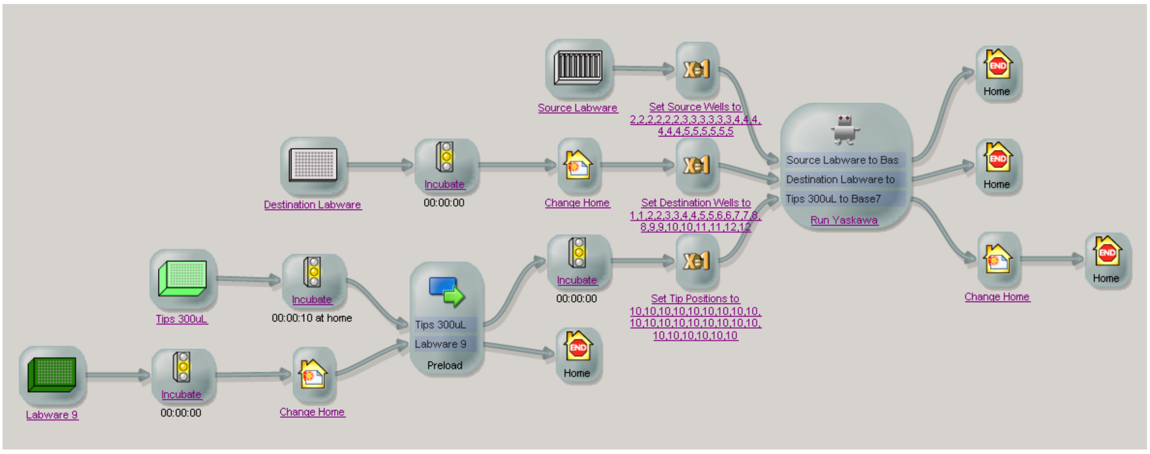


Figure 151. SAMI method of pipetting solutions.

C.3 Shaking Sample by Thermo Shaker

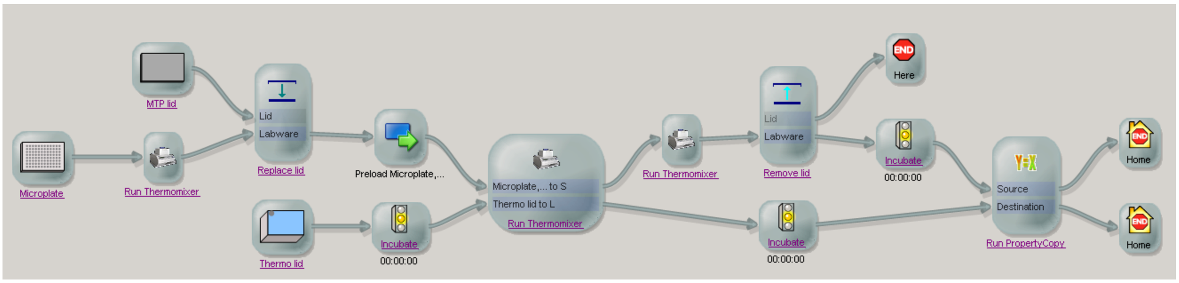


Figure 152. SAMI method about shaking samples by thermo shaker.

C.4 Sample Transport

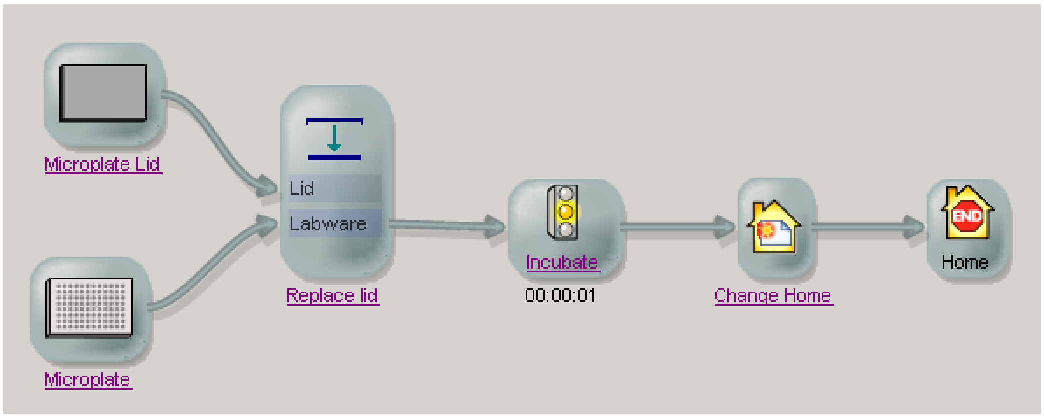


Figure 153. Transferring samples to LC autosampler.

D. Testing Records

D.1 Records about Chiral Compound Process

In a test record, four marks had been used to record the results, as following:

- S: The robot did a subtask **s**uccessfully.
- F: The robot was **f**ault to do a subtask.
- F_c: The robot was **f**ault to do a subtask, due to the wrong **c**onfiguration. Placing labware wrongly, changing settings of robot (e.g. tools or position variables) or changing the gripper fingers is possibly to cause the robot fault to do something.
- D: The robot did a subtask successfully in some degree, but not perfectly. So, it was marked as "**d**efect". Improvements are required.

A complete record table of the 1st test was given. The parts with failures or "defects" in the record tables of rest tests were provided, and the parts with no failures or "defects" were not shown.

Test result: 1st test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S		
		Unscrew cap from vial	S	S	S	S	S	S	S	S	S	S		
		Put cap to rack	S	S	S	S	S	S	S	S	S	S		
		Put vial to rack	S	S	S	S	S	S	S	S	S	S		
2	Transfer tip box	Pick box from shelf	S	S	S	S	S	S						
		Prepare to put to table	S	S	S	S	S	S						
		Put box to table	S	S	S	S	S	S						
		Pick box from table	S	S	S	S	S	S						
		Prepare to put to shelf	S	S	S	S	S	S						
		Put box to shelf	S	S	S	S	S	S						
3	10mL pipet	Pick from shelf	S	S										
		Load tip to pipet	S	S										
		Draw liquid	S	S	S	S								
		Push out liquid	S	S	S	S								
		Release tip from pipet	S	S										
		Put pipet to shelf	S	S										
4	5mL pipet	Pick from shelf	S	S	S	S								
		Load tip to pipet	S	S	S	S								
		Draw liquid	S	S	S	S								
		Push out liquid	S	S	S	S								
		Release tip from pipet	S	S	S	S								
		Put pipet to shelf	S	S	S	S								
5	1000uL Pipet	Pick from shelf	S											
		Load tip to pipet	S	S										
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S				
		Push out liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S				
		Release tip from pipet	S	S										
		Put pipet to shelf	S											
6	200uL pipet	Pick from shelf	S											
		Load tip to pipet	S	S										
		Draw liquid	S	S										
		Push out liquid	S	S										
		Release tip from pipet	S	S										
		Put pipet to shelf	S											
7	100uL pipet	Pick from shelf	S											
		Load tip to pipet	S	S	S	S	S	S						
		Draw liquid	Total times: 96, S: 96, F: 0.											

		Push out liquid	Total times: 96, S: 96, F: 0.											
		Release tip from pipet	S	S	S	S	S	S						
		Put pipet to shelf	S											
8	300uL 8-pipte	Pick from shelf	S											
		Load tip to pipet	S											
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Push out liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Release tip from pipet	S											
		Put pipet to shelf	S											
9	100uL 8-pipte	Pick from shelf	S											
		Load tip to pipet	S											
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
		Push out liquid	S	S	S	S	S	S	S	S	S	S	S	S
		Release tip from pipet	S											
		Put pipet to shelf	S											
10	10uL 8-pipte	Pick from shelf	S	S	S									
		Load tip to pipet	S	S	S	S	S	S	S	S	S	S	S	S
			S	S										
		Draw liquid	Total times: 48, S: 48, F: 0.											
		Push out liquid	Total times: 48, S: 48, F: 0.											
		Release tip from pipet	S	S	S	S	S	S	S	S	S	S	S	S
			S	S										
		Put pipet to shelf	S	S	S									
11	Glass pipet	Pick from shelf	S	S	S	S								
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Release liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put to shelf	S	S	S	S								
12	Transfer microplate	Pick from hotel	S	S										
		Prepare to put to table	F_c	S										
		Put to table	S	S										
13	Thermo shaker (TM)	Remove TM lid	S	S	S	S	S	S						
		Transfer microplate to TM	S	S	S									
		Transfer microplate lid to TM	S	S	S									
		Cover TM lid	S	S	S	S	S	S						
		Transfer microplate lid to table	S	S	S									
		Transfer microplate to table	S	S	S									
14	Autosampler	Transfer microplate to LC-tray	S											
		Open door of LC-autosampler	S											
		Transfer LC-tray to autosampler	S											
		Close door of LC-autosampler	S											
		Open door of LC-autosampler	S											
		Transfer LC-tray to table	S											
		Close door of LC-autosampler	S											
		Transfer microplate to table	S											

Test result: 2nd test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
8	300uL 8-pipte	Pick from shelf	S											
		Load tip to pipet	F											
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Push out liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Release tip from pipet	S											
		Put pipet to shelf	S											
9	100uL 8-pipte	Pick from shelf	S											
		Load tip to pipet	S											
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S

