Instructors

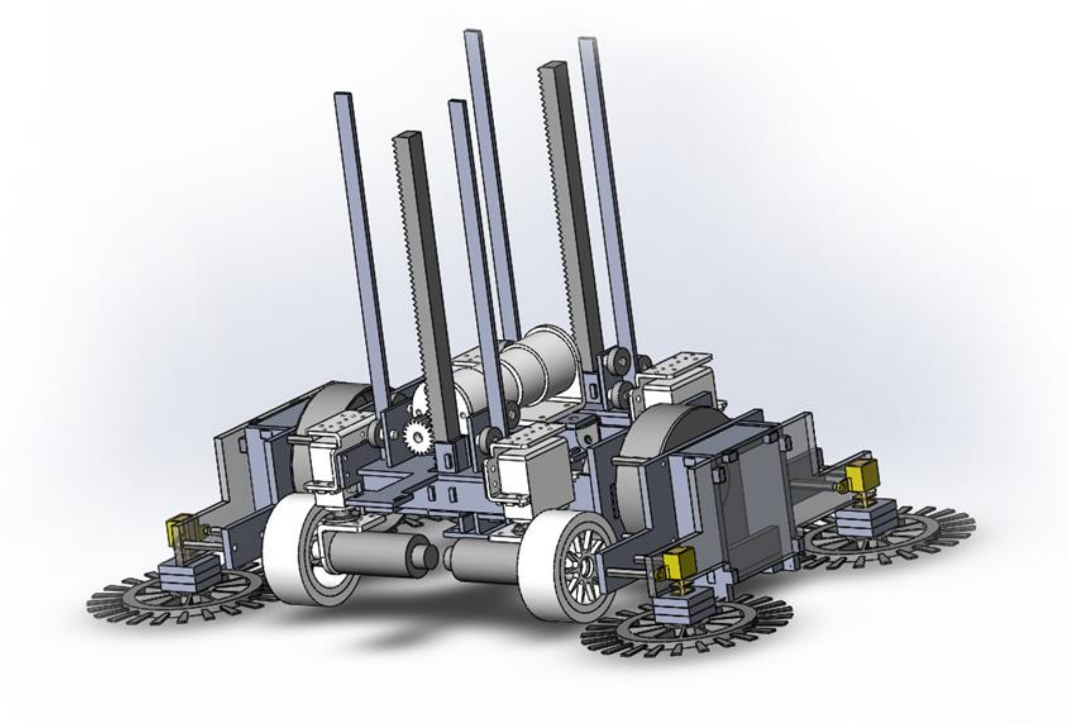
Dr. Yanfeng Shen

Dr. Cynthia Vagnetti

Stair Cleaning Robot

VG100 Introduction to Engineering

Summer 2016



Team #6 A Group

Zhu Jingyuan	515370910097
Ma Xucheng	515370910105
Liu Yihao	515370910207
Wen Yi	515370910107
Kim Sung Min	515370990014

August 10, 2016

Table of Contents

I. Executive Summary

Stairs have long existed in people's daily life, and the related cleaning has also been a common problem. Though there have been several cleaning machines available aimed at daily cleaning, none of them have dealt with the stair cleaning well. This project of Stair Cleaning Robot (shown in *Figure I.1*) is proposed to solve the specific problem of stair cleaning. The work of cleaning stairs can be simplified and so human efforts can be released from the tedious and repetitive labor work. The robot aims to achieve the goal by accomplishing two main functions: climbing stairs automatically and cleaning stairs skillfully. It is basically made up of four parts: the platform (for supporting), the cleaner (for cleaning), the car (for 2D movement), and the lifter (for climbing). In this report, we will introduce the robot functionally and show its performance in solving the problem of stair cleaning. In conclusion, our Stair Cleaning Robot can relieve the work of stair cleaning from human beings.

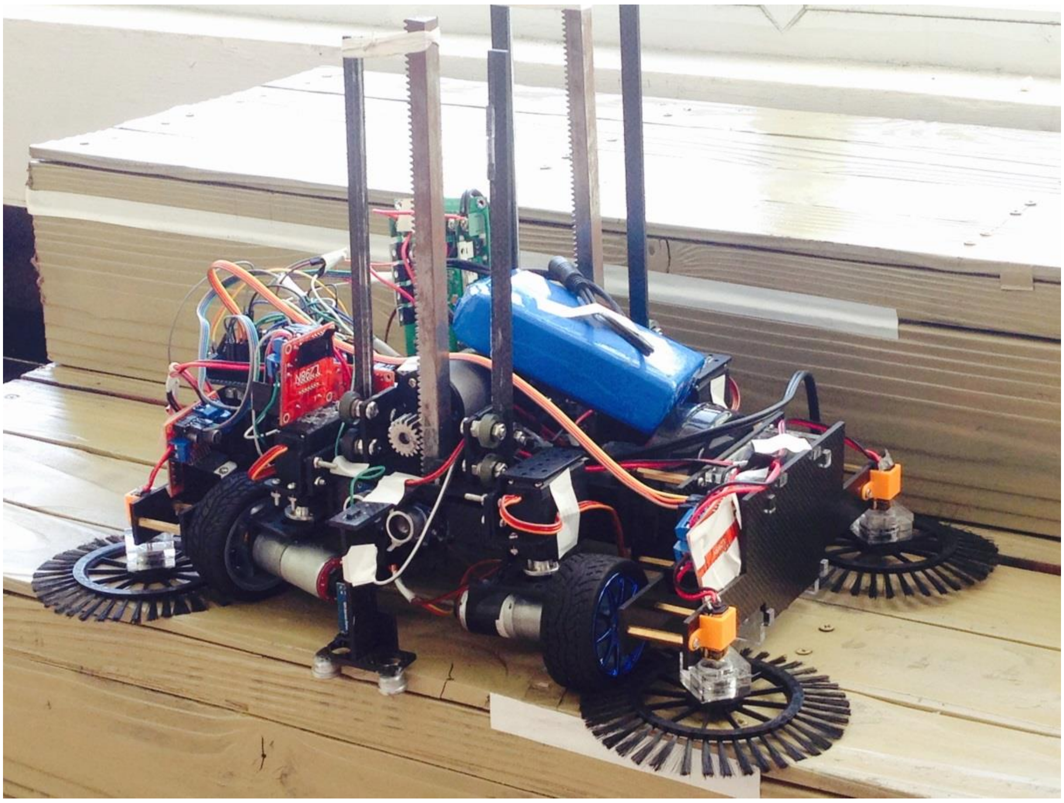


Figure I.1 The prototype of Stair Cleaning Robot.

II. Introduction

Our group name is “A Group”. We are a five-people group of freshmen in University of Michigan-Shanghai Jiao Tong University Joint Institute (UM-SJTU JI), which is located in the campus of SJTU, Minhang, Shanghai, China. JI is a leading international engineering college, aiming to educate innovative leaders.

Our members are Wen Yi, Kim Sung Min, Zhu Jingyuan, Ma Xucheng, and Liu Yihao (shown in *Figure II.1* in the same order from left to right).

Figure II.1 All the group members with the prototype in front of JI Building.

Our motivation for this project is to make the stair cleaning procedure performed with higher automation and to make the process short, simple, and easy. So, we have organized together to make the Stair Cleaning Robot. We are a group of ME and ECE students that use mechatronics to make this possible.

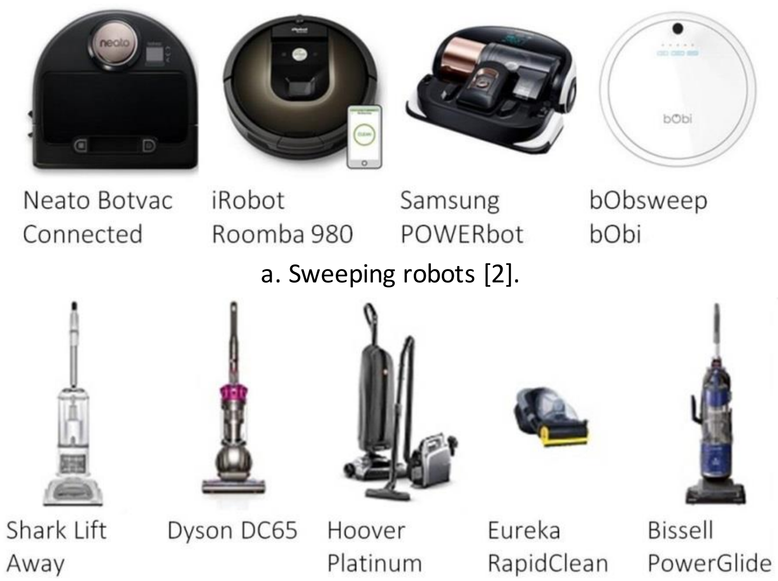
We have made a specific plan to complete our project and have managed to fulfill it. We have succeeded in producing an intelligent robot to satisfy the human need for automatic cleaning of stair areas.

III. Problem

A. Statement of the problem

There is a constant increase in the number of high skyscrapers, buildings that are higher than 200 meters. Only in 2015, there has been made 106 skyscrapers, making a world record of skyscraper made per year [1]. Buildings are getting higher and higher, which means that more stairs emerge.

However, the efficiency of tools for cleaning these stairs doesn't follow the high increasing rate. There are two models of cleaners as shown in *Figure III.1* that we consider as our competing goods. The first is sweeping robots and the second is stair cleaners.



a. Sweeping robots [2].

b. Stair cleaners [3].

Figure III.1 Two available cleaning machines.

Our team believes that there is a need to make an automatic stair cleaning robot. We have narrowed down our target customers to our product to both cleaning industries and houses. Cleaning industries will want a cost-efficient tool to clean the stairs of high buildings that they are in charge of. Not every domestic house is our target, but we believe there are potential customers. Some people's housings contain stairs. The houses that want premium services are open minded to new technologies.

B. Summary of the problem

In the area of cleaning tools, we still face three problems:

- **Stair inaccessibility**
- **Ignorance of stair cleaning**
- **Lack of automation**
-

IV. Needs and Validation

A. Needs

To solve the current problems that cleaning machines have, we require:

- **A stair-climbing vehicle**
  In order to have an access to the stair area and to move around the different levels, we need a stair-climbing vehicle.
- **A well-designed cleaner**
  For the final foal of cleaning the stairs, our product needs a cleaner that is capable of handling the special structure of stairs.
- **A precisely-detecting system**
  To complete the work independently, the robot needs to detect its surroundings very accurately.

B. Validation

We conducted a survey among our audience. The targets involved ten people: people of company management, employed cleaners, and people cleaning home stairs. *Figure IV.1* and *Figure IV.2* are respectively their idea of the product and expected prices.

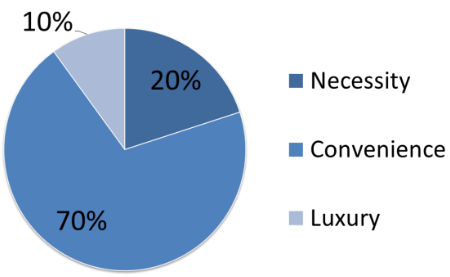


Figure IV.1 The pie chart for product positioning

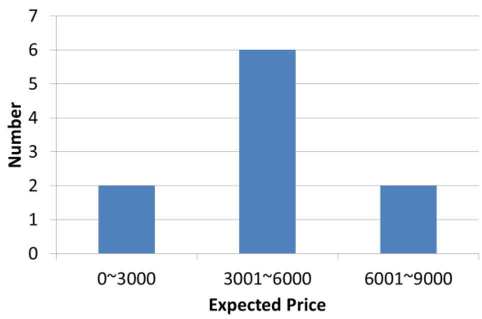


Figure IV.2 The bar chart for expected price.

The following are their primary concerns:

- Companies expected these robots to help induce payment to employees.
- Family users thought it would improve life quality.
- Employed cleaners were afraid that they would later lose their jobs.

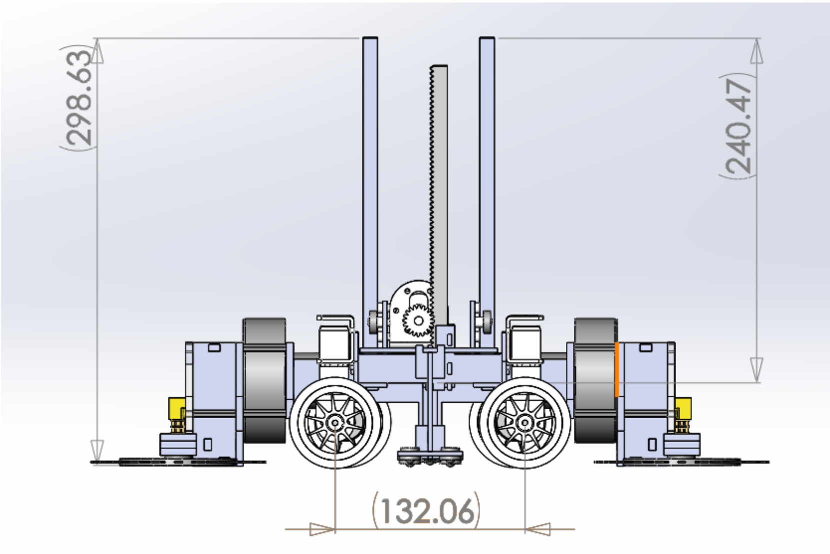
Based on the result of the survey, we have positioned our product as a convenience to help organizations to cut salary costs on human cleaners and assist the households with stairs through premium cleaning. Although the launch of such a product may induce unemployment to some extent, it will also create new positions for technicians and engineers. So, the people that will benefit from this product will be both cleaning industries with need for low-budget cleaning tools for high buildings and the individuals wanting premium services.

V. The Solution – Stair Cleaning Robot

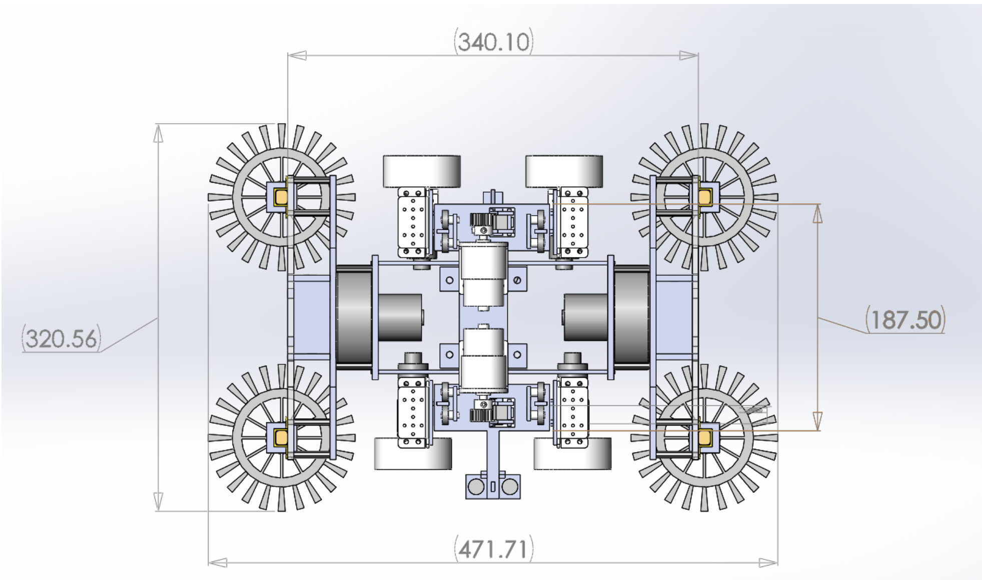
A. Entailment

As shown in *Figure V.1*, Stair Cleaning Robot entails:

- A smoothly-running vehicle that can move in all directions
- A stable lifting system to manage vertical movement
- An efficient cleaner that can deal with the special structure of stairs
- A comprehensive controlling system



a. The front view.



b. The top view.

Figure V.1 The dimension diagram of Stair Cleaning Robot (Unit: *mm*).

B. Function

The robot aims to accomplish climbing the stairs and cleaning the floor at the same time, which together can make the goal of cleaning the stairs come true. The rubbish is collected in certain containers for later dump.

C. Description

The concept diagram is shown in *Figure V.2*. Stair Cleaning Robot consists of four main parts: the platform, the cleaner, the car, and the lifter and the working consists of two main functions: climbing stairs and cleaning.

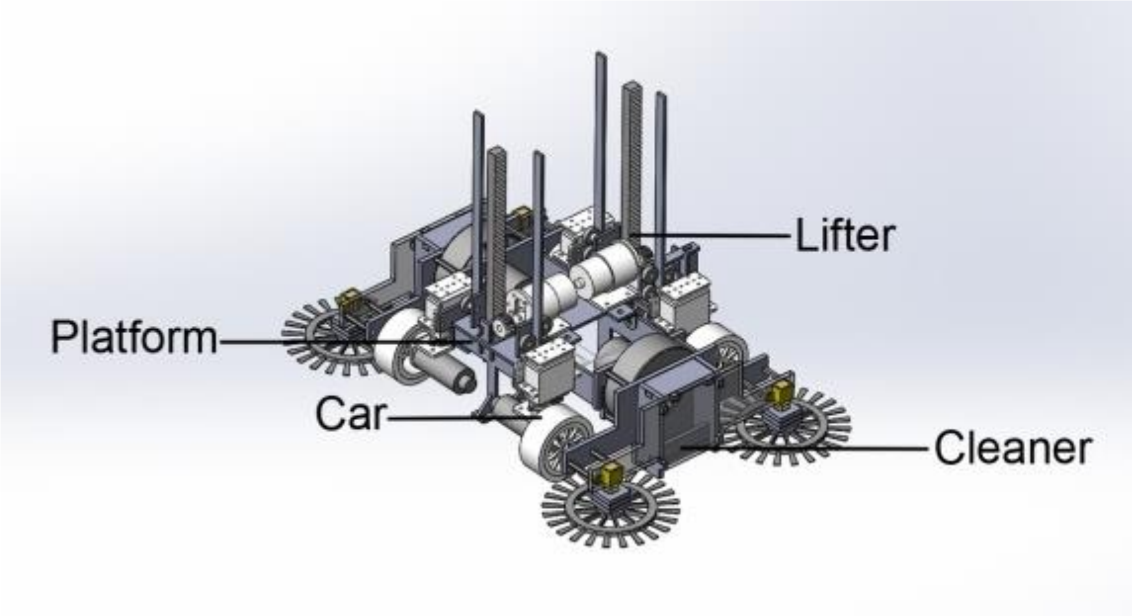


Figure V.2 The concept diagram with main parts of Stair Cleaning Robot.

Platform

The platform is the main supporting structure of the whole robot. It carries the cleaners and bears almost all the electronic components. In the working process, it is the first part to be sent to the upper stair, which is to fix on the upper stair when the following parts are lifted.

Cleaner

Our cleaners, consisting of brushes, fans and boxes, aim to handle regular rubbishes. They can effectively collect rubbishes of a certain range of sizes.

Car

The car, mainly made up of well-designed active wheels, functions as the driving system that enables the whole robot run perfectly in the two-dimension space.

Lifter

The lifter, mainly made up of racks and pinions, manages to change the height of different parts of the robot.

Climbing stairs

With the car and the lifter, climbing stairs is realized as shown in *Figure V.3*.

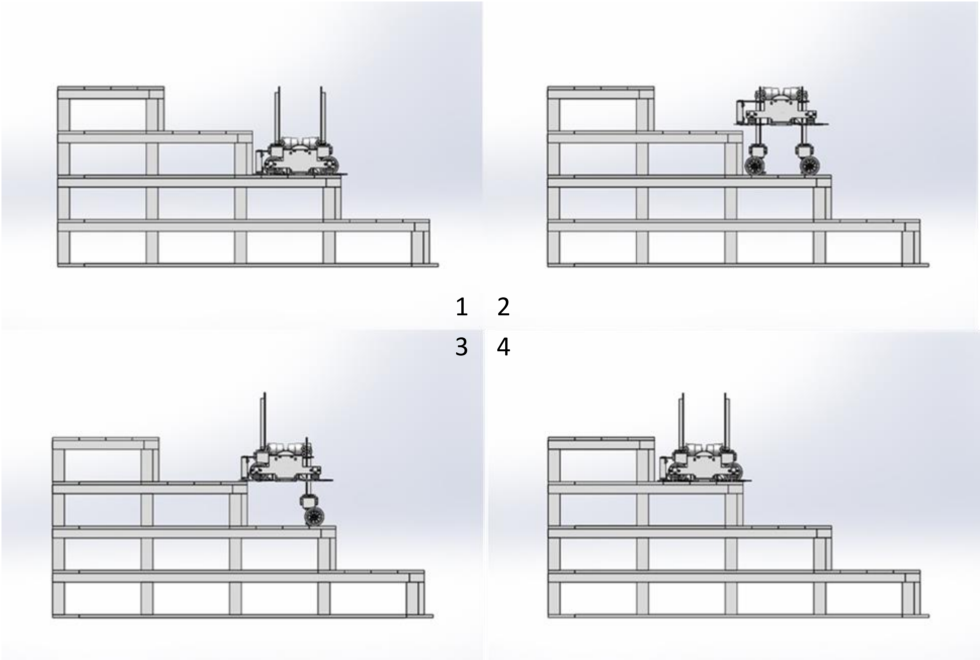


Figure V.3 The step-by-step figure of climbing stairs.

Cleaning stairs

With the brushes and the fans, climbing stairs is realized as shown in *Figure V.4*.

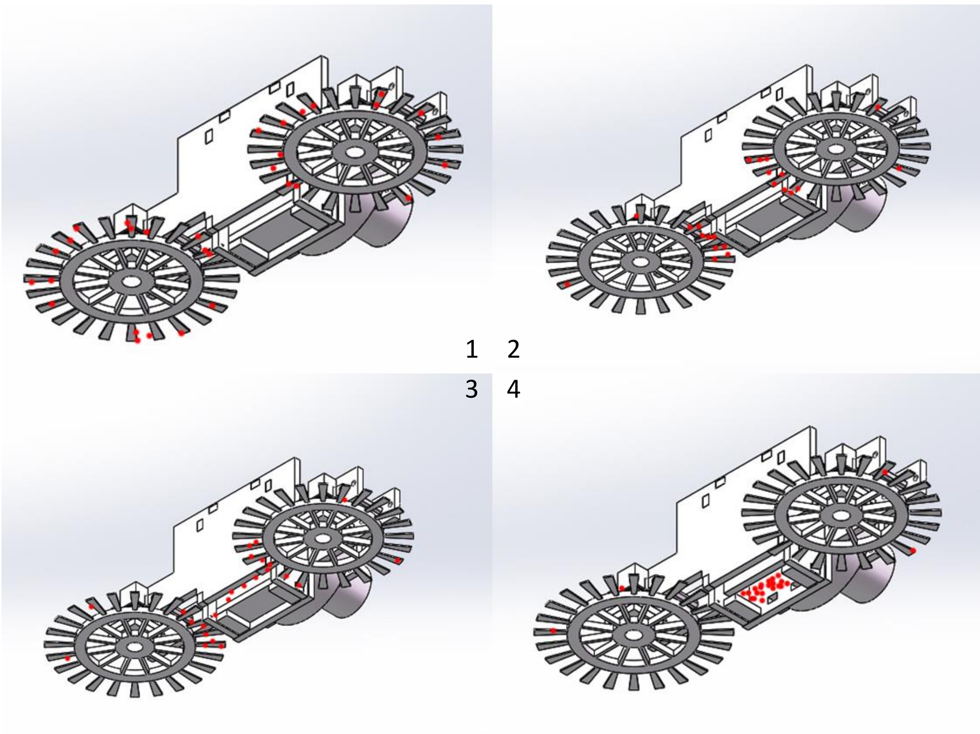


Figure V.4 The step-by-step figure of cleaning.

The overall working process was shown in *Figure V.5*.

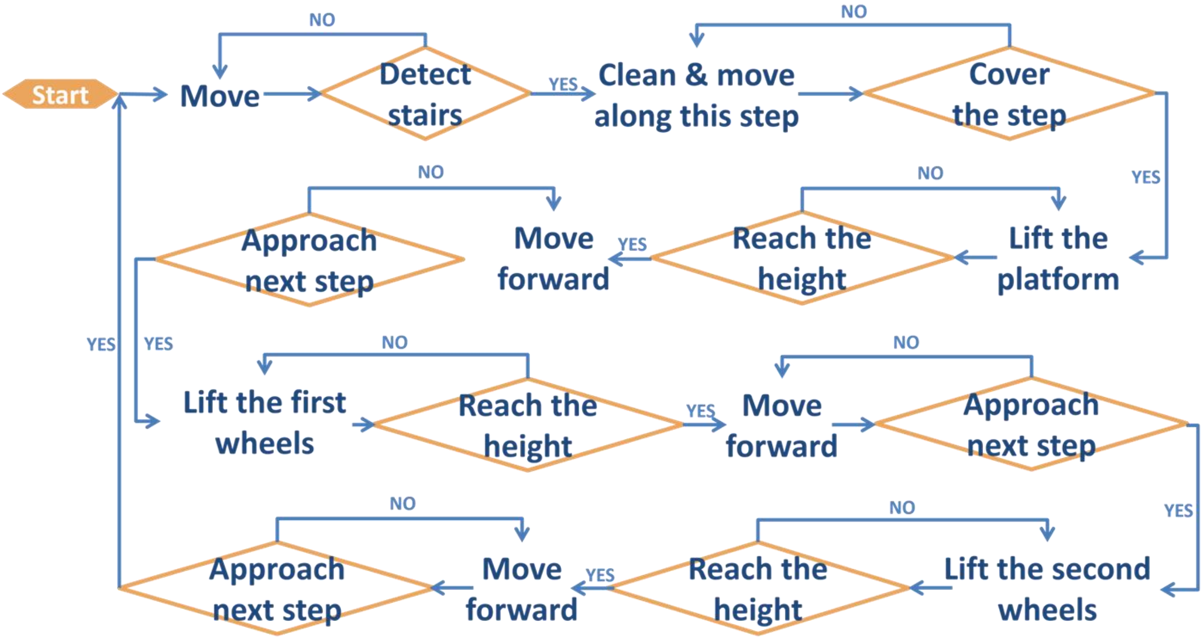


Figure V.5 The working process.

D. Functional Components

Components 1~4 were labelled in *Figure V.6*.

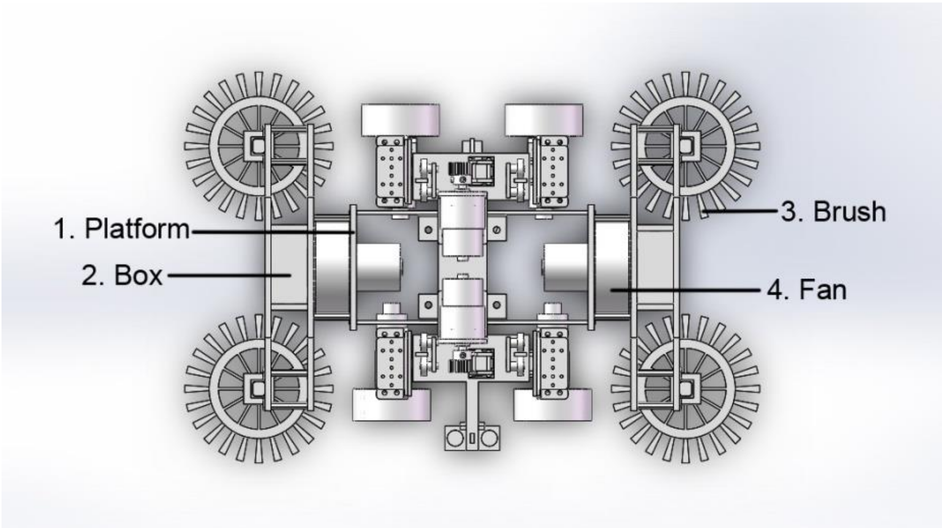


Figure V.6 The top view of the concept diagram of Stair Cleaning Robot.

1. The platform

The platform is a main body part of the robot, which supports and carries almost all the other parts.

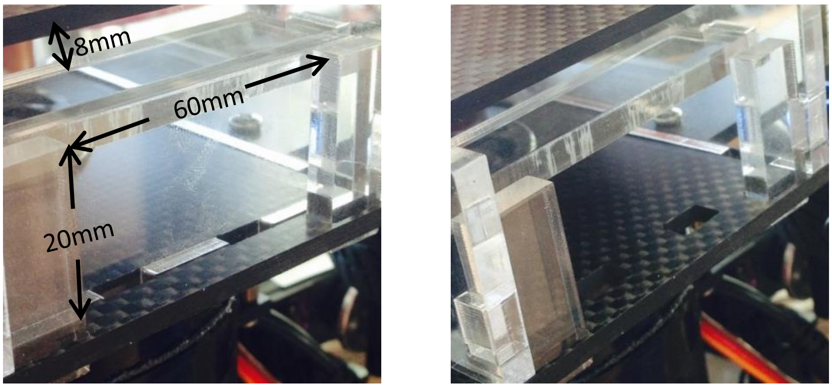
To lighten the weight, the materials are selected as carbon fibers (shown in *Figure V.7*). Carbon Fibers are fibers with the properties of good strength and low density [4], which satisfies the requirement for supporting structure.

According to the related law, all the stair steps are about 30 *cm* wide [5]. Hence, the platform was constructed for 30-cm-wide stair steps.

Figure V.7 Carbon Fiber.

2. Boxes as rubbish collectors on both sides

Two specially-designed boxes are attached on two sides of the platform. They work as rubbish collectors to contain rubbish temporarily. The boxes are deigned similar to drawers, which can be opened easily (shown in *Figure V.8*) when rubbish needs to be dumped.



a. The box closed b. The box opened

Figure V.8 The closed and opened status of the rubbish collector.

3. Round brushes at four corners

The round brushes at four corners are attached to motors in order to keep rotating when cleaning the stairs.

On each side, the two brushed rotate in different directions, as shown in *Figure V.9*, to send rubbish into the suction port. The design focuses on the corners, the tricky details which are everywhere in the structure of stairs.

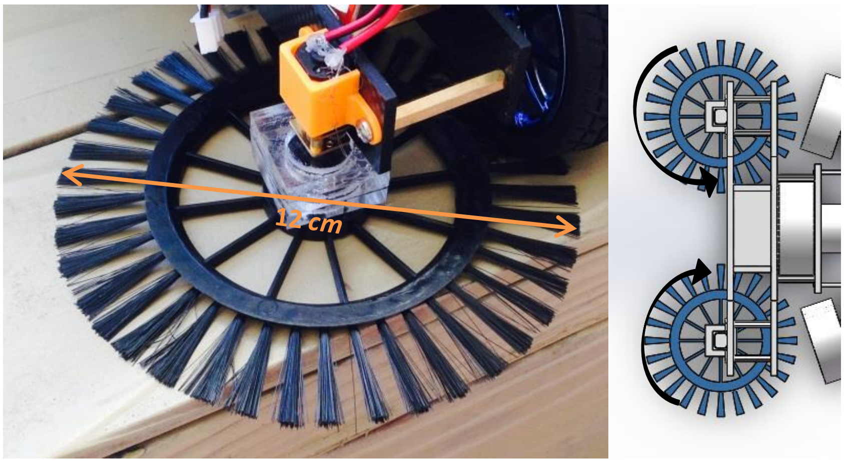


Figure V.9 The round brushed fixed at the corner of the robot.

4. Electric fans on both sides

Accessories of vacuum cleaners were used to make fans driven by motors, shown in *Figure V.10*. The housings of the fans are connected to the rubbish collectors. When the fans work, the forming air flows drag rubbish through the pressure difference passing from the inside of the housings to the collectors. The rubbish is then dragged through the entries of the collectors and stopped by the filter at the connection between the collectors and the fans. The parameters for the fan with a motor are shown in *Table V.1*.

Figure V.10
An electric fan.

Table V.1 The parameters for the fan with a motor [6].

Current	DC	Diameter of the Scuttle *d*	39 *mm*
Rated Voltage	12 *V*	Diameter of the Housing *D*	87 *mm*
Rate Power	100 *W*	Thickness of the Housing *a*	34 *mm*

Components 5~8 were labelled in *Figure V.11*.