		Push out liquid	D	D	D	D	D	D	D	D	D	D	D	D
		Release tip from pipet	S											
		Put pipet to shelf	S											
14	Autosampler	Transfer microplate to LC-tray	S											
		Open door of LC-autosampler	S											
		Transfer LC-tray to autosampler	S											
		Close door of LC-autosampler	S											
		Open door of LC-autosampler	S											
		Transfer LC-tray to table	S											
		Close door of LC-autosampler	S											
		Transfer microplate to table	F											

Test result: 3rd test about chiral compound

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	S	S	S	D	S	S	S	S	S	S		
		Unscrew cap from vial	S	S	S	S	S	S	S	S	S	S		
		Put cap to rack	S	S	S	S	S	S	S	S	S	S		
		Put vial to rack	S	S	S	S	S	S	S	S	S	S		
2	Transfer tip box	Pick box from shelf	S	S	S	S	S	S						
		Prepare to put to table	S	S	S	S	S	S						
		Put box to table	F_c	S	S	S	S	S						
		Pick box from table	S	S	S	S	S	S						
		Prepare to put to shelf	S	S	S	S	S	S						
		Put box to shelf	S	S	S	S	S	S						
3	10mL pipet	Pick from shelf	S	S										
		Load tip to pipet	F_c	F_c										
		Draw liquid	S	S	S	S								
		Push out liquid	S	S	S	S								
		Release tip from pipet	F_c	F_c										
		Put pipet to shelf	S	S										
5	1000uL Pipet	Pick from shelf	S											
		Load tip to pipet	S	S										
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S				
		Push out liquid	D	D	D	D	D	D	D	D	D	D	D	D
			D	D	D	D	D	D	D	D				
		Release tip from pipet	S	S										
		Put pipet to shelf	S											
10	10uL 8-pipte	Pick from shelf	S	S	S									
		Load tip to pipet	S	F	S	S	S	S	S	S	S	S	S	S
			S	S										
		Draw liquid	Total times: 48, S: 48, F: 0.											
		Push out liquid	Total times: 48, S: 48, F: 0.											
		Release tip from pipet	F	S	S	S	S	S	S	S	S	S	S	S
			S	S										
		Put pipet to shelf	S	S	S									
13	Thermo shaker (TM)	Remove TM lid	S	S	S	S	S	S						
		Transfer microplate to TM	S	S	S									
		Transfer microplate lid to TM	F	F	F									
		Cover TM lid	S	S	S	S	S	S						
		Transfer microplate lid to table	F	F	F									
		Transfer microplate to table	S	S	S									

Test result: 4th test about chiral compound

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12
3	10mL pipet	Pick from shelf	S	S										
		Load tip to pipet	F_c	F_c										
		Draw liquid	S	S	S	S								
		Push out liquid	S	S	S	S								
		Release tip from pipet	S	S										
		Put pipet to shelf	S	S										

5	1000uL Pipet	Pick from shelf	S											
		Load tip to pipet	S	S										
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S				
		Push out liquid	D	D	D	D	D	D	D	D	D	D	D	D
			D	D	D	D	D	D	D	D				
		Release tip from pipet	S	S										
		Put pipet to shelf	S											
10	10uL 8-pipte	Pick from shelf	S	S	S									
		Load tip to pipet	S	S	S	S	S	S	S	S	S	F	S	S
			S	S										
		Draw liquid	Total times: 48, S: 48, F: 0.											
		Push out liquid	Total times: 48, S: 48, F: 0.											
		Release tip from pipet	S	S	S	S	S	S	S	S	F	S	S	S
			S	S										
		Put pipet to shelf	S	S	S									

Test result: 5th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
8	300uL 8-pipte	Pick from shelf	S											
		Load tip to pipet	D											
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Push out liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Release tip from pipet	S											
		Put pipet to shelf	S											
10	10uL 8-pipte	Pick from shelf	S	S	S									
		Load tip to pipet	S	S	S	D	S	S	S	S	S	S	S	S
			S	S										
		Draw liquid	Total times: 48, S: 48, F: 0.											
		Push out liquid	Total times: 48, S: 48, F: 0.											
		Release tip from pipet	S	S	S	S	S	S	S	S	S	S	S	S
			S	S										
		Put pipet to shelf	S	S	S									

Test result: 6th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
5	1000uL Pipet	Pick from shelf	S											
		Load tip to pipet	S	S										
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S				
		Push out liquid	D	D	D	D	D	D	D	D	D	D	D	D
			D	D	D	D	D	D	D	D				
		Release tip from pipet	S	S										
		Put pipet to shelf	S											
10	10uL 8-pipte	Pick from shelf	S	S	S									
		Load tip to pipet	S	S	F	S	S	S	S	S	S	S	S	S
			S	S										
		Draw liquid	Total times: 48, S: 48, F: 0.											
		Push out liquid	Total times: 48, S: 48, F: 0.											
		Release tip from pipet	S	S	S	S	S	S	S	S	S	S	S	S
			S	S										
		Put pipet to shelf	S	S	S									

Test result: 7th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
2	Transfer tip box	Pick box from shelf	S	S	S	S	S	S						
		Prepare to put to table	S	S	S	S	S	S						
		Put box to table	F_c	S	S	S	S	S						

		Pick box from table	S	S	S	S	S	S						
		Prepare to put to shelf	S	S	S	S	S	S						
		Put box to shelf	S	S	S	S	S	S						
5	1000uL Pipet	Pick from shelf	S											
		Load tip to pipet	S	S										
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S				
		Push out liquid	D	D	D	D	D	D	D	D	D	D	D	D
			D	D	D	D	D	D	D	D				
		Release tip from pipet	S	S										
		Put pipet to shelf	S											
10	10uL 8-pipte	Pick from shelf	S	S	S									
		Load tip to pipet	S	S	F	S	S	S	S	S	S	S	S	S
			S	S										
		Draw liquid	Total times: 48, S: 48, F: 0.											
		Push out liquid	Total times: 48, S: 48, F: 0.											
		Release tip from pipet	S	S	S	S	S	S	S	S	S	S	S	S
			S	S										
		Put pipet to shelf	S	S	S									

Test result: 8th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 9th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	S	S	S	D	S	S	S	S	S	S		
		Unscrew cap from vial	S	S	S	S	S	S	S	S	S	S		
		Put cap to rack	S	S	S	S	S	S	S	S	S	S		
		Put vial to rack	S	S	S	S	S	S	S	S	S	S		
8	300uL 8-pipte	Pick from shelf	S											
		Load tip to pipet	F											
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Push out liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Release tip from pipet	S											
		Put pipet to shelf	S											
9	100uL 8-pipte	Pick from shelf	S											
		Load tip to pipet	S											
		Draw liquid	D	D	D	D	D	D	D	D	D	D	D	D
		Push out liquid	S	S	S	S	S	S	S	S	S	S	S	S
		Release tip from pipet	S											
		Put pipet to shelf	S											

Test result: 10th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
7	100uL pipet	Pick from shelf	S											
		Load tip to pipet	D	S	D	S	S	S						
		Draw liquid	Total times: 96, S: 96, F: 0.											
		Push out liquid	Total times: 96, S: 96, F: 0.											
		Release tip from pipet	S	S	S	S	S	S						
		Put pipet to shelf	S											
9	100uL 8-pipte	Pick from shelf	S											
		Load tip to pipet	F											
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
		Push out liquid	S	S	S	S	S	S	S	S	S	S	S	S
		Release tip from pipet	S											
		Put pipet to shelf	S											
11	Glass pipet	Pick from shelf	S	S	S	S								
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S

		Release liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put to shelf	D	D	S	S								
13	Thermo shaker (TM)	Remove TM lid	S	S	S	S	S	S						
		Transfer microplate to TM	S	S	D									
		Transfer microplate lid to TM	S	S	S									
		Cover TM lid	S	S	S	S	S	S						
		Transfer microplate lid to table	S	S	S									
		Transfer microplate to table	S	S	D									

Test result: 11th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 12th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
11	Glass pipet	Pick from shelf	S	S	S	S								
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Release liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put to shelf	D	S	S	S								

Test result: 13th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 14th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 15th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S		
		Unscrew cap from vial	S	S	S	S	S	S	S	S	S	F		
		Put cap to rack	S	S	S	S	S	S	S	S	S	S		
		Put vial to rack	S	S	S	S	S	S	S	S	S	S		

Test result: 16th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 17th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
11	Glass pipet	Pick from shelf	S	S	S	S								
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Release liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put to shelf	D	S	S	S								

Test result: 18th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 19th test about chiral compound

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12

Summary of failures in the 19 test records about chiral compound.