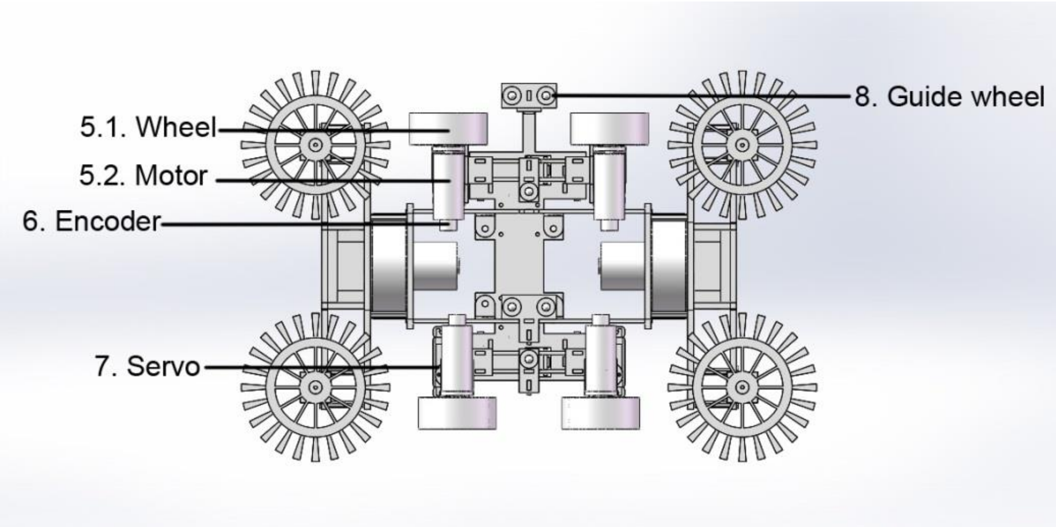


Figure V.11 The bottom view of the concept diagram of Stair Cleaning Robot.

5. Motors and wheels

Motors are used to activate wheels to manage the simplest movement of the robot, ensuring the covering over the two-dimension surface. The related parameters are shown in *Table V.2* and *Table V.3*.

Table V.2 The parameters for the active wheel.

Thickness of the Wheel	26 *mm*	Diameter of the Wheel	65 *mm*

Table V.3 The parameters for the motor attached to a wheel [7].

Operating Voltage	6~24 *V*	Nominal Voltage	12 *C*
Free-run Speed at 12 *V*	126 *RPM*	Free-run Current at 12 *V*	46 *mA*
Stall Current at 12 *V*	1 *A*	Stall Torque at 12 *V*	4.2 *kg·cm*
Redactor Size	21 *mm*	Weight	85 *g*

6. Encoders

For precision of move, encoders are used to convert angular motions of wheels to digital messages, based on which programming coordinates the wheels.

For our encoder, the Hall precision is 32.73° [7], which means that the encoder will give 11 pulse signal feedbacks when it rotates once. Then, we need to multiple the precision by the gear ratio, which is the ratio between motor and encoder. Our gear ratio is 74.83 [7], thus the hall resolution can be calculated as:

$$Hall\ resolution = Hall\ precision \times gear\ ratio = 823.1\ PPR\ ,$$

which means that the encoder will give about 823 feedbacks when the motor rotate once. Therefore, we can use this number to detect the speed of rotation and do the correction. [8]

7. Servos attached on wheels

Instead of applying different speeds on different wheels, the change in the movement direction is realized by the servos attached on the wheels. The servos can rotate the wheels directly, which means turning the robot immediately and accomplishing the sharp turn. The related parameters are shown in *Table V.4*

Table V.4 The parameters for the servo attached to a wheel [9].

Weight	55 *g*
Dimensions	40.7x19.7x42.9 *mm*
Stall Torque	9.4 *kgf·cm* (4.8 *V*), 11 *kgf·cm* (6 *V*)
Operating Speed	0.17 *s*/60° (4.8 *V*), 0.14 *s*/60° (6 *V*)
Operating Voltage	4.8~7.2 *V*
Running Current	500~900 *mA* (6 *V*)
Stall Current	2.5 *A* (6 *v*)
Dead Band Width	5 *µs*
Temperature Range	0 ~55°*C*

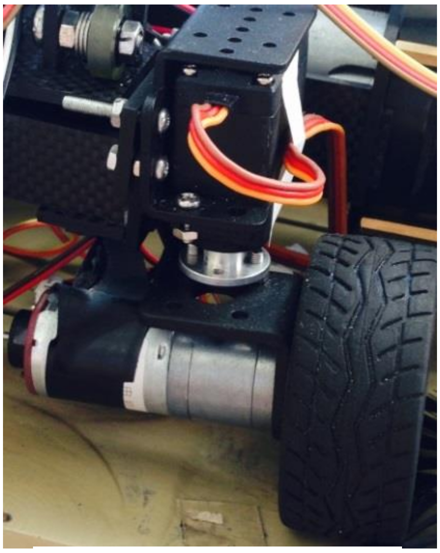


Figure V.12
A set of a wheel, a motor, an encoder, and a servo.

The Components 5~7 are shown in *Figure V.12*

8. Guide wheels at the bottom

The guide wheels, shown in *Figure V.13*, are an auxiliary system of moving. They work when the robot is climbing a step of the stairs. They help move the robot in the process of lifting and become force bearing points when one pair of wheels is lifted in the air.

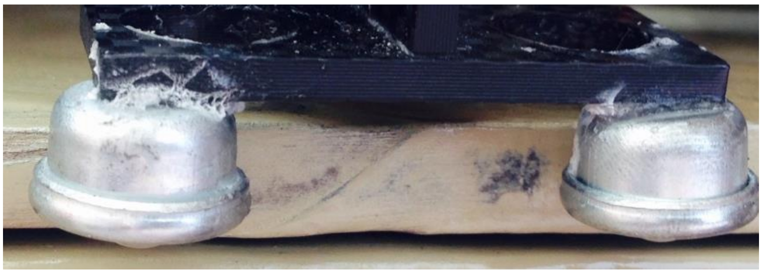


Figure V.13 Some guide wheels at the bottom.

Components 9~10 were labelled in *Figure V.14*.

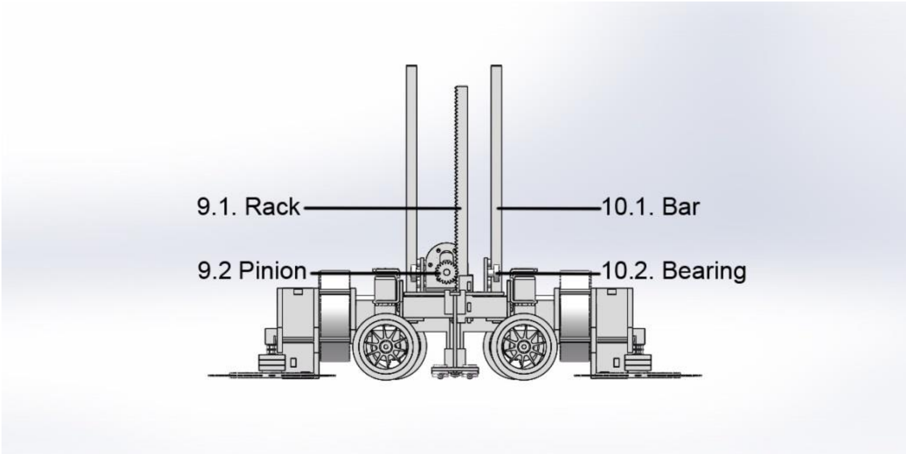


Figure V.14 The side view of the concept diagram of Stair Cleaning Robot.

9. Racks and pinions

A rack is a bar with teeth biting with pinions. According to Murthy [10], "[a]s the rack is driven by a pinion, the rotary motion is converted into a linear motion" (in *Figure V.15*).

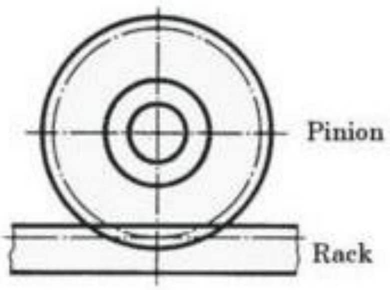


Figure V.15 Rack and pinion [10].

The linear motion in racks fixed in the robot, in *Figure V.16*, changes the vertical height of some corresponding parts.

Motors are attached to run the system. The parameters are in *Table V.5*.

Table V.5 The parameters for the motor attached to a pinion [11].

Rated Voltage	DC 12 *V*	No Load Speed	22 *rpm*
No Load Current	50 *mA*	Load Torque	4 *kg·cm*
Load Speed	18 *rpm*	Load Current	80 *mA*
Load Output	0.8 *w*	Stall Torque	16 *kg·cm*
Stall Current	1.2 *A*	Gear Ratio	270
Weight	164 *g*		

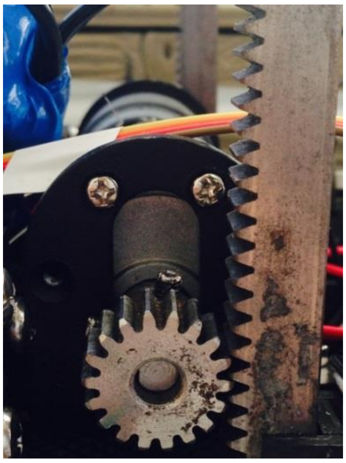


Figure V.17 A rack and a pinion.

Figure V.16 A bar and bearings.

10. A correction system of the vertical direction

A system of U groove bearings and bars, shown in *Figure V.17*, is created to correct the direction when lifting. Since the bearing weight may be too big for racks to support independently. The bars help to support together. Every set of three bearings seize the vertical bar in the form of a triangle. These ensure the vertical up direction.

Components 11~13 were labelled in *Figure V.18*

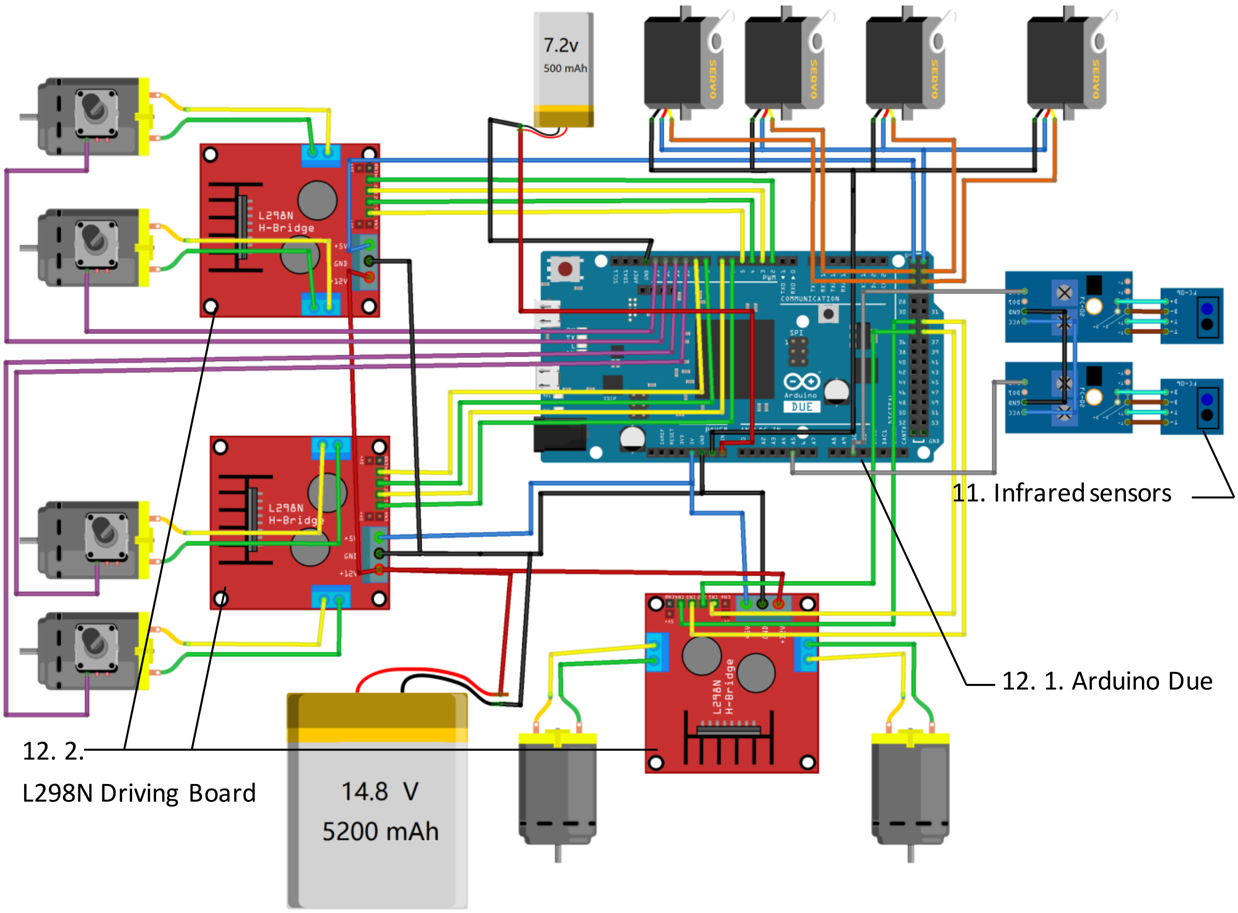


Figure V.18 The wiring diagram for Stair Cleaning Robot.

11. Infrared sensors

The infrared sensors (shown in *Figure V.19*) work by emitting and receiving infrared ray.

First, the transmitting part emits the infrared. If the receiver receives no feedback, the circuit won't be connected and the output voltage will always be high. If the infrared is reflected, the receiver will receive the reflected infrared and thus get through the circuit to let the output pin to have the low voltage. The reflected infrared's intensity will depend on the color of the object that the infrared touches.

By digital part's processing, the analog output will be distributed between 0-1023, where 0 is the highest intensity of infrared reflected and 1023 is no infrared received. [12]

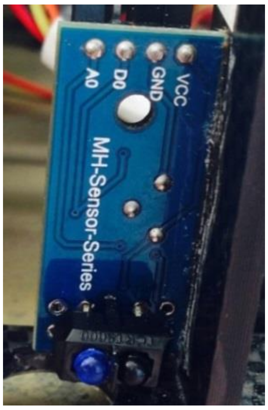


Figure V.19
An infrared ray sensor.

12. The controlling system

There are mainly two controlling boards in our project, the Arduino board and the driving board for motors.

For the controller, we choose Arduino Due (shown in *Figure 5.20*).

The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU and a 32-bit ARM core microcontroller. It has 54 digital input/output pins (of which 12 can be used as PWM outputs), 12 analog inputs, 4 UARTs (hardware serial ports), a 84 MHz clock, an USB OTG capable connection, 2 DAC (digital to analog), 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button. It contains everything needed to support the microcontroller. Connecting it to a computer with a micro-USB cable or powering it with an AC-to-DC adapter or battery can get it started. The Arduino Due is compatible with all Arduino shields that work at 3.3V and are compliant with the 1.0 Arduino pinout. [13]

The parameters for Arduino Due are shown in *Table V.6.*

Table V.6 The parameters for the Arduino Due [13].

Microcontroller	AT91SAM3X8E	Operating Voltage	3.3 *V*
Recommended Input Voltage	7~12 *V*	Limited Input Voltage	6~16 *V*
Digital I/O Pins	54 (12 PMW O)	Analog Input Pins	12
Analog Output Pins	2 (DAC)	DC Output Current on all lines	130 *mA*
DC Current for 3.3 *V* Pin	800 *mA*	DC Current for 5 V Pin	800 *mA*
Flash Memory	512 *KB*	SRAM	96 *KB*
Clock Speed	84 *MHz*	Length	101.52 *mm*
Width	53.3 *mm*	Weight	36 *g*

There are two reasons why we choose it:

First, we need many pins for the controlling, and Arduino Due (shown in *Figure V.20*) has 54 pins for output/input, 14 of which are available for pwm controlling.

Second, since we need to use the encoders, we need many pins with interrupt function to monitor the speed of motors anytime. Most Arduino boards only have one or two pins with interrupt function limited to their performance. However, Arduino Due has the property that all of its pins have the interrupt function because of its ARM architecture and its 32-bit processor.