Type	Failed subtasks	Total number of tests	Number of failures	Failed in which test	Description	Solution

F	Load tip to pipet (300 μL 8-channel)	10 (10 x 1)	2	2nd, 9th	Eight tips should be loaded. But at least one tips were not loaded in the 2nd, 9th tests.	Adjust robot position of loading tips.
	Load tip to pipet (100 μL 8-channel)	10 (10 x 1)	1	10th	At least one tips were not loaded in the 10th test.	Same as above.
	Load tip to pipet (10 μL 8-channel)	140 (10 x 14)	2	6th, 7th	At least one tips were not loaded in the 6th and 7th tests.	Same as above.
	Transfer microplate from LC-tray to table	10 (10 x 1)	1	2nd	Failed to pick microplate from the tray.	Improve robot motions of gripping microplate.
	Transfer lid of microplate	30 (10 x 3)	3	3rd	Failed to grip lid of microplate, as the robber on the top of griper finger was shed.	Fix the robber.
	Release tip from pipet (10 μL 8-channel)	140 (10 x 14)	2	3rd, 4th	At least one tip was not released from the pipet.	Press the button further to release tips.
F_c	Transfer microplate from hotel to table	20 (10 x 2)	1	1st	Microplate was not placed correctly in hotel.	-
	Put tip box to table	60 (10 x 6)	2	3rd, 7th	Collisions happened when putting tip box to table, as the setting of robot was changed.	Recover robot settings.
	Load tip (10mL pipet)	20 (10 x 2)	4	3rd, 4th	Tip was not loaded correctly, as the tip box was not placed correctly.	-
D	Push out liquid (100 μL 8-channel pipet)	120 (10 x 12)	24	2nd, 9th	Liquid kept dropping when moving pipet, as tip was not fixed very well on the pipet.	Adjust the depth of inserting into tip.
	Push out liquid (1000 μL channel pipet)	200 (10 x 20)	80	3rd, 4th, 6th, 7th	Same as above.	Same as above.
	Pick vial from rack	100 (10 x 10)	2	3rd, 9th	Rack was dragged a little up.	Adjust robot position of picking vials.
	Load tip to pipet (300 μL 8-channel)	10 (10 x 1)	1	5th	Slight collision happened between pipet and tips when inserting pipet into tips.	Adjust robot position of loading tips.
	Load tip to pipet (10 μL 8-channel)	140 (10 x 14)	1	5th	Same as above.	Same as above.
	Load tip to pipet (100 μL)	60 (10 x 6)	2	10th	Same as above.	Same as above.
	Releasing glass pipet	40 (4 x 10)	2	10th	Glass pipet fell down from rack	-
	Transfer microplate between thermo shaker and table	30 (10 x 3)	1	10th	Microplate was gripped loosely.	Adjust gripper position.

Summary of failures in last nine of 19 test records about chiral compound.

Type	Failed subtasks	Total number of tests	Number of failures	Failed in which test	Description	Solution
F	Unscrew cap from vial	90 (9 x 10)	1	15th	Failed to remove cap from a vial, possibly because the robber on the top of right gripper finger was worn.	Fix new robber.

D	Releasing glass pipet	36 (9 x 4)	2	12th, 17th	Glass pipet fell down from rack	-

D.2 Records about Cholesterol Process

In a test record, four marks had been used to record the results, as following:

- S: The robot did a subtask **s**uccessfully.
- F: The robot was **f**ault to do a subtask.
- F_c: The robot was **f**ault to do a subtask, due to the wrong **c**onfiguration. Placing labware wrongly, changing settings of robot (e.g. tools or position variables) or changing the gripper fingers is possibly to cause the robot fault to do something.
- D: The robot did a subtask successfully in some degree, but not perfectly. So it was marked as "**d**efect". Improvements are required.

A complete record table of the 1st test was given. The parts with failures or "defects" in the record tables of rest tests were provided, and the parts with no failures or "defects" were not shown.

Test result: 1st test about cholesterol

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	S	S	S	D	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Unscrew cap from vial	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put cap to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
2	Close vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Pick cap from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Screw cap on vial	S	S	S	S	S	S	S	F_c	S	S	S	S
			S	S	S	S	S	S	S					
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
3	Transfer tip box	Pick box from shelf	S	S	S	S								
		Prepare to put to table	S	S	S	S								
		Put box to table	S	S	S	S								
		Pick box from table	S	S	S	S								
		Prepare to put to shelf	S	S	S	S								
		Put box to shelf	S	S	S	S								
4	1000uL Pipet	Pick from shelf	S	S										
		Load tip to pipet	S	S										
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Push out liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Release tip from pipet	S	S										
		Put pipet to shelf	S	S										
5	100uL Pipet	Pick from shelf	S	S	S									
		Load tip to pipet	S	S	S									
		Draw liquid	Total times: 128, S: 128, F: 0.											
		Push out liquid	Total times: 128, S: 128, F: 0.											
		Release tip from pipet	S	S	S									
		Put pipet to shelf	S	S	S									
6		Pick vial from rack	S	S	S	S	S	S	S					

	Handling small shaker	Put vial to shaker	S	S	S	S	S	S	S					
		Press button	S	S	S	S								
		Pick vial from shaker	S	S	S	S	S	S	S					
		Put vial to rack	S	S	S	S	S	S	S					
7	Handling ultrasonic machine	Pick vial from shaker	S	S	S	S	S	S	S					
		Put vial to bath	S	S	S	S	S	S	S					
		Pick vial from bath	S	S	S	S	S	S	S					
		Put vial to shaker	S	S	S	S	S	S	S					
		Turn knob	S											
8	Filtering liquid	Pick syringe from rack	S	S	S	S	S	S	S					
		Load cannula to syringe	S	S	S	S	S	S	S					
		Draw liquid	S	S	S	S	S	S	S					
		Release cannula from syringe	S	S	S	S	S	S	S					
		Load filter to syringe	S	S	S	S	S	S	S					
		Push out liquid	S	S	S	S	S	S	S					
		Release syringe	S	S	S	S	S	S	S					

Test result: 2nd test about cholesterol

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	S	S	S	D	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Unscrew cap from vial	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put cap to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
2	Close vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Pick cap from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Screw cap on vial	S	S	S	S	S	S	S	F	S	S	S	S
			S	S	S	S	S	S	S					
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
8	Filtering liquid	Pick syringe from rack	S	S	S	S	S	S	S					
		Load cannula to syringe	S	S	S	S	S	S	S					
		Draw liquid	S	F	F	S	S	S	S					
		Release cannula from syringe	S	S	S	S	S	S	S					
		Load filter to syringe	S	S	S	S	S	S	S					
		Push out liquid	S	S	F	S	S	S	S					
		Release syringe	S	S	S	S	S	S	S					

Test result: 3rd test about cholesterol

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	S	S	S	D	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Unscrew cap from vial	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put cap to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
2	Close vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Pick cap from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Screw cap on vial	S	S	S	S	S	F	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
8		Pick syringe from rack	S	S	S	S	S	S	S					

	Filtering liquid	Load cannula to syringe	S	S	S	S	S	S	S					
		Draw liquid	F	S	S	F	S	S	S					
		Release cannula from syringe	S	S	S	S	S	S	S					
		Load filter to syringe	S	S	S	S	S	S	S					
		Push out liquid	S	F	S	S	S	S	S					
		Release syringe	S	S	S	S	S	S	S					

Test result: 4th test about cholesterol

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	S	S	S	D	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Unscrew cap from vial	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put cap to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
2	Close vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Pick cap from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Screw cap on vial	S	S	S	S	S	S	S	S	F	S	S	S
			S	S	F	S	S	S	S					
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
8	Filtering liquid	Pick syringe from rack	S	S	S	S	S	S	S					
		Load cannula to syringe	S	S	S	S	S	S	S					
		Draw liquid	S	S	F	S	S	S	S					
		Release cannula from syringe	S	S	S	S	S	S	S					
		Load filter to syringe	S	S	S	S	S	S	S					
		Push out liquid	S	S	S	S	F	S	S					
		Release syringe	S	S	S	S	S	S	S					

Test result: 5th test about cholesterol

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	S	S	S	D	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Unscrew cap from vial	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put cap to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
2	Close vials	Pick vial from rack	S	S	D	D	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Pick cap from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Screw cap on vial	S	S	S	F	S	S	S	S	S	S	S	S
			S	S	F	S	S	S	S					
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
8	Filtering liquid	Pick syringe from rack	S	S	S	S	S	S	S					
		Load cannula to syringe	S	S	S	S	S	S	S					
		Draw liquid	S	S	F	S	S	S	S					
		Release cannula from syringe	S	S	S	S	S	S	S					
		Load filter to syringe	S	S	S	S	S	S	S					
		Push out liquid	S	S	S	S	F	S	S					
		Release syringe	S	S	S	S	S	S	S					

Test result: 6th test about cholesterol

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12

1	Open vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Unscrew cap from vial	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put cap to rack	S	S	S	S	S	F	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
2	Close vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Pick cap from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Screw cap on vial	S	S	S	F	S	S	S	S	S	S	S	S
			S	F	S	S	S	S	S					
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					

Test result: 7th test about cholesterol

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	F	S	S	S	S	F	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Unscrew cap from vial	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put cap to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
2	Close vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Pick cap from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Screw cap on vial	S	S	F	F	F	S	S	F	S	S	S	S
			S	F	S	S	S	S	S					
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
4	1000uL Pipet	Pick from shelf	S	S										
		Load tip to pipet	S	S										
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Push out liquid	S	S	D	D	S	D	S	S	D	S	S	S
			S	D	S	S	S	S	S					
		Release tip from pipet	S	S										
		Put pipet to shelf	S	S										