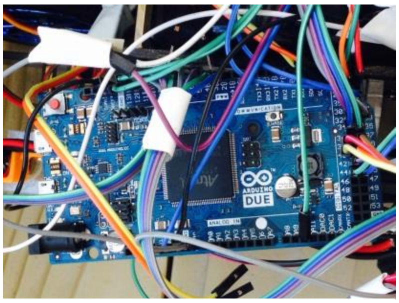


Figure V.20 Arduino Due.

For the driving board for motors, we choose to use the L298N (shown in *Figure V.21*) for the driving of motors due to its high coordination with Arduino, high current carrying capacity and availability of pwm control.

The parameters for the driving board are shown in *Table V.7.*

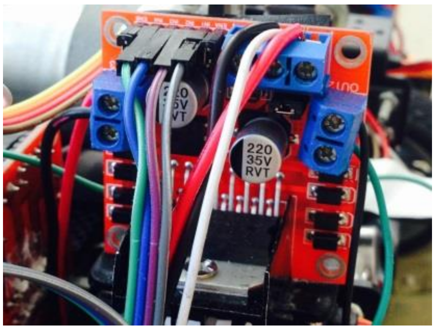


Figure V.21 L298N Driving Board.

Table V.7 The parameters for L298N Driving Board [14].

Logical Voltage	5 *V*	Drive Voltage	5~35 *V*
Logical Current	0~36 *mA*	Drive Current	2 *A*
Max Power	25 *W*	Dimensions	43x43x26 *mm*
Weight	26 *g*		

13. Battery

The battery in *Figure V.22* is one of the power supplies designed to be carried with the robot when working, which ensures the flexible movement in a wide range.

We choose Lithium Batteries for our robot due to their advantages of superior useable capacity and extended cycle life [15].

Figure V.22 The battery.

E. Fractional Procedures

Procedures are the concrete steps that we did to make the design of the solution come true. In this section, there will be specific information about the procedures to establish the structure of stair climbing, the structure of stair cleaning, the automatic control system and the modification.

The structure of stair climbing

1. File materials

We chose carbon fibers as the main materials for the structure. The tailored carbon fibers needed filing to slightly modify the inprecision of cutting in the process of material preparation. Files are shown in *Figure V.23*.

Figure V.23 The files.

2. Construct the platform

All the small blocks were built together as designed before. The connections were glued by 502.

3. Assemble four sets of active wheels

For one set, an encoder was attached to a motor. The motor was attached to an active wheel. At the connection between the motor and the wheel, an L-bar was fixed. A servo was vertically attached perpendicular to the motor set horizontally to the other plane of the L-bar. Then, one set was completed as shown in *Figure V.24.*

Figure V.24 One completed set of an active wheel.

The assembly of such a set was repeated for another three times to get four set in all.

4. Fix the sets of active wheels

The four sets were added to the bottom of the platform at the designed positions. They were connected at one side of the servo body to the platform.

5. Fix the racks

We designed grooves which have the same shapes and sizes for the racks to insert in. The racks would not shake if they were in the grooves. To make the fixing more stable and strong, we also glued the racks and the grooves with 502. This avoided the racks being pulled out by the motors when the wheels are lifted.

6. Fix the pinions

The pinions were fixed on the shafts of motors. There were many screw threads on both of the shafts and the pinions. Therefore, the screws could be tightened to make the shafts and pinions fixed together.

The pinions and the racks needed to be positioned precisely to make them bite each other well as shown in *Figure V.25*.

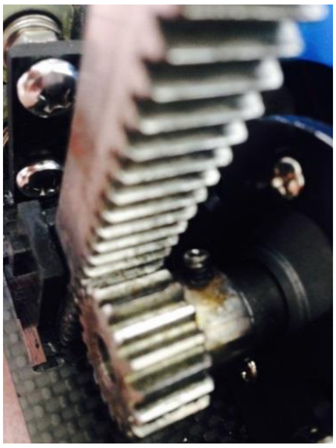


Figure V.25 The contact between a rack and a pinion.

7. Fix the bars

Four acrylic bars of size 22 *mm**4 *mm**3 *mm* was fixed on the inner side of the carbon fiber guiding carbon fiber scrip using 502 glue (as shown in *Figure V.26*). Before gluing the bars on, emery paper was used to polish the surface of the bar as well as the surface of the carbon fiber scrips to make them rough.

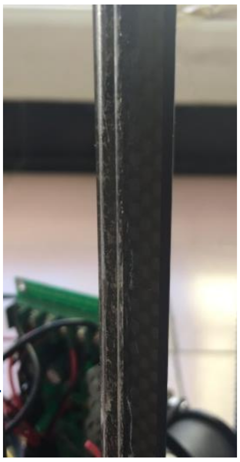


Figure V.26 An acrylic bar attached to a carbon fiber bar.

8. Assemble the sets of bearings and fix them

For each set of the three bearings, first three screws with length of 12 *mm* and diameter of 5 *mm* were assembled into the holes on the holder on the plate; then one nut and three gaskets were added onto each screw. Last, three bearings were assembled and nuts were used to fix them on the screws.

This was repeated to do four sets and all the sets were then fixed with the corresponding bars (as shown in *Figure V.27*).

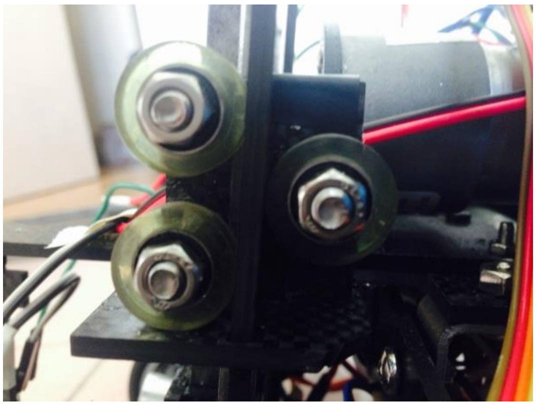


Figure V.27 A fixed set of bearings.

The structure of stair cleaning

1. File materials

We also chose carbon fibers as the materials for the cleaner. The inprecision of cutting still needed filing to modify.

2. Construct the easily open box

To be user-friendly, the rubbish collectors were designed to be easily-open boxes. The connections were glued by 502.

3. Attach the connection boards

With the suctions downward, both the long sides of the collectors were attached to boards. The connections were glued by 502. The boards extended on both ends for further connection to brushes. Small columns were added between two boards to stabilize them.

4. Fix the electric fans

The electric fans were fixed, with the motor side to the platform and the scuttle side to the collectors (as shown in *Figure V.28*). The connections were glued by 502. Thus, the fans also became connecting parts between the platform and the collectors, and in this way the collectors were attached to the platform.

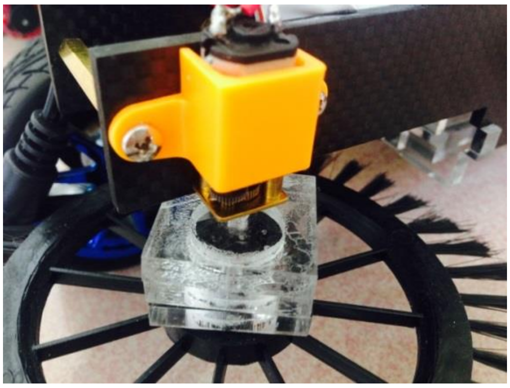


Figure V.29 A fixed round brush

Figure V.28 A fixed electric fan.

5. Fix the round brushes

The brushes were fixed to the ends of the connection boards with motors attached. The circle-shaped brushes were only fixed at the points of their centers. So, they had to be firmly connected to make sure their surfaces were horizontal. We employed a pile of small square pieces for attachment (as shown in *Figure V.29*).

The automatic control system

1. Fix sensors

IR sensors were used for detecting. Since there were no external forces on the sensors, 502 glue was enough to stick them on the carbon fiber in specific positions. The attachment is shown in *Figure V.30.*

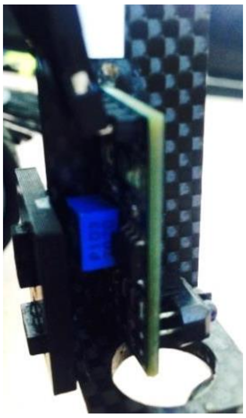


Figure V.30 The back of a fixed infrared sensor.

2. Wire all the electronic components

All the components were wired together into a whole system as shown in *Table V.8.*

Table V.8 Wiring.

Component	Component PIN	Arduino PIN
Servo (Left Forward)	Signal	22
Servo (Right Forward)	Signal	23
Servo (Left Backward)	Signal	24
Servo (Right Backward)	Signal	25
Driver Board 1	IN 1	2
	IN 2	3
	IN 3	4
	IN 4	5
Driver Board 2	IN 1	5
	IN 2	6
	IN 3	7
	IN 4	8
Driver Board 3	IN 1	32
	IN 2	33
	IN 3	34
	IN 4	35
Encoder (Left Forward)	PIN A	10
Encoder (Right Forward)	PIN A	11
Encoder (Left Backward)	PIN A	12
Encoder (Right Backward)	PIN A	13
IR Sensor (Forward)	A01	A10
IR Sensor (Backward)	A01	A05

* The PINs of 5V and GND of all components are connected to the PINs of 5V and GND of Arduino respectively (except that driving board's 5V PINs don't connect with the Arduino's 5V PINs).

3. Test the operation of wheel sets

The coding was implanted to the motors in the wheel sets. The robot was laid on the level ground to run. The coordination between wheels was adjusted.

The coding was implanted to the servos in the wheel sets. The wheels were kept in air (as shown in *Figure V.31*) and then the programming was started. The direction-control was adjusted.

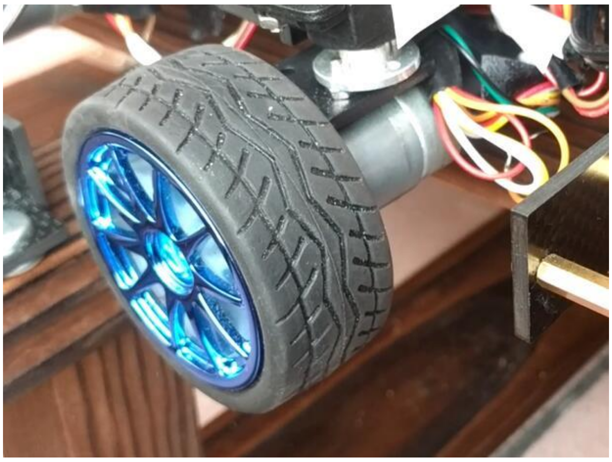


Figure V.31 A tested wheel in air.

4. Test the operation of the lifter

The coding was implanted to the motors attached to the pinions. The lifter was then tested. The speed of lifting was adjusted.

5. Try the circle of climbing

The robot was put in front of a certain stair step. The estimated time duration was used in the coding to make the robot accomplish the movement of climbing stairs without detecting the environment.

6. Test the operation of the cleaner

The coding was implanted to the motors attached to the fans and brushes. Pieces of rubbishes were scattered around the robot to verify its capacity of cleaning (as shown in *Figure V.32*).

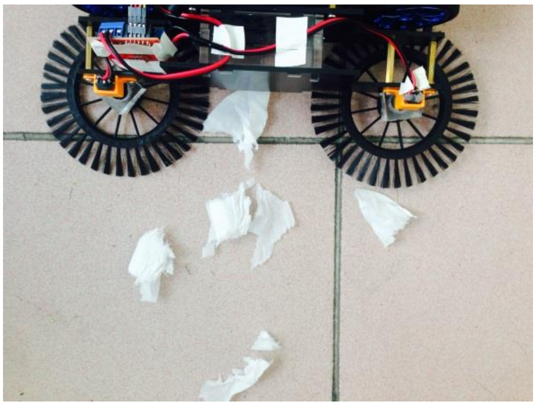


Figure V.32 The test on a cleaner.

7. Add sensors to the operations

The sensor started working to collect information from the outside. The coding was written and implanted to deal with the data and commanded the robot to response instantaneously. The robot was tested to climb a stair step with the help of the sensors.

8. Coordinate all the parts to complete the control system

All the testing coding was integrated into a whole program to realize all the functions at the same time.

9. Fix the all the components

The boards were glued to some specific positions with 502. Some wires were fixed with tapes (as shown in *Figure V.33*).

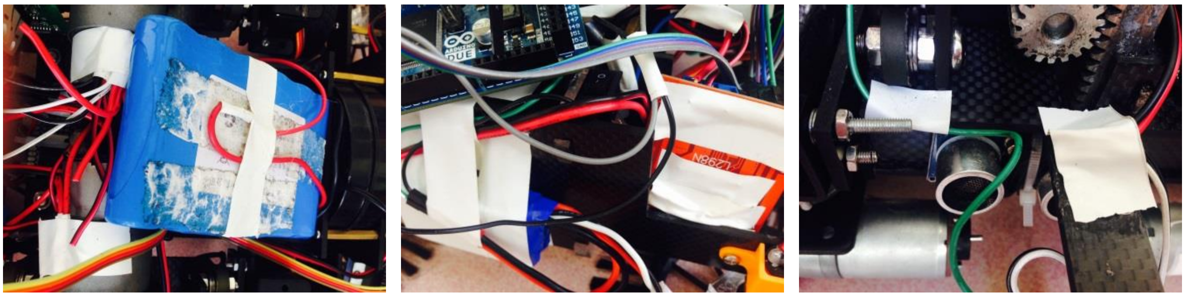


Figure V.33 Some of the fixations.

10. Weld the connections of wires

The wires were welded to the components (as shown in *Figure V.34*).

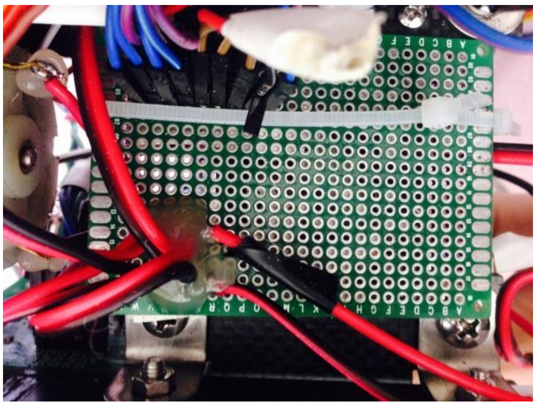


Figure V.34 One of the welded boards.

The modification

1. Change the supply of power

After integrating all the functions, the battery was found to fail to support all the components well. The addition of batteries provided the stronger efficiency to the electric fans.

2. Check the chassis height

When testing stair climbing, the robot tended to be stuck. Thus, the chassis height needed to be continuously adjusted. The bottom of the robot is shown in *Figure V.35.*

Firstly, some guide wheels were removed from the bottom of the robot to simplify the structure without influencing the climbing function.

Then, fine tuning was done on all the other bottom parts, especially the collectors and brushes in the cleaning system to avoid being stuck in the process of stair climbing.

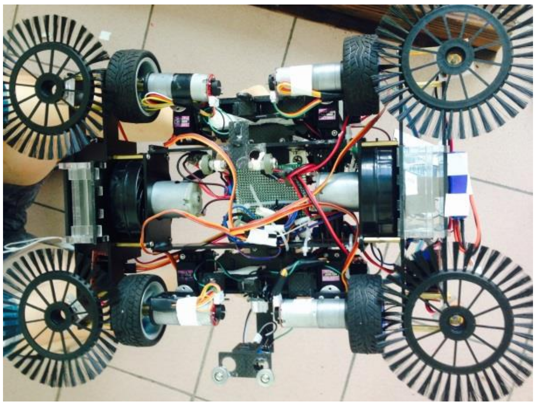


Figure V.35 The bottom of the robot.

3. Modify the sensors

The lifting would continue for some time to let the wheels be higher than the stairs when it detected that there is no block in the front. For this time interval, we tested for several times to get the proper data.

Also, the working range of the IR sensors was low, so we needed to make sure that the robot was close enough to the upper stair.

4. Modify the control over direction

When the servos rotated the active wheels to change the direction, according to our previous design, the wheels were still laid on the ground, which meant that the friction exerted by the ground would hint the rotation of the wheels. This was an influential factor for inaccuracy of direction-control.

Hence, the design was modified into supporting the robot at the guide wheels when rotating the active wheels. In the new design, the wheels were in air when rotating to change the direction.

F. Results of Tests

In the experiments on the separated functional components for test, the data are measured as shown in *Table V.9.*

Table V.9 Tests on separate functional components.

Object	Result
Deviation angle of moving	≤ 1°
Maximum velocity of moving	≥ 20 *m/min*
Maximum bearing of lifting	≥ 2 *kg*
Coverage of cleaning	≥ 30 *cm*
Life of the battery	≥ 1.5 *hr*
Range of Sensoring	≥ 1.5 *cm*

VI. Objectives

Objective 1: Full preparation

Full preparation brings out a good start and helps form a general outlook on the whole project.

Objective 2: A moving system working in 3D

The system concerning movement in 3D enables the robot to climb stairs, solving the problem of stair inaccessibility.

Objective 3: A cleaning system specializing in corner-cleaning

The ability to clean the corners makes the robot capable of cleaning the stair structure effectively.

Objective 4: Intelligent automation

Working independently of manual operations can release human efforts from the tedious and repetitive work of stair cleaning.

Objective 5: Modification and improvement

Through ongoing testing and altering, the robot can be improved continuously until becoming a mature product.

VII. Tasks

Tasks are things done to realize the objectives mentioned above. In this section, there will be specific information about the tasks of this project.

Full preparation

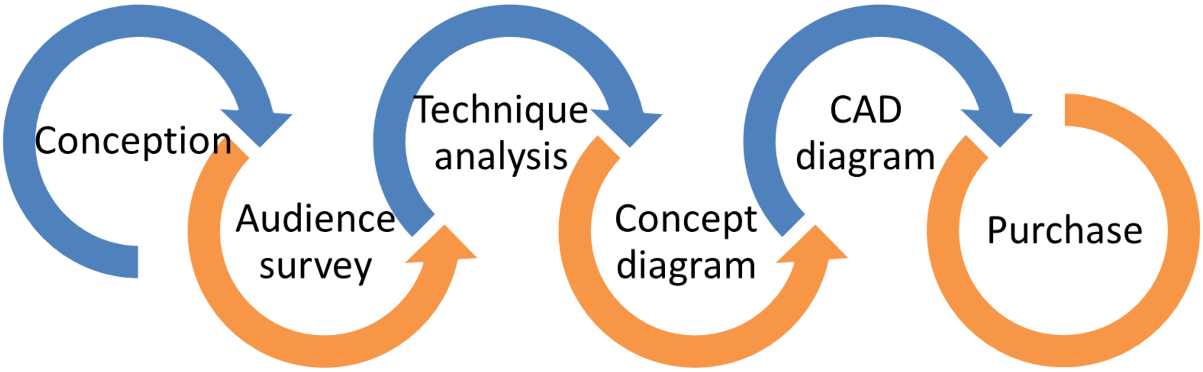


Figure VII.1 The flow chart for full preparation.

Task 1: Conceive the draft idea

To decide a well-defined goal of the project, the draft idea started from the cleaning machine and then focused on the characteristic of cleaning the stair areas. So, as shown in *Figure VII.2*, the specific goal was determined as a combination of the function of climbing stairs and the function of cleaning.

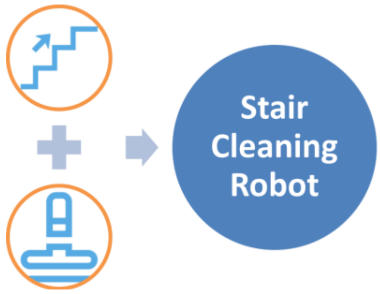


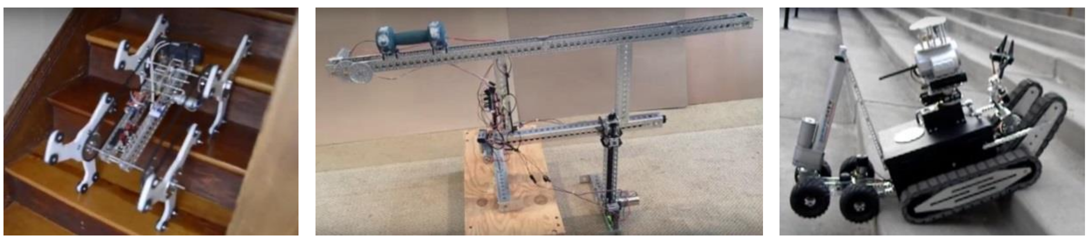
Figure VII.2 The goal of this project.

Task 2: Conduct an audience survey

For background research, a survey was conducted among our audience. From the survey, the overall comments on the robot are obtained from the audience.

Task 3: Analyze the existing techniques related to the functions

The existing techniques were studied. For the function of cleaning, there were brooms, mops, cleaning robots [2], and stair cleaners [3]. For the function of climbing stairs, there were methods of planetary wheels [16], racks and pinions [17], caterpillar treads [18] (shown in *Figure VII.3*).



a. Planetary wheels. b. Racks and pinions. c. Caterpillar treads.

Figure VII.3 The realized techniques of stair climbing [16][17][18].

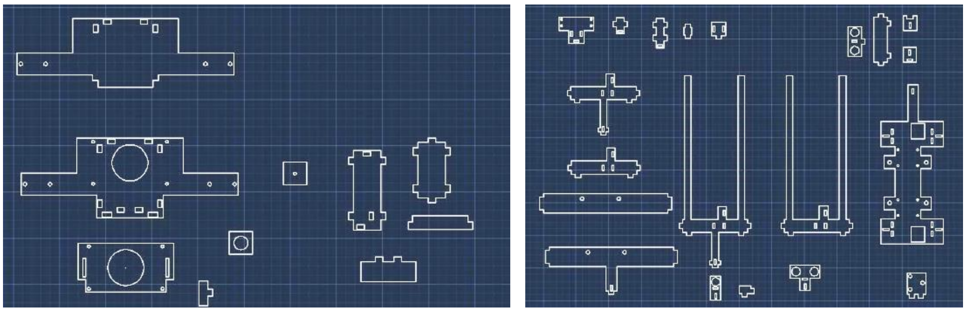
Based on the research, the general working principles were determined, as described in detail in *Section V.*

Task 4: Draw the concept diagram

The concept diagram was drawn to show the design vividly, as shown in *Section V.* Both the general and detailed conceptual designs were visualized.

Task 5: Draw the CAD diagrams

The structure components of the cleaners were specifically arranged and the exact sizes of the body of moving parts were decided. The CAD diagrams were drawn as shown respectively in *Figure VII.4*.



a. CAD of cleaning. b. CAD of moving.

Figure VII.4 The CAD diagrams.

Task 6: Purchase materials

We purchased materials online. Since our project required demanding structure, the design was being revised during the process of purchasing: properties of available materials were sometimes too limited to realize our design.

After the completion of <u>full preparation</u>:

- ✓ **The specific goal of this project was determined.**
- ✓ **The problems and needs of the target audience are thoroughly studied.**
- ✓ **The final design was finished.**
- ✓ **Material preparation was accomplished.**

A moving system working in 3D

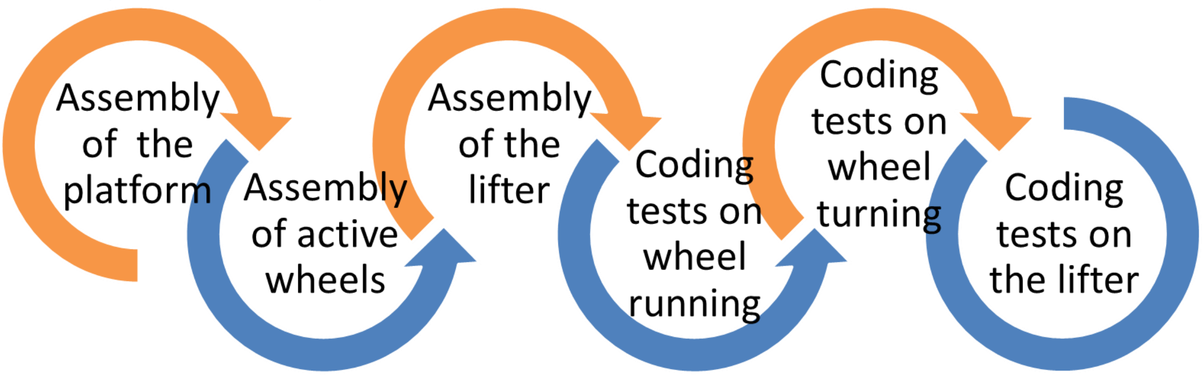


Figure VII.5 The flow chart for a moving system working in 3D.

Task 7: Assemble the platform

The platform, shown in *Figure VII.6*, was the main body of the robot. Many of he components were prepared for later connections since the platform was the supporting part for all the other parts after completing.

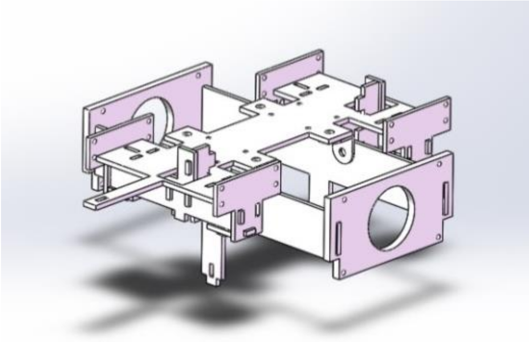


Figure VII.6 The platform.

Task 8: Assemble the active wheels

As shown in *Figure VII.7*, the four sets of wheels were added to the bottom of the platform. They were expected to make the robot run on the level ground in any direction.

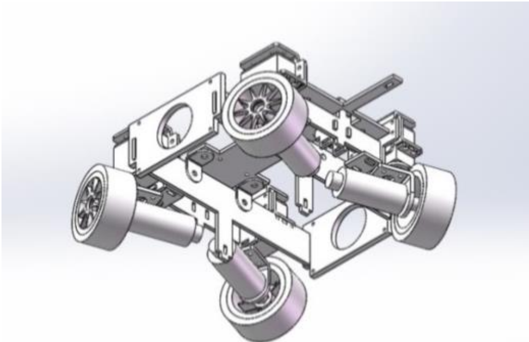


Figure VII.7 The active wheels attached.

Task 9: Assemble the lifter

As shown in *Figure VII.8,* the lifting components were attached to different parts of the robot to enable the robot to move vertically in parts.

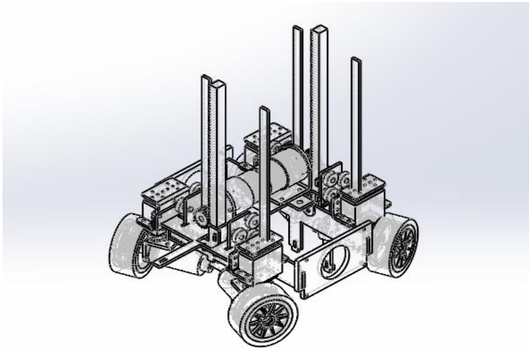


Figure VII.8 The driving and lifting parts.

Task 10: Implant codes to test wheel running

The coding was applied to the motors attached to the wheels to see whether wheels could rotate well. During this process, the encoders were also tested to coordinate the angular speeds of each wheel. Since the direction-turning was not achieved by the different angular speeds of different wheels, it was only required to ensure that wheels could rotate at the same speed.

Task 11: Implant codes to test wheel turning

The coding was applied to the servos attached to the wheels to test whether wheels could turn directions. A servo had the function of control the angular displacement. Thus, the precise control of direction turning was ensured by servos attached to the active wheels.

Task 12: Implant codes to test lifting

The coding was applied to the motors attached to the pinions to test whether racks and pinions could lift things successfully.

After the completion of <u>a moving system working in 3D</u>:

- ✓ The overall structure of moving in 3D space was finished.
- ✓ The driving part could work well.
- ✓ The lifting part could work well.

A cleaning system specializing in corner-cleaning

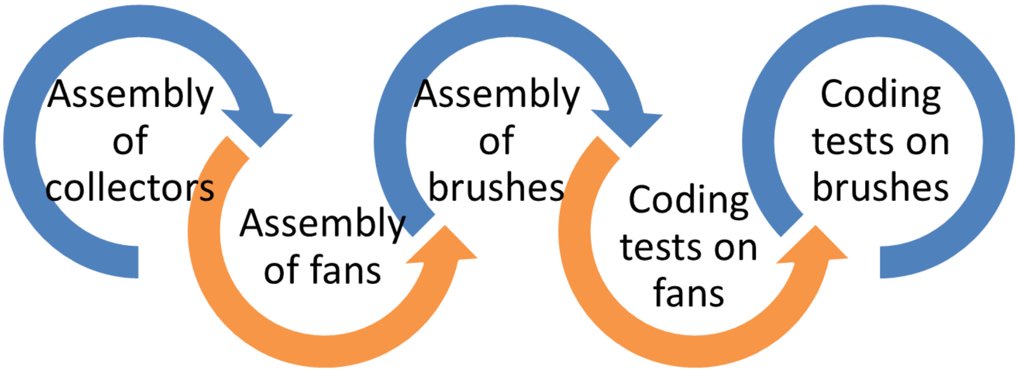


Figure VII.9 The flow chart for a cleaning system specializing in corner-cleaning.

Task 13: Assemble the rubbish collectors

The rubbish collectors were assembled as designed. They aimed to contain the collected rubbish when the cleaning work was going on.

Task 14: Assemble the electric fans

The electric fans aimed to make the gas move to form gas pressure difference. The air flows produced by the pressure difference dragged rubbish into the collector.

Task 15: Assemble the round brushes

The brushes expanded a larger range. They worked to collect rubbish all into the effective working area of the fans.

The complete cleaning part was shown in *Figure VII.10*.

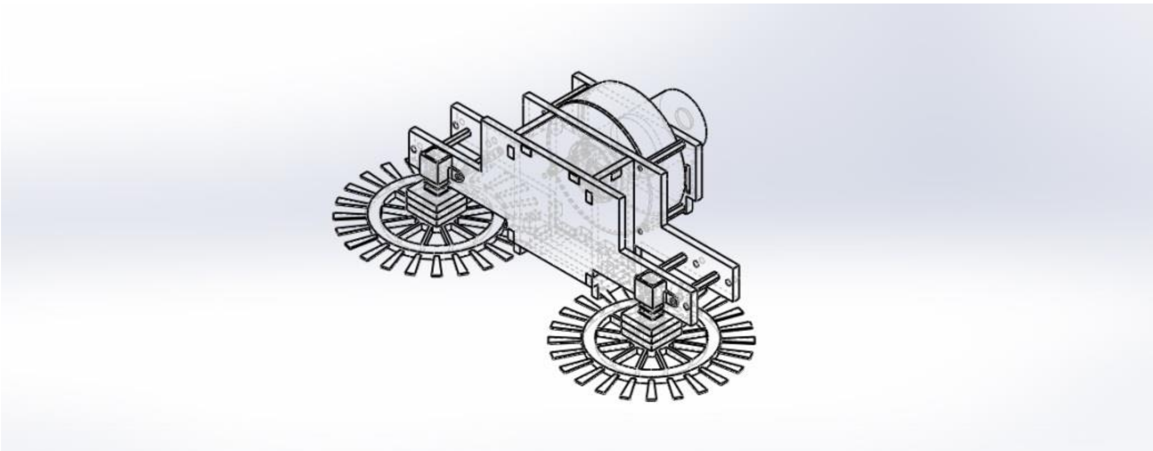


Figure VII.10 The cleaning part on one side.

Task 16: Implant codes to test fans

The coding was applied to the motors attached to the fans to see whether the fans could form air flows strong enough to draw the rubbish. The gas tightness of the cleaning system was also tested to make sure pressure difference was formed.

Task 17: Implant codes to test brushes

The coding was applied to the motors attached to the brushes to adjust the speed and direction of their rotation. The brushed needed to have a proper rotation speed to collect rubbish close to the suctions. The large range of the brushes could reach the corners in the stair structure.