Test result: 8th test about cholesterol

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Unscrew cap from vial	F	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put cap to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
8	Filtering liquid	Pick syringe from rack	S	S	S	S	S	S	S					
		Load cannula to syringe	S	S	S	S	S	S	S					
		Draw liquid	S	F	F	S	F	S	S					
		Release cannula from syringe	S	S	S	S	S	S	S					
		Load filter to syringe	S	S	S	S	S	S	S					
		Push out liquid	S	S	S	S	S	S	S					
		Release syringe	S	S	S	S	S	S	S					

Test result: 9th test about cholesterol

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Unscrew cap from vial	S	S	F	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put cap to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S

Test result: 10th test about cholesterol

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Unscrew cap from vial	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put cap to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	F	S	F	F	F	S	S
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
2	Close vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Pick cap from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Screw cap on vial	S	S	F	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
4	1000uL Pipet	Pick from shelf	S	S										
		Load tip to pipet	S	S										
		Draw liquid	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Push out liquid	S	S	S	S	D	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Release tip from pipet	S	S										
		Put pipet to shelf	S	S										
8	Filtering liquid	Pick syringe from rack	S	S	S	S	S	S	S					
		Load cannula to syringe	S	S	S	S	S	S	S					
		Draw liquid	F	S	S	S	S	S	S					
		Release cannula from syringe	S	S	S	S	S	S	S					
		Load filter to syringe	S	S	S	S	S	S	S					
		Push out liquid	S	S	S	S	S	S	S					
		Release syringe	S	S	S	S	S	S	S					

Test result: 11th test about cholesterol

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 12th test about cholesterol

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Unscrew cap from vial	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put cap to rack	S	S	S	S	S	S	S	S	F	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S	S	S	S	S	S
8	Filtering liquid	Pick syringe from rack	S	S	S	S	S	S	S					
		Load cannula to syringe	S	S	S	S	S	S	S					

		Draw liquid	S	S	F	S	S	S	S					
		Release cannula from syringe	S	S	S	S	S	S	S					
		Load filter to syringe	S	S	S	S	S	S	S					
		Push out liquid	S	S	S	S	S	S	S					
		Release syringe	S	S	S	S	S	S	S					

Test result: 13th test about cholesterol

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 14th test about cholesterol

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
2	Close vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Pick cap from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Screw cap on vial	S	S	F	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					

Test result: 15th test about cholesterol

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
8	Filtering liquid	Pick syringe from rack	S	S	S	S	S	S	S					
		Load cannula to syringe	S	S	S	S	S	S	S					
		Draw liquid	S	S	S	S	S	S	S					
		Release cannula from syringe	S	S	S	S	S	S	S					
		Load filter to syringe	S	S	S	S	S	S	S					
		Push out liquid	S	S	S	F	S	S	S					
		Release syringe	S	S	S	S	S	S	S					

Test result: 16th test about cholesterol

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 17th test about cholesterol

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 18th test about cholesterol

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12
1	Open vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Unscrew cap from vial	S	S	S	S	S	S	S	S	F	S	S	S
			S	S	S	S	S	S	S					
		Put cap to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
2	Close vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Pick cap from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Screw cap on vial	S	S	S	S	F	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					

Test result: 19th test about cholesterol

	Parts	**Key subtask**	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 20th test about cholesterol

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12
2	Close vials	Pick vial from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Pick cap from rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					
		Screw cap on vial	S	S	S	S	S	S	S	S	S	F	S	S
			S	S	S	S	S	S	S					
		Put vial to rack	S	S	S	S	S	S	S	S	S	S	S	S
			S	S	S	S	S	S	S					

Test result: 21th test about cholesterol

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 22th test about cholesterol

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 23th test about cholesterol

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 24th test about cholesterol

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12

Test result: 29th test about cholesterol

	Parts	Key subtask	1	2	3	4	5	6	7	8	9	10	11	12
8	Filtering liquid	Pick syringe from rack	S	S	S	S	S	S	S					
		Load cannula to syringe	S	S	S	S	S	S	S					
		Draw liquid	S	S	S	S	S	S	S					
		Release cannula from syringe	S	S	S	S	S	S	S					
		Load filter to syringe	S	S	S	S	S	S	S					
		Push out liquid	S	S	S	S	S	F	S					
		Release syringe	S	S	S	S	S	S	S					

Summary of failures in first 13 of 36 test records about cholesterol.

Type	Failed subtasks	Total number of tests	Number of failures	Failed in which test	Description	Solution
F	Screw cap on vial	247 (13 x 19)	14	2nd, 3rd, 4th, 5th, 6th, 7th, 10th	Failed to screw cap on vial, with cap not screwed correctly or fell down from vial.	-
	Draw liquid (syringe)	91 (13 x 7)	10	2nd, 3rd, 4th, 5th, 8th, 10th	Collision happened between cannula and vial, when moving cannula into a vial to draw liquid.	Change the gripping position at syringe, more detail can be seen at section 11.1.2.5.
	Push out liquid (syringe)	91 (13 x 7)	4	2nd, 3rd, 4th, 5th	Collision happened between filter and vial.	Adjust robot position of moving filter into vial.
	Put cap to rack	247 (13 x 19)	6	6th, 10th, 12th	Cap was not put on the rack correctly.	Adjust robot position of putting cap on the rack.
	Pick vial from rack	247 (13 x 19)	2	7th	Rack was dragged from the ALP completely.	Adjust robot position of picking vials.
	Unscrew cap from vial	247 (13 x 19)	2	8th, 9th	If the cap was screwed too tightly on the vial, the robot will fail to unscrew it.	Avoid screwing the cap too tightly.
F_c	Screw cap on vial	247 (13 x 19)	1	1st	A wrong SAMI method was used.	-

	Put cap to rack	247 (13 x 19)	1	7th	This failure was due to a bugger of a related robot job, which was generated when expanding this job to handling new type of vials.	The bugger was fixed.
	Draw liquid (syringe)	91 (13 x 7)	1	12th	That syringe became invalid after reused several times. It was hard to drag its rod out, causing the syringe was dragged out from the gripper.	Syringes were reused in some tests. But in real applications, a syringe will be used one time. New syringe has no such problem.
D	Pick vial from rack	247 (13 x 19)	7	1st, 2nd, 3rd, 4th, 5th	Rack was dragged a little up.	Adjust robot position of picking vials.
	Push out liquid (1mL pipet)	247 (13 x 19)	6	7th, 10th	Liquid kept dropping when moving pipet, as tip was not fixed very well on the pipet.	Adjust the depth of inserting into tip.

Summary of failures in last 23 of 26 test records about cholesterol.

Type	Failed subtasks	Total number of tests	Number of failures	Failed in which test	Description	Solution
F	Screw cap on vial	437 (23 x 19)	3	14th, 18th, 20th	Cap was sloped.	-
	Push out liquid (syringe)	161 (23 x 7)	2	15th, 29th	Collision happened between filter and vial. Failed to grip a syringe.	As discussed in section 11.1.2.5.
	Unscrew cap from vial	437 (23 x 19)	1	18th	Cap was screwed too tightly on the vial.	Avoid screw the cap too tightly.

References

[1] Hitomi, K., Automation - its concept and a short history. *Technovation* 1994, *14*, 121–128.

[2] Parasuraman, R., Riley, V., Humans and automation: Use, misuse, disuse, abuse. *Human Factors* 1997, *39*, 230–253.

[3] Goldberg, K., What is automation? *IEEE Transactions on Automation Science and Engineering* 2012, *9*, 1–2.

[4] Zhang, M., Felder, R. A., Kim, E. S., Nelson, B. et al., Editorial Special Issue on Life Science Automation. *Automation Science and Engineering* 2006, *3*, 137–140.

[5] Kong, F., Yuan, L., Zheng, Y. F., Chen, W., Automatic Liquid Handling for Life Science: A Critical Review of the Current State of the Art. *Journal of Laboratory Automation* 2012, *17*, 169–185.

[6] Michael, S., Auld, D., Klumpp, C., Jadhav, A. et al., A Robotic Platform for Quantitative High-Throughput Screening. *ASSAY and Drug Development Technologies* 2008, *6*, 637–657.

[7] South, S. F., Casina, T. S., Li, L., Exponential error reduction in pretransfusion testing with automation. *Transfusion* 2012, *52*, 81S-87S.

[8] Drotning, W., Wapman, W., Fahrenholtz, J., Kimberly, H. et al., System design for safe robotic handling of nuclear materials. *ASCE Specialty Conference, Proceedings* 1996, 241–247.

[9] Puccinelli, J. P., Su, X., Beebe, D. J., Automated High-Throughput Microchannel Assays for Cell Biology: Operational Optimization and Characterization. *JALA - Journal of the Association for Laboratory Automation* 2010, *15*, 25–32.

[10] Bogue, R., Robots in the laboratory: a review of applications. *Industrial Robot: An International Journal* 2012, *39*, 113–119.

[11] Wood M.D., Franchetti J.A., Laboratory automation using robotics and information management systems. *Current Opinion in Biotechnology* 1993, *4*, 91–94.