After the completion of a cleaning system specializing in corner-cleaning:

- ✓ **The cleaning part could work well.**
- ✓ **The two main functions of climbing and cleaning were both basically accomplished.**

Intelligent automation

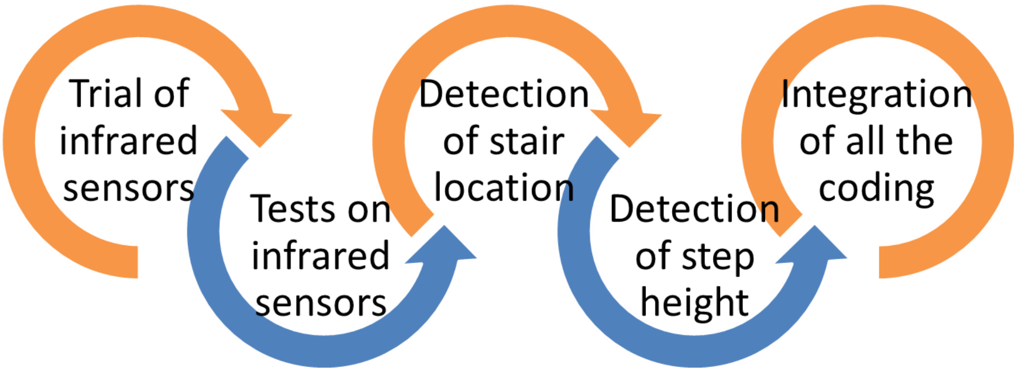


Figure VII.11 The flow chart for intelligent automation.

Task 18: Try infrared sensors

Familiarity with the sensors laid important foundation for the later realization of detecting function.

Task 19: Test infrared sensors

Test on sensors measured exact data of the sensors' capacity, which influenced the program design.

Task 20: Detect the locations of stairs

Detection of the stair location was a key in the automatic running. Stair is the target of our main function. Locating the stairs was a basic task.

Task 21: Detect the height of steps

Detection of the step height was a key in the automatic climbing. This detection would decide when to stop or continue the climbing.

Task 22: Integrate all the coding into a complete program

All the testing coding was integrated into one program. The program was revised to be a complete one, which would be the controlling program for the robot.

After the completion of <u>intelligent automation</u>:

- ✓ **The electronic part was all finished.**
- ✓ **The robot could climb the stairs automatically.**

Modification and improvement

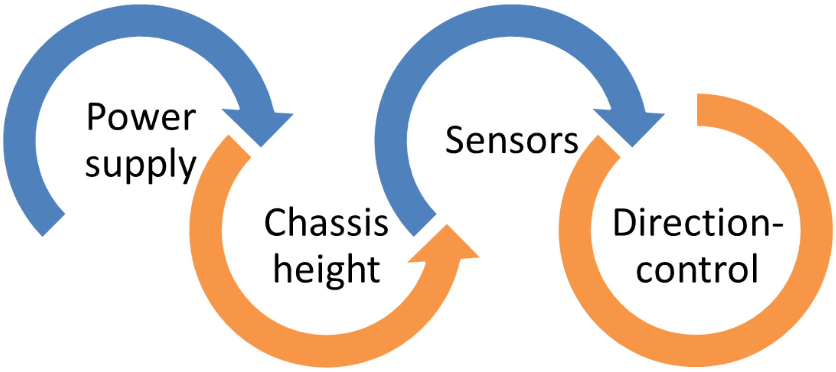


Figure VII.12 The flow chart for modification and improvement.

Task 23: Modify the supply of power

To provide more energy to the moving and cleaning system, the supply of power was changed to make sure every part worked well.

Task 24: Modify the chassis height

When climbing stairs, the robot lacked stability since our design and assembly of the robot were not precisely detailed. Thus, the chassis height needed to be continuously modified.

Task 25: Modify the sensors

When it worked, the robot sometimes appeared dull to the outside conditions. Thus, the sensors were modified to more sensitive.

Task 26: Modify the control over direction

When the active wheels changed the direction of the robot movement, the errors were bigger than expected. To perfect the performance, the control over direction had to be modified.

After the completion of <u>modification and improvement</u>:

- ✓ **The robot was all finished.**
- ✓ **The function realization was stable.**

VIII. Conclusion

In conclusion, the whole project is generally a success although we have encountered some problems and failures during the process.

A. Success

First of all, one of the successful aspects is the structure design. The complexity of structure design comes from the combination of the cleaning part, climbing part and the moving part, which is a section of the climbing part but also a separate functional part making it possible for the robot to move along the stairs while cleaning. In order to make the robot able to move in two perpendicular directions, we have designed four independent driving sections consisting, each of which consists of a servo and a driving motor. In doing so, the assembling height of each driving section is at 12 *cm*. The whole structure can be divided into majorly tow layer. The lower layer containing the moving section, while the upper layer section containing the lifting section and cleaning section. It is clear that the cleaning section in the upper layer should be close to the ground enough so that the brushes and vacuum cleaner can work efficiently. However, as mentioned before, the assembling height of each driving section is already at 12 *cm*, which is a significant distance from ground. We cannot put the cleaning section in the lower layer and the moving section in the upper section since the wheels are always the last part of the car that is lifted while climbing (without the support of the wheels on the lower stair, the climbing is impossible). To solve this problem, we have placed the cleaning part in the empty space between front wheels and back wheels. The plane with lifting motor assembled on it is the main structure of the upper layer. The beams connecting two front wheels and two back wheels are under this plane, so the minimum height of the plane (when the plane is not lifted) is larger than 12 *cm*. Therefore, there is an empty space between front and back moving section, and under the plane for us to place two beams which support the cleaning part. In this way, the cleaning part and the climbing part, as well as the moving part, can work properly without interfering each other.

Another successful aspect of our project is that during programming we have first realized every function of the device by writing a series of fundamental functions, and then have written a series of process functions corresponding to different working process, which consist of the fundamental functions. Fundamental functions including four categories: lifting, moving, turning and cleaning functions. Fundamental functions in different processes functions are slightly different; for example, the moving functions in the moving process toward the stair and moving process parallel to the stair are slightly different since the rotating direction of the wheel is not the same. The process functions include the climbing and the cleaning functions. In this way, we can first achieve the fundamental function of the device and then finished the programing for the process, which making the testing part of the project more efficient and reasonable.

Undoubtedly, automation is another success of this project. Based on the data collected from the infrared sensors and the program saved in the Arduino board, the robot is able to determine when to stop lifting, when to stop moving, and so on. The purpose of our project is to provide people with a device that can free them from the tedious and tired work of cleaning stairs, so the automation is the key of the project. Only by turning on the switch, one can get the stairs cleaning thoroughly and quickly without doing anything more.

B. Failure

Generally speaking, the project is a success; however, we indeed encountered some problems and failures during the process.

One major failure when we first conducted the test was that the plane couldn't remain horizontal while being lifted. This is because while being lifted the plane were supported only by rotating gear. The lifting motors exerted a torque (about 0.4 *N/m*) on the gear. According to Newton's third law, when something exerts a force on another, the latter one will exert a force of the same magnitude and the opposite direction on the former [19]. So, the gear exerted a force and corresponding torque with the same magnitude on the plane (or saying on the lifting motor). However, there was not a torque that together with this torque exerted by the gear on the plane could make the external torque of the plane vanish; then the plane would incline. We ignored this problem while designing and therefore it led to a failure. Fortunately, we have solved this problem by adding some acrylic strips on the original strip made of carbon fiber. This acrylic strips have made the guiding wheel move only in the vertical direction so that the plane can only move vertically. Although the plane cannot move perfectly vertically, the current small incline angle does not influence climbing at all.

Another failure is that when assembling we found out some screws interfered with the structure. For example, the screws used to fix the servo got the plane stuck. This was because we didn't add screws in 3D concept diagram so that we could not find out this kind of interference while designing. In fact, we found several interferences from the 3D concept diagram such as two lifting motor overlap each other, and we revised these designing mistake before we began to process components. Fortunately, we have solved this interfering problem by further processing the components.

C. Experience

According to these failures and problems, we conclude some experience.

While conducting a device with complex structure, assembling all the components in the 3-D concept diagram is very important; with detailed concept diagram, one can find out the design mistakes and revise them before processing and assembling the components. The static 3-D concept diagram can help us solve the interference problem between components, while adding imitating animation can help us find out if the system moves as we design, if there lacks some degree of freedom or there are some extra degree of freedom that we don't expect. It is of significant importance to check every design mistake before beginning to conduct the prototype since a failure prototype with design mistakes means needs for new prototype and therefore more unnecessary costs if the problem cannot be solved by further processing the components.

D. Suggestions for Future Improvement

Structure

In order to make the total external torque of the plane vanish as we discuss above, we can use four lifting motors of smaller size than the current ones and four corresponding racks of smaller size; the left two ones rotate in the same direction, while the right two ones rotate in the opposite direction. By doing so, we can improve the lifting stability of the robot, and, in addition, we can omit the twelve guiding wheels on the current prototype, meaning a simpler structure.

Another improvement can be made by replacing the current driving sections consisting of a servo, a motor, and a wheel with a set of a Mecanum wheel and a motor. As shown in *Figure VIII.1*, a Mecanum wheel is a kind of universal wheel: by applying different voltages on the four motors that drive four Mecanum wheels, these four wheels can rotating at the different speed, making it possible to make the device move in any direction as we expected [20]. After removing the servos, we can lower the plane so that the robot can move more stably since the center of mass is lower than before; the structure is also simplified.

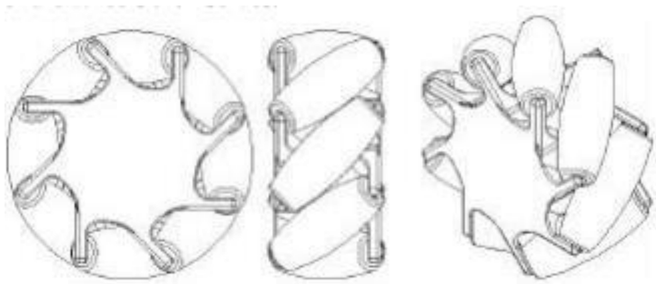


Figure VIII.1 Mecanum wheels [20].

Function

By using the Mecanum wheels, we can let the robot move in any direction we want, so that the robot not only can clean the stairs but clean the floor better. In addition, after cleaning a series of stairs, the robot can easily turn itself around and continue to clean upper series of stairs

Appearance

In order to make the robot more delicacy and improve audiences' first impression of the outlook of the product, we can design a shell cover all the mechanic and electric part of the robot. It can surly improve the user-friendliness of the product, therefore, making it more competitive.

IX. Schedule

In the beginning, it was even hard to plan out the whole schedule, since we had abstract ideas of how our product will end up. So, we allotted group meetings, hence, the idea conception and preparation period occupied a lot in our schedule. However, after going through trials and ordering the parts that were needed for our product, we were able to make the parts fast. Wheels, motors, and servos, platform, racks, and pinions, brushes, fans and dust box were made relatively quickly. However, after the parts were all made, it was hard to coordinate them all together. Assembly and modification also took us considerable efforts. However, we succeeded in completing our stair cleaning robot within time and under schedule.

The schedule of preparation is shown in *Figure IX.1.*

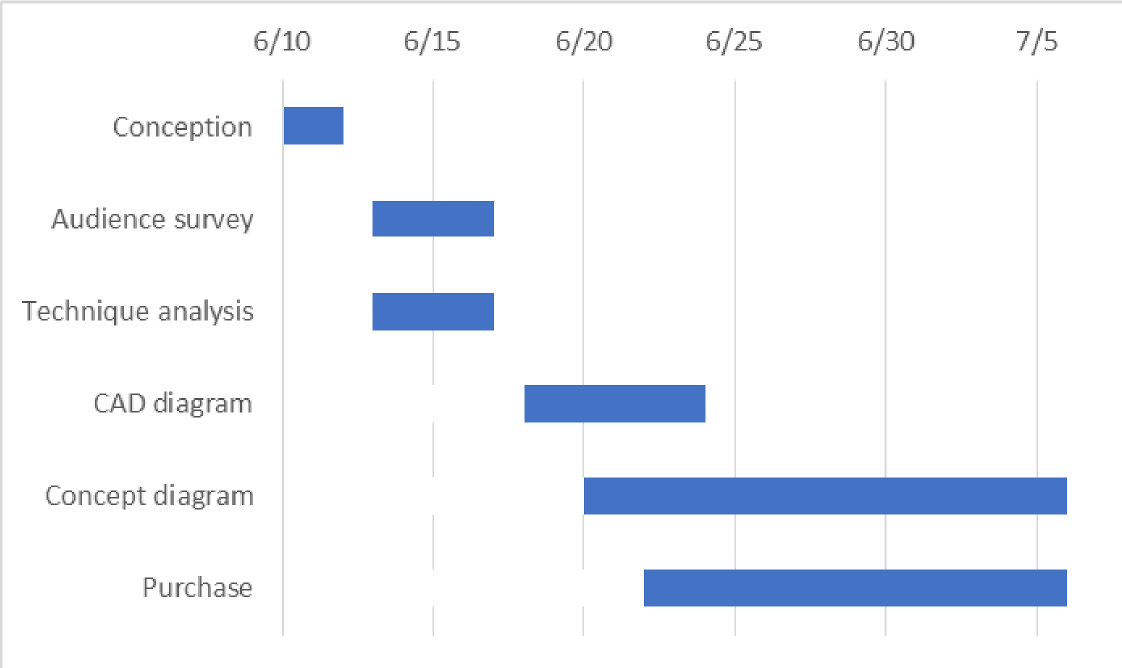


Figure IX.1 The Gantt chart of full preparation.

The selection of idea was hard in the beginning. By the help of discussion with our professors and the TA, we settled to the current idea of Stair Cleaning Robot within few days (June 10~13). Afterwards we did the audience survey (June 13~17). To know the need of the product and by working parallel with technical analysis, we made the big picture of our product. After the general picture was made, we drew our expected product by CAD diagrams and concept diagrams and later started to purchase.

The schedule of the moving structure is shown in *Figure IX.2.*

As the parts started to arrive, we started to make the parts on July 1. We started to make the main platform and the active wheels and finished both on July 3. The lifter took us a lot of time to make it properly function until July 8. Afterwards, this was the part that we had mainly refined. We also started the coding of the active wheel turning and the active wheel running. Afterwards, we started the lifting coding.

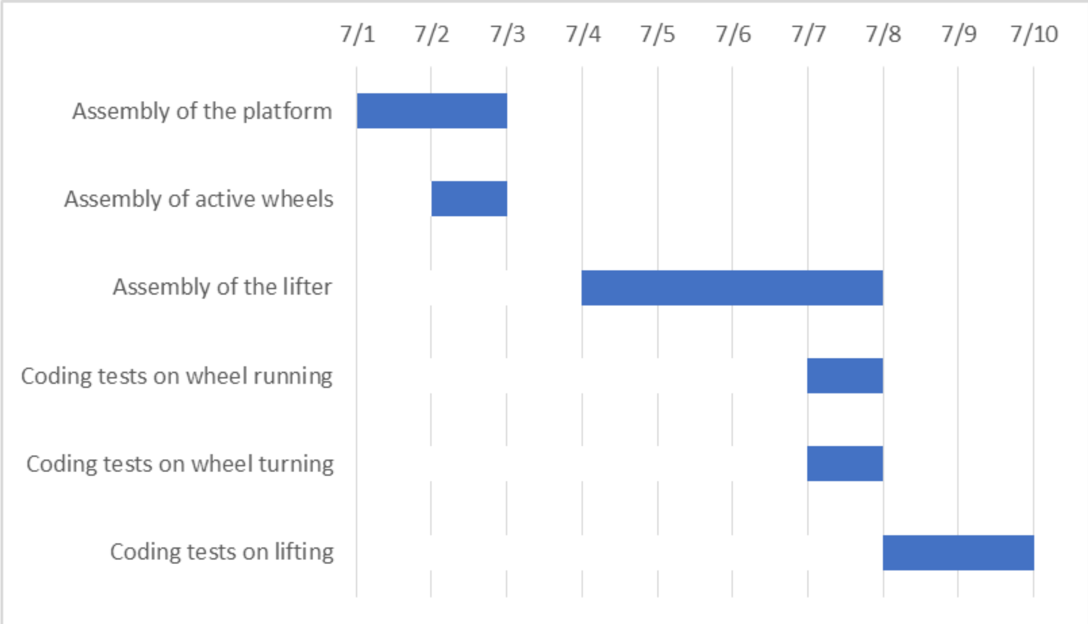


Figure IX.2 The Gantt chart of a moving system working in 3D.