[12] Gallert, C., Lehmann, R., Roddelkopf, T., Junginger, S. et al., High Throughput Screening System for screening of 3D cell cultures. *2015 IEEE I2MTC - International Instrumentation and Measurement Technology Conference, Proceedings* 2015, 1302–1307.

[13] Sun, H., Xia, M., Austin, C. P., Huang, R., Paradigm Shift in Toxicity Testing and Modeling. *The AAPS Journal* 2012, *14*, 473–480.

[14] Heidari Khajepour, M. Y., Vernede, X., Cobessi, D., Lebrette, H. et al., REACH: Robotic Equipment for Automated Crystal Harvesting using a six-axis robot arm and a micro-gripper. *Acta Crystallographica Section D* 2013, *69*, 381–387.

[15] Vogt, G., Multi-axis robots bring automation to life sciences. *Industrial Robot: An International Journal* 2002, *29*, 49–52.

[16] Moore, K. W., Newman, R., Chan, G. K., Leech, C. et al., Implementation of a High Specification Dual-Arm Robotic Platform to Meet Flexible Screening Needs. *Journal of the Association for Laboratory Automation* 2007, *12*, 115–123.

[17] Smith, C., Karayiannidis, Y., Nalpantidis, L., Gratal, X. et al., Dual arm manipulation—A survey. *Robotics and Autonomous Systems* 2012, *60*, 1340–1353.

[18] Park, C., Park, K., Kim, D., Design of dual arm robot manipulator for precision assembly of mechanical parts. *ICSMA 2008 - International Conference on Smart Manufacturing Application* 2008, 424–427.

[19] Zanchettin, A. M., Bascetta, L., Rocco, P., Acceptability of robotic manipulators in shared working environments through human-like redundancy resolution. *Applied Ergonomics* 2013, *44*, 982–989.

[20] Kock, S., Vittor, T., Matthias, B., Jerregard, H. et al., Robot concept for scalable, flexible assembly automation: A technology study on a harmless dual-armed robot. *Proceedings - 2011 IEEE International Symposium on Assembly and Manufacturing* 2011.

[21] Do, H. M., Park, C., Choi, T. Y., Kyung, J. H., Design and control of dual-arm robot for cell manufacturing process. *2013 IEEE International Conference on Mechatronics and Automation* 2013, 1419–1423.

[22] Li, M., Automation in the bioanalytical laboratory: what is the future? *Bioanalysis* 2013, *5*, 2859–2861.

[23] Kaber, D. B., Stoll, N., Thurow, K., Human-automation Interaction Strategies for Life Science Applications: Implications and Future Research. *IEEE Conference on Automation Science and Engineering* 2007, 615–620.

[24] Bilitewski, U., Protein-sensing assay formats and devices. *Analytica Chimica Acta* 2006, *568*, 232–247.

[25] Kronholm J., Hartonen K., Riekkola M.-L., Analytical extractions with water at elevated temperatures and pressures. *TrAC - Trends in Analytical Chemistry* 2007, *26*, 396–412.

[26] Ding Y., Zhang W., Gu C., Xagoraraki I. et al., Determination of pharmaceuticals in biosolids using accelerated solvent extraction and liquid chromatography/tandem mass spectrometry. *Journal of Chromatography A* 2011, *1218*, 10–16.

[27] Fleischer H., Vorberg E., Thurow K., Warkentin M. et al., Determination of calcium and phosphor in bones using microwave digestion and ICP-MS: Comparison of manual and automated methods using ICP-MS. *5th IMEKO TC19 Symposium on Environmental Instrumentation and Measurements* 2014, 94–99.

[28] Lucinda, C. H., Surrendering to the robot army: why we resist automation in drug discovery and development. *Bioanalysis* 2012, *4*, 985–987.

[29] Phillips, K., McCallum, N., Welch, L., A comparison of methods for forensic DNA extraction: Chelex-100 and the QIAGEN DNA Investigator Kit (manual and automated). *Forensic Science International: Genetics* 2012, *6*, 282–285.

[30] Gaisford, W., Robotic Liquid Handling and Automation in Epigenetics. *Journal of Laboratory Automation* 2012, *17*, 327–329.

[31] Pereira, D. A., Williams, J. A., Origin and evolution of high throughput screening. *British Journal of Pharmacology* 2007, *152*, 53–61.

[32] Hertzberg, R. P., Pope, A. J., High-throughput screening: new technology for the 21st century. *Current Opinion in Chemical Biology* 2000, *4*, 445–451.

[33] Mayr, L. M., Fuerst, P., The Future of High-Throughput Screening. *Journal of Biomolecular Screening* 2008, *13*, 443–448.

[34] American National Standards Institute, Society for Laboratory Automation and Screening, Microplate Standards [Internet] [cited 25-12-15]. Available from: http://www.slas.org/resources/information/industry-standards/.

[35] Liu Y., Yao Y., Chen L., Robotic liquid handling system for microdispensing of highly viscous reagent. *Proceedings - 3rd International Conference on Measuring Technology and Mechatronics Automation, ICMTMA* 2011, *3*, 171–174.

[36] Uematsu C., Makino I., Matsunaga T., Harada K., An automated liquid handling system for polymerase chain reaction sample preparation. *Journal of Bioscience and Bioengineering* 2004, *97*, 432–435.

[37] Support - Beckman Coulter, Inc [Internet] [cited 29-12-15]. Available from: https://www.beckmancoulter.com/wsrportal/WSR/research-and-discovery/products-and-services/research-automation/index.htm.

[38] Liu H., Stoll N., Junginger S., Thurow K., A fast method for mobile robot transportation in life science automation. *Conference Record - IEEE Instrumentation and Measurement Technology Conference* 2013, 238–242.

[39] Lorenz, M. G. O., Liquid-Handling Robotic Workstations for Functional Genomics. *Journal of Laboratory Automation* 2004, *9*, 262–267.

[40] Dunn, D. A., Feygin, I., Challenges and solutions to ultra-high-throughput screening assay miniaturization: submicroliter fluid handling. *Drug Discovery Today* 2000, *5*, 84–91.

[41] Deller M.C., Rupp B., Approaches to automated protein crystal harvesting. *Acta Crystallographica Section F:Structural Biology Communications* 2014, *70*, 133–155.

[42] Viola R., Carman P., Walsh J., Miller E. et al., Operator-assisted harvesting of protein crystals using a universal micromanipulation robot. *Journal of Applied Crystallography* 2007, *40*, 539–545.

[43] Tung H.-W., Sargent D.F., Nelson B.J., Protein crystal harvesting using the RodBot: A wireless mobile microrobot. *Journal of Applied Crystallography* 2014, *47*, 692–700.

[44] Collins, F. S., Gray, G. M., Bucher, J. R., Transforming Environmental Health Protection. *Science (New York, N.Y.)* 2008, *319*, 906–907.

[45] Andersen, M. E., Al-Zoughool, M., Croteau, M., Westphal, M. et al., The future of toxicity testing. *Journal of Toxicology and Environmental Health - Part B: Critical Reviews* 2010, *13*, 163–196.

[46] Attene-Ramos, M. S., Miller, N., Huang, R., Michael, S. et al., The Tox21 robotic platform for the assessment of environmental chemicals - From vision to reality. *Drug Discovery Today* 2013, *18*, 716–723.

[47] Philippeos C., Hughes R.D., Dhawan A., Mitry R.R., Introduction to cell culture. *Methods in Molecular Biology* 2012, *806*, 1–13.

[48] Triaud, F., Clenet, D., Cariou, Y., Le Neel, T. et al., Evaluation of automated cell culture incubators. *JALA - Journal of the Association for Laboratory Automation* 2003, *8*, 82–86.

[49] Thomas R.J., Chandra A., Hourd P.C., Williams D.J., Cell Culture Automation and Quality Engineering: A Necessary Partnership to Develop Optimized Manufacturing Processes for Cell-Based Therapies. *JALA - Journal of the Association for Laboratory Automation* 2008, *13*, 152–158.

[50] Kempner, M. E., Felder, R. A., A Review of Cell Culture Automation. *Journal of Laboratory Automation* 2002, *7*, 56–62.

[51] Bernard C.J., Connors D., Barber L., Jayachandra S. et al., Adjunct automation to the cellmate™ cell culture robot. *JALA - Journal of the Association for Laboratory Automation* 2004, *9*, 209–217.

[52] Soloveva, V., LaRocque, J., McKillip, E., When Robots are Good: Fully Automated Thermo LAS Robotic Assay System with Dual FLIPRTETRA and TAP SelecT Robotic Cell Culture System. *Journal of the Association for Laboratory Automation* 2006, *11*, 145–156.

[53] Gallert, C., Lehmann, R., Roddelkopf, T., Junginger, S. et al., Biological high throughput screening of 2D and 3D cell cultures for future industrial up-scaling. *IEEE International Conference on Automation Science and Engineering* 2015, 1527–1532.

[54] Lob W.S., Robotic transportation. *Clinical Chemistry* 1990, *36*, 1544–1550.

[55] DeSouza G.N., Kak A.C., Vision for mobile robot navigation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2002, *24*, 237–267.

[56] Liu H., Stoll N., Junginger S., Thurow K., Mobile robot for life science automation. *International Journal of Advanced Robotic Systems* 2013, *10*, 1–14.