The schedule of the cleaning structure is shown in *Figure IX.3.*

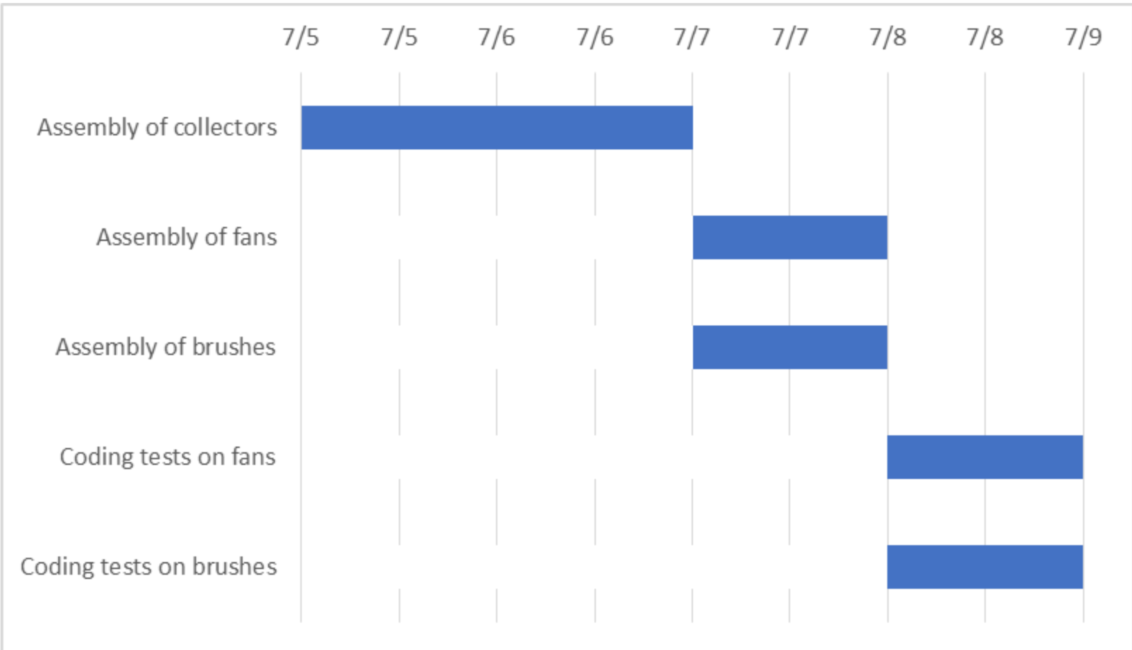


Figure IX.3 The Gantt chart of a cleaning system specializing in corner-cleaning.

In the cleaning part, the assembly of the collectors took the most time (July 4~7) because it had to coincide with the main platform. Afterwards, we assembled the fans and the brushes (July 7~8). Then, we were able start coding to test the fans and brushes that were assembled (July 8~9). The cleaning parts took us less time than the main mechanical part.

The schedule of automation is shown in *Figure IX.4.*

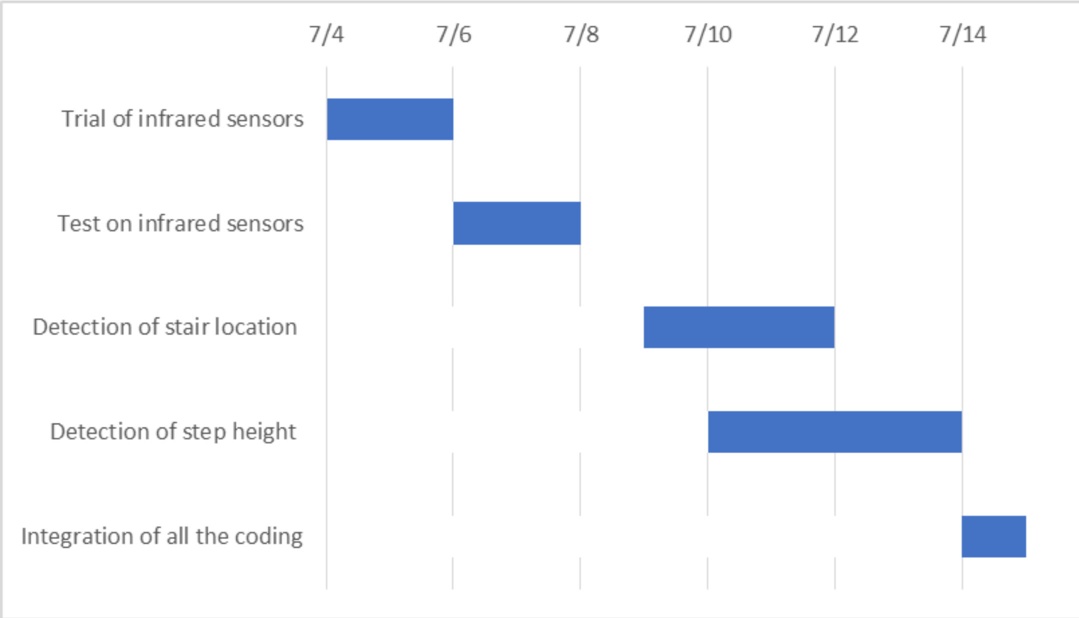


Figure IX.4 The Gantt chart of intelligent automation.

After we had assembled most of the parts, we started to try the sensors, test their functions and install them (July 4~8). Then, we focused on the stair. Because every stair has their own unique characteristic, we needed sensors to adapt the robot to various surroundings. In order for our robot to function correspondingly to the stair, we had to detect the stairs precisely and accurately. After that, we integrated our coding altogether.

The schedule of modification is shown in *Figure IX.5.*

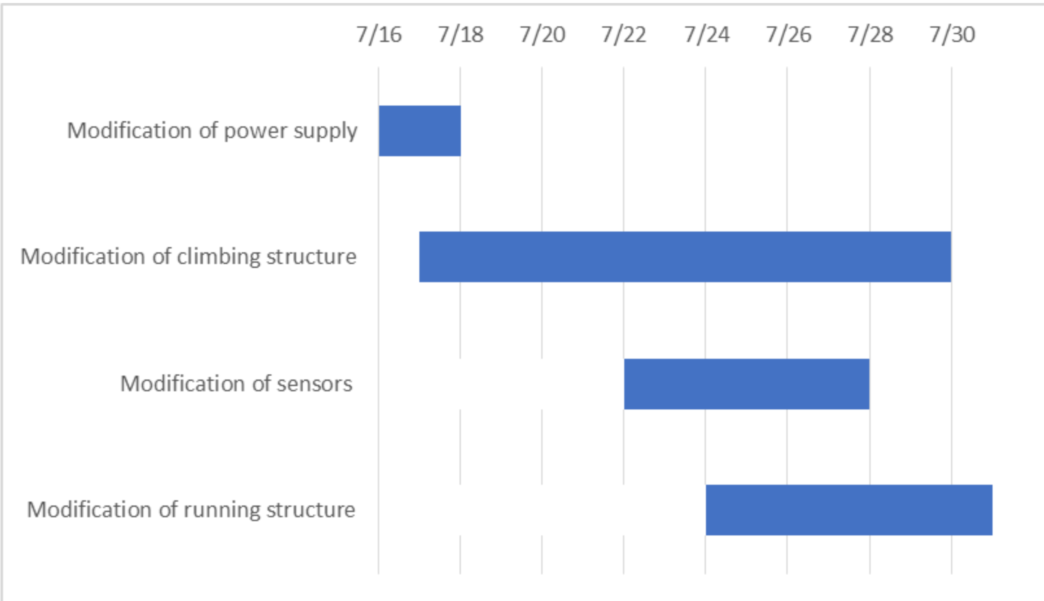


Figure IX.5 The Gantt chart of modification and improvement.

After all parts were put together and the general coding was done, we made the modification plans. Through the modification, we have made our product more organized and sophisticated. As mentioned before, the modification of the climbing structure took us the most time. Afterwards we managed to accomplish our Stair Cleaning Robot.

X. Budget

As shown in *Figure 10.1*, the total budget is 980 RMB, which is less then 1000 RMB. The total budget is higher than other teams in this summer VG100. However, compared to the other automatic vacuum cleaners available, the prices of which start from 1400 RMB with a wide range [2], our product is highly compatible.

Materials	Price
Supporting structures	¥450
Racks & pinions and bearings & bars	¥ 25
Servos, motors and wheels	¥300
Brushes	¥ 15
Electric fans	¥ 50
Electric boards	¥120
Sensors	¥ 15
Wires	¥ 5

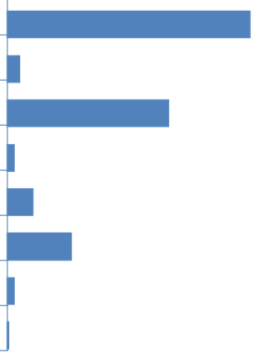


Figure X.1 The budget table with a bar chart for Stair Cleaning Robot.

As we look one by one into the *Figure X.1*, we can find out that supporting structures consumes the highest proportion in the whole budget. It is because we have used carbon fiber for the strenth and supportablity. And the weight of it is smaller than the other materials that we can currently use. If we can have a more alternative for the materials, we may lower the budget.

Because of the complexity of our product, there is a huge number of motors and servos constructed in this product. And the parts that we have assemble had to be reliable. So, servos, motors and wheels took much.

In conclusion, the total budget for our project remains reasonable and accepatable.

XI. Key Personnel

As shown in *Figure XI.1*, many members in our team are responsible for more than one task.

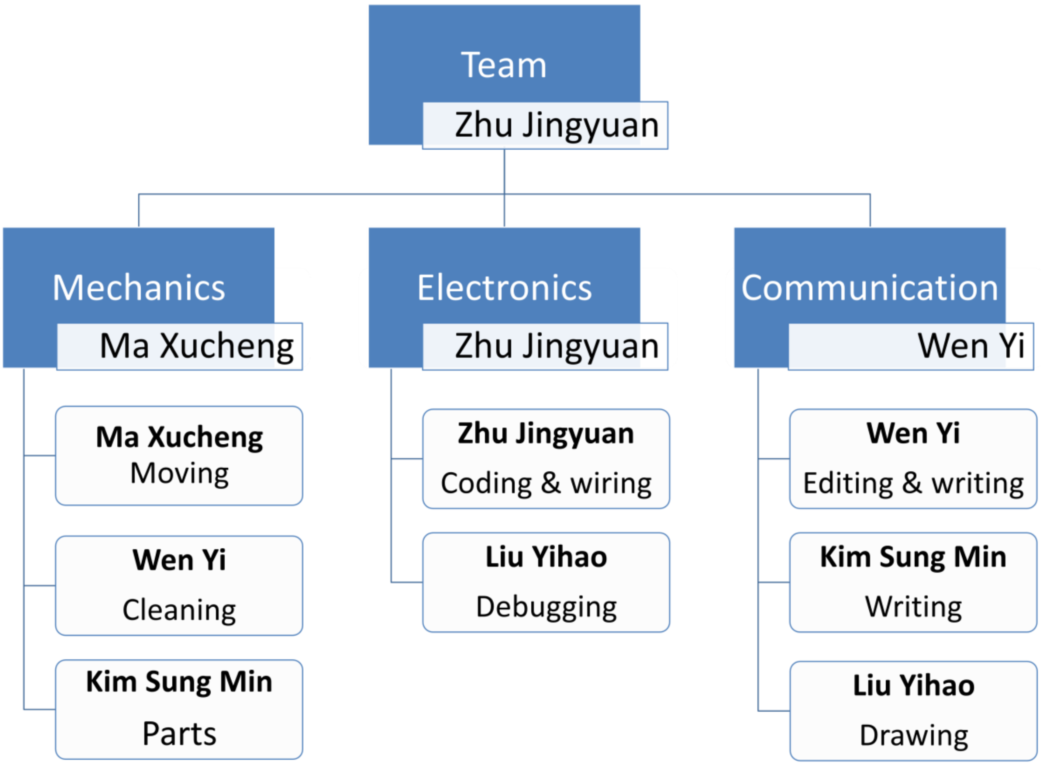


Figure XI.1 The organizational tree of A Group for Stair Cleaning Robot.

Zhu Jingyuan is our team leader and he has successfully managed and led our team. He is also mainly in charge of the electronics part with Liu Yihao. Zhu Jingyuan has done the coding and wiring and Liu Yihao has done the debugging. They have sucessfuly coded the program implanted on our Stair Cleaning Robot.

Ma Xucheng has managed most of the hardware parts of our Stair Cleaning Robot. He has successfully assembled the parts together so that the robot can serve its purpose. Many delicate movements of the robot are able because of his hard work. Wen Yi and Kim Sung Min have assisted him.

For the communication part, Wen Yi has done a fabulus work. Wen Yi and Kim Sung Min have done the writing and throgout editing. Liu Yihao has drawn the conceptual diagrams to help communication when explaining our product to others.

XII. References

[1] The dezeen magazine, 20 January 2016, "Record number of skyscrapers completed in 2015," *Dezeen* website, accessed at http://www.dezeen.com/2016/01/20/record-number-skyscrapers-completed-2015-council-tall-buildings-urban-habitat-ctbuh/ on August 2, 2016.

[2] Best Vacuum Cleaners for 2016, 2016, "Stairs," *Vacuum Cleaner Reviews and more* website, accessed at http://vacuumcleanersv.com/stairs/ on August 2, 2016.

[3] Purch, 2016, "The Best Robot Vacuums of 2016," *Top Ten Reviews* website, accessed at http://robot-vacuum-review.toptenreviews.com/ on August 2, 2016.

[4] Morgan, P., 2005, *Carbon Fibers and Their Composites*, CRC Press, Boca Raton, pp.791-859, Chap. 20.

[5] Ministry of Construction of the People's Republic of China, 2005, *Code for Design of Civil Buildings*, Ch. 6.7.

[6] Taobao, 2016, "An Electric fan with a Housing," *Taobao* website, accessed at https://item.taobao.com/item.htm?spm=a1z09.2.0.0.J4mDOO&id=530237242952 on August 3, 2016.

[7] WordPress, 2016, "JGA25-371 DC Gearmotor with Encoder," *Open Impulse* website, accessed at https://www.openimpulse.com/blog/products-page/25d-gearmotors/jga25-371-dc-gearmotor-encoder-126-rpm-12-v-2/ on August 3, 2016.

[8] Fraden, J., 2010, *Handbook of Modern Sensors*, Springer Science & Business Media, New York, pp. 293-297, Chap 7.3.4.

[9] Electronicos Caldas, 2013, "MG996R High Torque Metal Gear Dual Ball Bearing Servo," *Electronicos Caldas* website, accessed at www.electronicoscaldas.com/datasheet/MG996R_Tower-Pro.pdf on August 3, 2016.

[10] Murthy, S. T., 2010, *Textbook of Elements of Mechanical Engineering*, I. K. International Pvt Ltd, New Delhi, pp. 213, Chap.18.10.

[11] Seeed, 2016, "ASLONG JGB37-520 12V DC Worm Gear Motor," *Seeed Studio* website, accessed at http://www.seeedstudio.com/depot/ASLONG-JGB37520-12V-DC-Worm-Gear-Motor-p-1888.html on August 3, 2016.

[12] Fraden, J., 2010, *Handbook of Modern Sensors*, Springer Science & Business Media, New York, pp. 310-314, Chap 7.4.6.

[13] Arduino, 2016, "Arduino Board Due," *Arduino* website, accessed at https://www.arduino.cc/en/Main/ArduinoBoardDue on August 3, 2016.

[14] Reichenstein7, 2014, "Arduino Modules - L298N Dual H-Bridge Motor Controller," *Instructables* website, accessed at http://www.instructables.com/id/Arduino-Modules-L298N-Dual-H-Bridge-Motor-Controll/ on August 4, 2016.

[15] Yoshio, M., Brodd, R. J., Kozawa, A., 2009, *Lithium-Ion Batteries*, Springer Science & Business Media, New York, pp. 1-7, Chap. 1.

[16] TeamJaeger, 2015, "Stair Climbing Robot," *Instructables* website, accessed at http://www.instructables.com/id/Stair-Climbing-Robot-1/ on August 4, 2016.

[17] christopherbegley, 2015, "Stair Climbing Robot," *Instructables* website, accessed at

http://www.instructables.com/id/Stair-climbing-robot/ on August 4, 2016.
[18] chad.giddings.71, 2015, "Robot," *Instructables* website, accessed at http://www.instructables.com/id/robot-9402/ on August 4, 2016.
[19] Young, H. D., Freedman, R. A., 2012, *University Physics*, Pearson Education, Inc., San Francisco, pp. 120-123, Chap. 4.5.
[20] Diegel, O., Badve, A., Bright, G., Potgieter, J., Tlale, S., 2002, "Improved Mecanum Wheel Design for Omni-directional Robots," *Australasian Conference on Robotics and Automation*, Auckland, pp. 117-121.

XIII. Appendix

The integrated program implanted on Stair Cleaning Robot is attached below.

```
#include <Servo.h>

/* Declarations */

enum POSITION
{
    LF, RF, LB, RB
};

void countLF(), countRF(), countLB(),
countRB();

void (*__count[])() = {countLF, countRF,
countLB, countRB};

class Motor
{
public:
    Motor(POSITION, int, int, int, int,
int);

    void (*countFunc)();

    void count();

    int change(int, int, int);

    int oldtime, time, c, val;
    int pinInter, pinServo, pin1, pin2,
pinA;
    Servo servo;
};


Motor::Motor(POSITION pos, int
pinInter, int pinServo, int pin1, int
pin2, int pinA)
{
    this->pinInter = pinInter;
    this->pinServo = pinServo;
    this->pin1 = pin1;
    this->pin2 = pin2;
    this->pinA = pinA;

this->servo.attach(this->pinServo);
    this->oldtime = this->time =
this->c = 0;
    this->val = 90;
    this->countFunc = __count[pos];
    attachInterrupt(this->pinInter,
this->countFunc, FALLING);
}

void Motor::count()
{
    if ((millis() - this->time) > 5)
    {
        this->c++;
    }
    this->time = millis();
}

int Motor::change(int pin, int want,
int val_A)
{
    if (millis() - this->oldtime >= 1000)
    {
        detachInterrupt(pin);
        int rpm = this->c * 60 / 825;
        int rpm1 = (want - rpm);
        val_A = val_A + rpm1 * 2.5;
        if (val_A > 255)
        {
            val_A = 255;
        }
        if (val_A < 0)
        {
            val_A = 0;
        }
        this->oldtime = millis();
        this->c = 0;
        attachInterrupt(pin,
this->countFunc, FALLING);
    }
    return val_A;
}

Motor *motor[4];

#define COUNT_FUNC(pos) void
count##pos(){motor[pos]->count();}

COUNT_FUNC(LF)

COUNT_FUNC(RF)

COUNT_FUNC(LB)

COUNT_FUNC(RB)
```

```
int keyLR = 0;
bool light;
int stateIR = 0;
float distance;
int stateD = 0;
int stateD2 = 0;
int stateD3 = 0;
int stateD4 = 0;
int stateD5 = 0;

const int trigFor = 20;
const int EchoFor = 21;
const int trigBac = 14;
const int EchoBac = 15;
const int trigL = 26;
const int EchoL = 27;
const int trigR = 28;
const int EchoR = 39;
const int trigUp = 10;
const int EchoUp = 11;

const int Forw1 = 32;
const int Forw2 = 33;
const int Backw1 = 34;
const int Backw2 = 35;

int ForA = 10;
int BackA = 10;


const int VacL1 = 12;
const int VacR1 = 13;
const int VacL2 = 28;
const int VacR2 = 29;

const int IRpin = A10;
const int IRpinBac = A5;

/* Functions */

void setup()
{
    Serial.begin(9600);
    motor[LF] = new Motor(LF, 45, 22, 2,
3, 18);
    motor[RF] = new Motor(RF, 46, 23, 4,
5, 19);
    motor[LB] = new Motor(LB, 47, 24, 6,
7, 20);
    motor[RB] = new Motor(RB, 48, 25, 8,
9, 21);
    pinMode(IRpin, INPUT);
    pinMode(IRpinBac, INPUT);
    pinMode(trigFor, OUTPUT);
    pinMode(EchoFor, INPUT);
    pinMode(trigBac, OUTPUT);
    pinMode(EchoBac, INPUT);
    pinMode(trigUp, OUTPUT);
    pinMode(EchoUp, INPUT);
    for (int i = 2; i < 11; i++)
    {
       pinMode(i, OUTPUT);
    }
    for (int i = 31; i < 36; i++)
    {
       pinMode(i, OUTPUT);
    }
    for (int i = 22; i < 26; i++)
    {
       pinMode(i, OUTPUT);
    }
    for (int i = 38; i < 46; i++)
    {
       pinMode(i, OUTPUT);
    }
    pinMode(VacL1, OUTPUT);
    pinMode(VacL2, OUTPUT);
    pinMode(VacR1, OUTPUT);
    pinMode(VacR2, OUTPUT);
    digitalWrite(Forw1, LOW);
    digitalWrite(Forw2, LOW);
    digitalWrite(Backw1, LOW);
    digitalWrite(Backw2, LOW);
}

void delayy(int times)
{
    float initime = millis();
    while (millis() - initime <= times)
    {
    }
    return;
}

void delayM(int times)
{
    float initime = micros();
    while (micros() - initime <= times)
    {
    }
    return;
}

float ultrad(int trig, int Echo)
{
    float cm;
    digitalWrite(trig, LOW);
    delayM(2);
    digitalWrite(trig, HIGH);
    delay(10);
```

```
    digitalWrite(trig, LOW);
    cm = pulseIn(Echo, HIGH) / 58.0;
    return cm;
}

float IRd(int IRpin)
{
    float tmp;
    tmp = analogRead(IRpin);
    tmp = (10787.0 / ((float) tmp - 3.0))
- 4.0;
    return tmp;
}

bool IR(int IRpin)
{
    int lightEx = analogRead(IRpin);
    Serial.println(lightEx);
    if (lightEx < 950)
    {
        return true;
    }
    else
    {
        return false;
    }
}

void Lifting()
{
    digitalWrite(Forw1, HIGH);
    digitalWrite(Forw2, LOW);
    digitalWrite(Backw1, HIGH);
    digitalWrite(Backw2, LOW);
}

void LiftStop()
{
    delayy(500);
    digitalWrite(Forw1, LOW);
    digitalWrite(Forw2, LOW);
    digitalWrite(Backw1, LOW);
    digitalWrite(Backw2, LOW);
}

void LiftFor()
{
    digitalWrite(Forw1, LOW);
    digitalWrite(Forw2, HIGH);
}

void LiftBack()
{
    digitalWrite(Backw1, LOW);
    digitalWrite(Backw2, HIGH);
}

void DownBack()
{
    digitalWrite(Backw1, HIGH);
    digitalWrite(Backw2, LOW);
}

void DownFor()
{
    digitalWrite(Forw1, HIGH);
    digitalWrite(Forw2, LOW);
}

void Move()
{
    for (int i = 0; i < 4; i++)
    {
        motor[i]->val =
motor[LF]->change(motor[i]->pinInter,
50, motor[i]->val);
    }
    for (int i = 0; i < 4; i++)
    {
        analogWrite(motor[i]->pin1,
motor[i]->val);
        digitalWrite(motor[i]->pin2,
LOW);
    }
}


void MoveBack()
{
    for (int i = 0; i < 4; i++)
    {
        motor[i]->val =
motor[LF]->change(motor[i]->pinInter,
80, motor[i]->val);
    }
    for (int i = 0; i < 4; i++)
    {
        analogWrite(motor[i]->pin2,
motor[i]->val);
        digitalWrite(motor[i]->pin1,
LOW);
    }
}


void MoveHigh()
{
    for (int i = 0; i < 4; i++)
    {
        motor[i]->val =
```

```
motor[LF]->change(motor[i]->pinInter, 80, motor[i]->val);
    }
    for (int i = 0; i < 4; i++)
    {
        analogWrite(motor[i]->pin1, motor[i]->val);
        digitalWrite(motor[i]->pin2, LOW);
    }
}

void MoveLeft()
{
    for (int i = 0; i < 4; i++)
    {
        motor[i]->val =
motor[LF]->change(motor[i]->pinInter, 30, motor[i]->val);
    }
    for (int i = 0; i < 4; i++)
    {
        analogWrite(motor[i]->pin2, motor[i]->val);
        digitalWrite(motor[i]->pin1, LOW);
    }
}

void MoveRight()
{
    for (int i = 0; i < 4; i++)
    {
        motor[i]->val =
motor[LF]->change(motor[i]->pinInter, 30, motor[i]->val);
    }
    for (int i = 0; i < 4; i++)
    {
        analogWrite(motor[i]->pin1, motor[i]->val);
        digitalWrite(motor[i]->pin2, LOW);
    }
}

void MoveStop()
{
    for (int i = 0; i < 4; i++)
    {
        digitalWrite(motor[i]->pin1, LOW);
        digitalWrite(motor[i]->pin2, LOW);
    }
}

void LiftDown()
{
    digitalWrite(Forw1, LOW);
    digitalWrite(Forw2, HIGH);
    digitalWrite(Backw1, LOW);
    digitalWrite(Backw2, HIGH);
}

void test()
{
    Turning(0);
    if (IR(IRpin) == true)
    {
        LiftDown();
    }
    else
    {
        LiftStop();
    }
}

void Turning(int state)
{
    if (state == 0)
    {
        motor[LF]->servo.write(10);
        motor[RF]->servo.write(165);
        motor[LB]->servo.write(180);
        motor[LB]->servo.write(17);
        state = 1;
    }
    else
    {
        motor[LF]->servo.write(97);
        motor[RF]->servo.write(73);
        motor[LB]->servo.write(92);
        motor[LB]->servo.write(105);
        state = 0;
    }
}

void Climbing()
{
    stateD = 0;
    stateD2 = 0;
    stateD3 = 0;
```

```
stateIR = 0;
stateD4 = 0;
Turning(0);
delayy(2000);
light = IR(IRpin);
while (light == true || stateIR < 48)
{
    Lifting();
    if (light == false)
    {
        stateIR++;
    }
    else
    {
        stateIR = 0;
    }
    light = IR(IRpin);
}
delayy(500);
LiftStop();
delayy(2000);
Move();
delayy(500);
MoveStop();
delayy(2000);
LiftFor();
delayy(8000);
LiftStop();
delayy(500);
analogWrite(motor[LF]->pin1, 90);
digitalWrite(motor[LF]->pin2, LOW);
analogWrite(motor[RF]->pin1, 120);
digitalWrite(motor[RF]->pin2, LOW);
analogWrite(motor[LB]->pin1, 90);
digitalWrite(motor[LB]->pin2, LOW);
analogWrite(motor[RB]->pin1, 90);
digitalWrite(motor[RB]->pin2, LOW);
delayy(500);
MoveStop();
delayy(500);
analogWrite(motor[LB]->pin1, 80);
digitalWrite(motor[LB]->pin2, LOW);
analogWrite(motor[RB]->pin1, 80);
digitalWrite(motor[RB]->pin2, LOW);
delayy(200);
MoveStop();
delayy(2000);
Lifting();
delayy(10);
LiftStop();
delayy(1000);
Move();
delayy(1200);
MoveStop();
delayy(500);
    analogWrite(motor[LF]->pin2, 80);
    digitalWrite(motor[LF]->pin1, LOW);
    analogWrite(motor[RF]->pin2, 80);
    digitalWrite(motor[RF]->pin1, LOW);
    analogWrite(motor[LB]->pin2, 80);
    digitalWrite(motor[LB]->pin1, LOW);
    analogWrite(motor[RB]->pin2, 80);
    digitalWrite(motor[RB]->pin1, LOW);
    delayy(500);
    MoveStop();
    delayy(1000);
    LiftBack();
    delayy(8000);
    digitalWrite(Forw1, LOW);
    digitalWrite(Forw2, LOW);
    digitalWrite(Backw1, LOW);
    digitalWrite(Backw2, LOW);
    delayy(1000);
    delayy(500);
    light = IR(IRpin);
    stateIR = 0;
    while (light == false || stateIR <
35)
    {
        MoveHigh();
        if (light == false)
        {
            stateIR++;
        }
        else
        {

            stateIR = 0;
        }
        light = IR(IRpin);
    }
    MoveStop();
    delayy(2000);
}


void GoesDown()
{
    delayy(500);
    Prepare(0);
    delayy(500);
    MoveBack();
    delayy(1600);
    MoveStop();
    delayy(500);
    DownBack();
    delayy(7100);
    LiftStop();
    delayy(500);
    MoveBack();
```

```
    delayy(2300);
    MoveStop();
    delayy(500);
    DownFor();
    delayy(7000);
    LiftStop();
    delayy(500);
    MoveBack();
    delayy(400);
    MoveStop();
    delayy(500);
    analogWrite(motor[LB]->pin1, 90);
    digitalWrite(motor[LB]->pin2, LOW);
    analogWrite(motor[RB]->pin1, 90);
    digitalWrite(motor[RB]->pin2, LOW);
    delayy(200);
    MoveStop();
    delayy(500);
    LiftDown();
    delayy(7000);
    LiftStop();
    delayy(500);
    Move();
    delayy(200);
    MoveStop();
    delayy(500);
}

void Vacuum()
{
    analogWrite(VacL1, 200);
    digitalWrite(VacL2, LOW);
    analogWrite(VacR1, 200);
    digitalWrite(VacR2, LOW);
}

void VacuumStop()
{
    digitalWrite(VacL1, LOW);
    digitalWrite(VacL2, LOW);
    digitalWrite(VacR1, LOW);
    digitalWrite(VacR2, LOW);
}

void Brush()
{
    digitalWrite(38, HIGH);
    digitalWrite(39, LOW);
    digitalWrite(40, HIGH);
    digitalWrite(41, LOW);
    digitalWrite(42, HIGH);
    digitalWrite(43, LOW);
    digitalWrite(44, HIGH);
    digitalWrite(45, LOW);
}

void BrushStop()
{
    digitalWrite(38, LOW);
    digitalWrite(39, LOW);
    digitalWrite(40, LOW);
    digitalWrite(41, LOW);
    digitalWrite(42, LOW);
    digitalWrite(43, LOW);
    digitalWrite(44, LOW);
    digitalWrite(45, LOW);
}

void Prepare(int state)
{
    if (state == 0)
    {
        LiftDown();
        delayy(200);
        LiftStop();
        delayy(1000);
        Turning(0);
        delayy(1000);
        Lifting();
        delayy(60);
        LiftStop();
    }
    else if (state == 1)
    {
        LiftDown();
        delayy(200);
        LiftStop();
        delayy(1000);
        Turning(1);
        delayy(1000);
        Lifting();
        delayy(60);
        LiftStop();
    }
}

void Cleaning()
{
    Vacuum();
    Brush();
}

void CleaningStop()
```

```
{
    VacuumStop();
    BrushStop();
}


void Whole()
{
    Climbing();
    delayy(500);
    Prepare(1);
    delayy(500);
    Cleaning();
    MoveLeft();
    delayy(6000);
    MoveStop();
    CleaningStop();
    delayy(1000);
    Prepare(0);
    delayy(500);
    Move();
    delayy(200);
    MoveStop();
    delayy(500);

    Climbing();
    delayy(500);
    Prepare(1);
    delayy(500);
    Cleaning();
    MoveRight();
    delayy(6000);
    MoveStop();
    CleaningStop();
}


void loop()
{
    Prepare(1);
    delayy(500);
    MoveRight();
    delayy(8000);
    MoveStop();
    delayy(500);
    Prepare(0);
    delayy(500);
    Whole();
    delayy(1000);
    Climbing();
    delayy(500);
    Prepare(1);
    delayy(500);
    MoveLeft();
    delayy(8000);
    MoveStop();
    delayy(500);
    GoesDown();
    delayy(500);
    GoesDown();
    delayy(500);
    GoesDown();
    delayy(500);
}
```