[57] Liu H., Stoll N., Junginger S., Thurow K., A common wireless remote control system for mobile robots in laboratory. *IEEE I2MTC - International Instrumentation and Measurement Technology Conference, Proceedings* 2012, 688–693.

[58] Liu H., Stoll N., Junginger S., Thurow K., An application of charging management for mobile robot transportation in laboratory environments. *Conference Record - IEEE Instrumentation and Measurement Technology Conference* 2013, 435–439.

[59] Abdulla A.A., Liu H., Stoll N., Thurow K., Multi-floor navigation method for mobile robot transportation based on StarGazer sensors in life science automation. *Conference Record - IEEE Instrumentation and Measurement Technology Conference* 2015, 428–433.

[60] King R.D., Whelan K.E., Jones F.M., Reiser P.G.K. et al., Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature* 2004, *427*, 247–252.

[61] Larisa, S. N., Clare,, A., Sparkes, A., King, R. D., An ontology for a Robot Scientist. *Bioinformatics* 2006, *22*, e464-e471.

[62] Liakata, M., Rowland, J., Aubrey, W., Markham, M. et al., The Robot Scientist Adam. *Computer* 2009, *42*, 46–54.

[63] King, R. D., Rowland, J., Oliver, S. G., Young, M. et al., The Automation of Science. *Science* 2009, *324*, 85–89.

[64] Qi D., King R.D., Hopkins A.L., Bickerton G.R. et al., An ontology for description of drug discovery investigations. *Journal of integrative bioinformatics* 2010, *7*.

[65] Williams K., Bilsland E., Sparkes A., Aubrey W. et al., Cheaper faster drug development validated by the repositioning of drugs against neglected tropical diseases. *Journal of the Royal Society Interface* 2015, *12*.

[66] Han C.-S., "Human-robot cooperation technology" an ideal midway solution heading toward the future of robotics and automation in construction. *Proceedings of the 28th International Symposium on Automation and Robotics in Construction* 2011, 13–18.

[67] Goetz J., Kiesler S., Powers A., Matching robot appearance and behavior to tasks to improve human-robot cooperation. *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication* 2003, 55–60.

[68] Connolly, C., Motoman markets co-operative and humanoid industrial robots. *Industrial Robot: An International Journal* 2009, *36*, 417–420.

[69] YASKAWA Europe GmbH [Internet]. Available from: http://www.motoman.eu/products/controllers-for-robots/?no_cache=1.

[70] Bogue, R., The role of robots in the food industry: a review. *Industrial Robot: An International Journal* 2009, *36*, 531–536.

[71] Bloss, R., Innovations like two arms, cheaper prices, easier programming, autonomous and collaborative operation are driving automation deployment in manufacturing and elsewhere. *Assembly Automation* 2013, *33*, 312–316.

[72] Kusuda, Y., The use of robots in the Japanese food industry. *Industrial Robot: An International Journal* 2010, *37*, 503–508.

[73] Kruse, D., Radke, R. J., Wen, J. T., A sensor-based dual-arm tele-robotic manipulation platform. *IEEE International Conference on Automation Science and Engineering* 2013, 350–355.

[74] Ding H., Schipper M., Matthias B., Collaborative behavior design of industrial robots for multiple human-robot collaboration. *2013 44th International Symposium on Robotics* 2013.

[75] Ding, H., Heyn, J., Matthias, B., Staab, H., Structured collaborative behavior of industrial robots in mixed human-robot environments. *IEEE International Conference on Automation Science and Engineering* 2013, 1101–1106.

[76] Polverini M.P., Zanchettin A.M., Rocco P., Real-time collision avoidance in human-robot interaction based on kinetostatic safety field. *IEEE International Conference on Intelligent Robots and Systems* 2014, 4136–4141.

[77] Ding H., Schipper M., Matthias B., Optimized task distribution for industrial assembly in mixed human-robot environments - Case study on IO module assembly. *IEEE International Conference on Automation Science and Engineering* 2014, 19–24.

[78] Albers, A., Brudniok, S., Ottnad, J., Sauter, C. et al., Upper Body of a new Humanoid Robot - the Design of ARMAR III. *Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots* 2006, 308–313.

[79] Asfour T., Regenstein K., Azad P., Schröder J. et al., ARMAR-III: An integrated humanoid platform for sensory-motor control. *Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS* 2006, 169–175.

[80] Vahrenkamp N., Berenson D., Asfour T., Kuffner J. et al., Humanoid motion planning for dual-arm manipulation and re-grasping tasks. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems* 2009, 2464–2470.

[81] Vahrenkamp, N., Boge, C., Welke, K., Asfour, T. et al., Visual servoing for dual arm motions on a humanoid robot. *IEEE-RAS International Conference on Humanoid Robots* 2009, 208–214.

[82] Ott, C., Eiberger, O., Friedl, W., Bauml, B. et al., A Humanoid Two-Arm System for Dexterous Manipulation. *Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots* 2006, 276–283.

[83] Albu-Schäffer, A., Ott, C., Hirzinger, G., A Unified Passivity-based Control Framework for Position, Torque and Impedance Control of Flexible Joint Robots. *The International Journal of Robotics Research* 2007, *26*, 23–39.

[84] Butterfass, J., Grebenstein, M., Liu, H., Hirzinger, G., DLR-Hand II: next generation of a dextrous robot hand. *Proceedings - IEEE International Conference on Robotics and Automation* 2001, *1*, 109–114.

[85] Borst C., Wimböck T., Schmidt F., Fuchs M. et al., Rollin' Justin - Mobile platform with variable base. *Proceedings - IEEE International Conference on Robotics and Automation* 2009, 1597–1598.

[86] Fuchs M., Borst Ch., Giordano P.R., Baumann A. et al., Rollin' Justin - Design considerations and realization of a mobile platform for a humanoid upper body. *Proceedings - IEEE International Conference on Robotics and Automation* 2009, 4131–4137.

[87] Wimböck T., Nenchev D., Albu-Schaffer A., Hirzinger G., Experimental study on dynamic reactionless motions with DLR's humanoid robot Justin. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems* 2009, 5481–5486.

[88] Birbach O., Frese U., Bäuml B., Realtime Perception for Catching a Flying Ball with a Mobile Humanoid. *Proceedings - IEEE International Conference on Robotics and Automation* 2011, 5955–5962.

[89] Do, H. M., Park, C., Kyung, J. H., Dual arm robot for packaging and assembling of IT products. *IEEE International Conference on Automation Science and Engineering* 2012, 1067–1070.

[90] Choi, T., Do, H., Park, K. T., Kim, D. et al., Small sized industrial dual-arm robot with convenient program interface. *2013 44th International Symposium on Robotics* 2013.

[91] Park, D. I., Park, C., Do, H., Choi, T. et al., Development of dual arm robot platform for automatic assembly. *International Conference on Control, Automation and Systems* 2014, 319–321.

[92] Do, H. M., Choi, T.-Y., Kyung, J. H., Automation of cell production system for cellular phones using dual-arm robots. *International Journal of Advanced Manufacturing Technology* 2015, *83*, 1349–1360.

[93] Park, K.-T., Park, C.-H., Shin, Y.-J., Performance Evaluation of Industrial Dual-Arm Robot. *ICSMA 2008 - International Conference on Smart Manufacturing Application* 2008, 437–440.

[94] Chanhun Park, KyoungTaik Park, Dong-Il Park, Jin-Ho Kyung, Dual arm robot manipulator and its easy teaching system. *IEEE International Symposium on Assembly and Manufacturing* 2009, 242–247.

[95] Bluethmann, W., Ambrose, R., Diftler, M., Askew, S. et al., Robonaut: A robot designed to work with humans in space. *Autonomous Robots* 2003, *14*, 179–197.

[96] Ambrose, R. O., Aldridge, H., Askew, R. S., Burridge, R. R. et al., Robonaut: NASA's space humanoid. *IEEE Intelligent Systems and Their Applications* 2000, *15*, 57–62.

[97] Lovchik, C. S., Diftler, M. A., Robonaut hand: a dextrous robot hand for space. *Proceedings - IEEE International Conference on Robotics and Automation* 1999, *2*, 907–912.

[98] Rehnmark, F., Bluethmann, W., Mehling, J., Ambrose, R. O. et al., Robonaut: The 'short list' of technology hurdles. *Computer* 2005, *38*, 28–37.

[99] Diftler, M. A., Mehling, J. S., Abdallah, M. E., Radford, N. A. et al., Robonaut 2 - The first humanoid robot in space. *Proceedings - IEEE International Conference on Robotics and Automation* 2011, 2178–2183.

[100] Diftler, M. A., Ahlstrom, T. D., Ambrose, R. O., Radford, N. A. et al., Robonaut 2 — Initial activities on-board the ISS. *IEEE Aerospace Conference Proceedings* 2012.

[101] Mukai, T., Hirano, S., Nakashima, H., Kato, Y. et al., Development of a nursing-care assistant robot RIBA that can lift a human in its arms. *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems* 2010, 5996–6001.

[102] Mukai, T., Hirano, S., Yoshida, M., Nakashima, H. et al., Whole-body contact manipulation using tactile information for the nursing-care assistant robot RIBA. *IEEE International Conference on Intelligent Robots and Systems* 2011, 2445–2451.

[103] Ding, M., Ikeura, R., Mukai, T., Nagashima, H. et al., Comfort estimation during lift-up using nursing-care robot - RIBA. *Proceedings of the 2012 1st International Conference on Innovative Engineering Systems* 2012, 225–230.

[104] Matsubara, T., Shinohara, D., Kidode, M., Reinforcement learning of a motor skill for wearing a T-shirt using topology coordinates. *Advanced Robotics* 2013, *27*, 513–524.

[105] Tamei, T., Matsubara, T., Rai, A., Shibata, T., Reinforcement learning of clothing assistance with a dual-arm robot. *IEEE-RAS International Conference on Humanoid Robots* 2011, 733–738.

[106] Hynes, P., Dodds, G. I., Wilkinson, A. J., Uncalibrated visual-servoing of a dual-arm robot for MIS suturing. *Proceedings of the First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics* 2006, 420–425.

[107] Simaan, N., Xu, K., Wei, W., Kapoor, A. et al., Design and Integration of a Telerobotic System for Minimally Invasive Surgery of the Throat. *International Journal of Robotics Research* 2009, *28*, 1134–1153.

[108] Haidegger, T., Benyo, Z., Surgical robotic support for long duration space missions. *Acta Astronautica* 2008, *63*, 996–1005.

[109] Wu, H., Guo, J., Chen, S., Liu, X. et al., Recent developments in qualitative and quantitative analysis of phytochemical constituents and their metabolites using liquid chromatography–mass spectrometry. *Journal of Pharmaceutical and Biomedical Analysis* 2013, *72*, 267–291.

[110] Patel, R. S., Roy, M., Dutta, G. K., Mass spectrometry- a review. *Veterinary World* 2012, *5*, 185–192.

[111] Cuyckens, F., Claeys, M., Mass spectrometry in the structural analysis of flavonoids. *Journal of Mass Spectrometry* 2004, *39*, 1–15.

[112] Lewen, N., Mathew, S., Schenkenberger, M., Raglione, T., A rapid ICP-MS screen for heavy metals in pharmaceutical compounds. *Journal of Pharmaceutical and Biomedical Analysis* 2004, *35*, 739–752.

[113] Tainer, J. A., Roberts, V. A., Getzoff, E. D., Metal-binding sites in proteins. *Current Opinion in Biotechnology* 1991, *2*, 582–591.

[114] Hann, S., Dernovics, M., Koellensperger, G., Elemental analysis in biotechnology. *Analytical Biotechnology* 2015, *31*, 93–100.

[115] Ryszard, L., Sabine, B. J., Hiroki, H., Bibundhendra, S., Metallomics: Guidelines for terminology and critical evaluation of analytical chemistry approaches (IUPAC Technical Report). *Pure and Applied Chemistry* 2010, *82*, 493–504.

[116] Morlacchi, S., Pennati, G., Petrini, L., Dubini, G. et al., Influence of plaque calcifications on coronary stent fracture: A numerical fatigue life analysis including cardiac wall movement. *Journal of Biomechanics* 2014, *47*, 899–907.

[117] Zhao, S., Gu, L., Froemming, S. R., Finite Element Analysis of the Implantation of a Self-Expanding Stent: Impact of Lesion Calcification. *Journal of Medical Devices, Transactions of the ASME* 2012, *6*.

[118] Żabiński, J., Maciejewska, D., Kaźmierczak, P., Structural analysis of bis-amidines and bis-nitriles in solid-state by combining 5NMR6 spectroscopy and molecular modeling. *Journal of Molecular Structure* 2009, *923*, 132–140.

[119] Wu, L., Vogt, F. G., A review of recent advances in mass spectrometric methods for gas-phase chiral analysis of pharmaceutical and biological compounds. *Journal of Pharmaceutical and Biomedical Analysis* 2012, *69*, 133–147.

[120] Smith, S. W., Chiral toxicology: It's the same thing only different. *Toxicological Sciences* 2009, *110*, 4–30.

[121] Fleischer, H., Thurow, K., Innovative software solution for special data evaluation in mass spectrometry. *Conference Record - IEEE Instrumentation and Measurement Technology Conference* 2013, 1624–1629.

[122] Fleischer, H., Thurow, K., Fast mass spectrometry-based enantiomeric excess determination of proteinogenic amino acids. *Amino Acids* 2013, *44*, 1039–1051.

[123] Maxfield, F. R., Tabas, I., Role of cholesterol and lipid organization in disease. *Nature* 2005, *438*, 612–621.

[124] Kratzer, W., Mason, R. A., Kächele, V., Prevalence of gallstones in sonographic surveys worldwide. *J. Clin. Ultrasound* 1999, *27*, 1–7.

[125] Marschall, H.-U., Einarsson, C., Gallstone disease. *Journal of Internal Medicine* 2007, *261*, 529–542.

[126] Dowidar, N., Hj, K., Lyon, H., Matzen, P., Clogging of Biliary Endoprostheses: A Morphologic and Bacteriologic Study. *Scandinavian Journal of Gastroenterology* 1991, *26*, 1137–1144.

[127] Cetta, F., Rappuoli, R., Montalto, G., Baldi, C. et al., New biliary endoprosthesis less liable to block in biliary infections: Description and in vitro studies. *European Journal of Surgery* 1999, *165*, 782–785.

[128] Chu, X., Fleischer, H., Roddelkopf, T., Stoll, N. et al., Automated sample preparation using a dual-arm robotic platform. *American Laboratory* 2016, *48*, 44–45.

[129] Support - Beckman Coulter, Inc [Internet] [cited 17-2-15]. Available from: https://www.beckmancoulter.com/wsrportal/WSR/research-and-discovery/index.htm.

[130] Chu, X., Fleischer, H., Stoll, N., Klos, M. et al., Application of Dual-arm Robot in Biomedical Analysis: Sample Preparation and Transport. *International Instrumentation and Measurement Technology Conference (I2MTC)* 2015, 500–504.

[131] Lin, J.-M., Hong, Z.-W., Fang, G.-M., Lee, C.-T., A style for integrating MS-Windows software applications to client–server systems using Java technology. *Software: Practice and Experience* 2007, *37*, 417–440.

[132] Lutz, M. H., Laplante, P. A., C# and the .NET Framework: Ready for Real Time? *IEEE Software* 2003, *20*, 74–80.

[133] Sneed, H. M., Using XML to integrate existing software systems into the Web. *Proceedings - IEEE Computer Society's International Computer Software and Applications Conference* 2002, 167–172.

[134] Chu, X., Fleischer, H., Roddelkopf, T., Stoll, N. et al., A LC-MS Integration Approach in Life Science Automation: Hardware Integration and Software Integration. *IEEE International Conference on Automation Science and Engineering* 2015, 979–984.

[135] Chu, X., Roddelkopf, T., Fleischer, H., Stoll, N. et al., Flexible Robot Platform for Sample Preparation Automation with a User-friendly Interface. *IEEE International Conference on Robotics and Biomimetics* 2016, accepted.

Declaration

This dissertation ‘Automation Strategies for Sample Preparation in Life Science Applications’ is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions. The work of this dissertation has been done by me under the guidance of Prof. Dr. -Ing. habil. Kerstin Thurow and Prof. Dr. -Ing. Norbert Stoll, at University of Rostock, Germany. Also, the dissertation has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Xianghua Chu

Rostock, 03 November, 2016

List of Publications

Publications:

1. Chu, X., Fleischer, H., Stoll, N., Klos, M., Thurow, K., Application of Dual-arm Robot in Biomedical Analysis: Sample Preparation and Transport. International Instrumentation and Measurement Technology Conference (I2MTC) 2015, 500–504. DOI: 10.1109/I2MTC.2015.7151318
2. Chu, X., Fleischer, H., Roddelkopf, T., Stoll, N., Klos, M., Thurow, K., A LC-MS Integration Approach in Life Science Automation: Hardware Integration and Software Integration. IEEE International Conference on Automation Science and Engineering (CASE) 2015, 979–984. DOI: 10.1109/CoASE.2015.7294226
3. Chu, X., Fleischer, H., Roddelkopf, T., Stoll, N., Klos, M., Thurow, K., Automated sample preparation using a dual-arm robotic platform. American Laboratory 2016, 48, 44–45.
4. Chu, X., Roddelkopf, T., Fleischer, H., Stoll, N., Klos, M., Thurow, K., Flexible Robot Platform for Sample Preparation Automation with a User-friendly Interface. IEEE International Conference on Robotics and Biomimetics (ROBIO) 2016, accepted.

Oral presentations:

1. Chu, X., “A LC-MS Integration Approach in Life Science Automation: Hardware Integration and Software Integration”, IEEE International Conference on Automation Science and Engineering (CASE), Gothenburg; Sweden, 26-August-2015.
2. Chu, X., “Flexible Robot Platform for Sample Preparation Automation with a User-friendly Interface”, IEEE International Conference on Robotics and Biomimetics (ROBIO), Qingdao, China, 06-Dec.-2016.

Theses

1. An automated platform based on an industrial dual-arm robot is developed for automating sample preparation in life science applications.
2. A variety of tools are supported by the dual-arm robot to prepare samples. These tools are the same ones with that used by manual, including pipettes, syringes, glass pipettes and so on.
3. The manual processes of sample preparation are transferred to automated procedures by the dual-arm robot platform without major changes, including the sample preparing process of chiral compounds and the preparing process of cholesterol samples.
4. The analysis results of samples prepared by the dual-arm robot were similar with those of samples by manual.
5. For using the tools, the robot has to do a lot of motions. These motions are separated to many units. Each motion unit is defined as a motion element and is realize by a robot job.
6. A motion element moves the robot from a position (beginning position) to another position (ending position). Such a beginning position is named as front node and such an ending position is named as end node. There are two types of motion elements, including path motion element which has a fix moving path and relative motion element which has a changeable moving path.
7. As a motion element is about a short movement, several motions should be integrated for carrying out a certain task (e.g. pipetting liquid). Motion elements sharing the same position nodes, are allowed to be integrated together. A call-file is defined for realizing such integrations, which can be generated manually by using a teach pendant or on a computer or automatically by a program.
8. A database, named as M-database, has been defined for recording the information of motion elements, in order to make it feasible for a program to integrate motion elements according tasks. In M-database, each motion element has nine properties, including names, position nodes and the list of related motion elements.
9. A user-friendly interface is provided by integrating the robot system with the commercial SAMI software system. In this way, users, without skills of programming to the robot, can also operate the robot platform to prepare samples by designing processes with SAMI software.
10. A program interface, named as R-interface, had been written for integrating the robot with SAMI, which mainly includes five parts: two communication modules, a message processing module, a robot motion integration module and the M-database module.
11. The communication modules include the module for communicating with SAMI software, which is based on the socket (TCP protocol), and the module for communicating with robot controller FS100, which is based on the High-speed Ethernet protocol (called HSE protocol).
12. The module based on HSE protocol is used to handle data packets. A data packet contains two parts: command part (32 bytes) and data part (no more than 479 bytes). By sending and receiving data packets, the R-interface can do file control and robot control, such as downloading robot jobs, running robot jobs or reading robot status.

13. The message processing module is to handle the task messages sent from SAMI software. Such messages are used to command the robot to transport lab ware (e.g. vials, microplates and tip boxes), pipet liquid using different pipets or glass pipets, open/close vials and so on.
14. The robot motion integration module is used to integrate robot motions according the task message sent from SAMI, including indexing motion elements from M-database and generating a call-file.
15. The M-database module is to generate an M-database automatically by analyzing the contents of robot job files. So, when motion elements are changed, the M-database can be updated quickly.
16. The software of the analytical instruments is also integrated with the SAMI software. In this way, after prepared samples are transported to the analytical instrument, the analysis can be started automatically.
17. A program interface, named as LC-interface, had been written for integrating the LC-MS instrument with SAMI. This LC-interface mainly has three parts: a communication module, a message processing module, and a software GUI control module.
18. A program interface, named as GC-interface, had been written for integrating the GC-MS instrument with SAMI. The GC-interface has a similar structure with LC-interface, but they are different programs.
19. A software GUI control module is to simulate the keyboard and mouse inputs to the GUI, in order to control the software.
20. Grippers are integrated with robot. Thus, the grippers can be programmed by using robot jobs.

Abstract

Automation is broadly applied in life science field, with robots playing critical roles. In this dissertation, a platform based on a Yaskawa industrial dual-arm robot (CSDA10F) is presented, which is to automate the sample preparation processes and to integrate analytical instruments. A user-friendly interface has been provided by integrating the platform with SAMI Workstation EX Software.

For automating the sample preparation processes, the robot needs to use various commercial tools, including pipette, syringe, microplate, vial, thermo shaker, ultrasonic machine and so on. A series robot programs (robot jobs) had been built to realize the robot motions for using the lab ware and devices. The main challenges come from the complex experiment process and the narrow space environment restricting robot motions. For addressing the challenges, robot motions are separated to many units. Such a motion unit is named as a motion element and is realized by a robot job. According different experiment processes, the motion elements can be re-combined.

The LC-MS and the GC-MS instruments had been integrated into the platform. The prepared samples can be transferred to the analytical instruments by the robot. The integration of LC-MS instrument is a challenging task since it is a closed system without a common interface for automated sample delivery from other automation systems or robots. The software of analytical instruments has been integrated with the SAMI software. Thus, after samples are transferred to the analytical instruments, SAMI software can control the instruments to analyze the samples automatically. The challenge of software integration comes from that the lack of available APIs. For solving the problem, the software integration was achieved by simulating mouse and keyboard inputs to the graphical user interface of the analytical instrument's software.

A user-friendly interface is provided for users to operate the dual-arm robot platform, by integrating the robot system with SAMI software. It is easy to make an assay process using SAMI software. Then SAMI software will control the robot to fulfil the designed assay process. In this case, the user without robot programming skills can also use this robot platform to prepare samples.

A robot program interface (called R-interface) had been created for integrating the robot system with the SAMI software. The R-interface can communicate with SAMI software to receive commands sent from SAMI software and send back results to SAMI software. The R-interface can also generate a database (named as M-database) for recording the information of motion elements. The M-database can be refreshed, with newly built motion elements included to the M-database automatically. According the name and parameters of the command, R-interface selects related motion elements from M-database and integrate these motion elements by generating a special robot job. This special robot job is to drive the robot to fulfill the command. Besides, the R-interface can communicate with the robot, for handling robot jobs and controlling robot.

Zusammenfassung

Die Automation wird viel in Life-Science-Bereich angewendet, wobei die Roboter eine entscheidende Rolle spielen. In dieser Arbeit wird eine Plattform auf der Basis eines Yaskawa industriellen Dual-Arm-Roboter (CSDA 10F) vorgestellt, um Probenvorbereitungsprozesse zu automatisieren und analytische Messgeräte zu integrieren. Eine benutzerfreundliche Schnittstelle wurde durch die Integration der Plattform mit SAMI Workstation EX Software zur Verfügung gestellt.

Um Probenvorbereitungsprozesse zu automatisieren muss der Roboter verschiedene kommerzielle Werkzeuge verwenden, einschließlich Pipette, Spritze, Mikrotiterplatten, Vial, Thermoschüttler, Ultraschallgerät etc. Eine Reihe Roboterprogramme (Roboter Jobs) wurden für die Verwendung der Labormaterialien und Geräte geschrieben, um die Roboterbewegungen zu realisieren. Die wichtigsten Herausforderungen kommen von dem komplexen Prozess und der beschränkten Roboterbewegungen im Raum. Für diese Herausforderungen sind die Roboterbewegungen in viele Einheiten getrennt. Eine derartige Einheit wird als Bewegungselement bezeichnet und wird von einem Roboter als Job realisiert. Nach verschiedenen experimentellen Verfahren können die Bewegungselemente kombiniert werden.

Die LC-MS und GC-MS wurden in die Plattform integriert. Die hergestellten Proben werden durch den Roboter zum analytischen Messgerät transportiert. Die Integration der LC-MS ist eine anspruchsvolle Aufgabe, da es sich um ein geschlossenes System ohne eine gemeinsame Schnittstelle für die automatisierte Probenabgabe von anderen Automatisierungssystemen oder Robotern handelt. Die Software der analytischen Messgeräte wurde ebenfalls mit der SAMI Software integriert. Nachdem die Proben zum analytischen Messgerät transportiert sind, kann die SAMI Software, die Instrumente steuern und die Proben automatisch analysieren. Die Herausforderung der Software-Integration kommt vom Mangel an verfügbaren APIs. Für die Lösung des Problems wurde die Software-Integration durch die Simulation von Maus und Tastatureingabe auf die grafische Benutzeroberfläche der analytischen Instrumentensoftware erreicht.

Eine benutzerfreundliche Schnittstelle ist für den Nutzer des Dual-Arm-Roboters durch Integration des Robotersystems mit SAMI-Software vorgesehen. Es ist leicht ein Assay-Prozess mit SAMI Software zu realisieren. SAMI-Software kontrolliert die Roboter, um die entwickelten Assayverfahren zu erfüllen. In diesem Fall kann der Benutzer ohne Roboterprogrammierkenntnisse auch diese Roboterplattform verwenden.

Eine Roboter-Programmschnittstelle (sogenannte R-Schnittstelle) wurde für die Integration des Robotersystems mit der SAMI Software erstellt. Die R-Schnittstelle kann mit der SAMI Software kommunizieren, Befehle von der SAMI-Software empfangen und zurück zur SAMI Software senden. Die R-Schnittstelle kann auch eine Datenbank (mit dem Namen als M-Datenbank) erzeugen, die Information der Bewegungselemente speichert. Die M-Datenbank kann mit neu eingebauten Bewegungselementen, die dann automatisch in der M-Datenbank enthalten sind, aktualisiert werden. In Abhängigkeit des Namens und den Parameter des Befehls kann die R-Schnittstelle die zugehörigen Bewegungselemente aus der M-Datenbank auswählen und die Bewegungselemente integrieren, indem ein spezieller Roboterjob erzeugt wird. Diese spezielle Aufgabe wird an den Roboter gesendet, um den Befehl auszuführen. Außerdem kann die R-Schnittstelle mit dem Roboter kommunizieren.