ENGD3000 Individual Project

Final Year Project

2016-2017

Intelligent Household System Based on Arduino

CHAOYUN GONG

P15023824

Supervisor: Seng Chong

Data:04/05/2017

Abstract

This project will develop an intelligent household system, it uses Arduino UNO to control the system and uses Arduino IDE to write the program for the system.

This system includes three devices (smart car, the lighting module, the heating module). The smart car is used to monitor the house and control all the system, it can be controlled by mobile phone. The lighting module will turn on the light if the room is dark and turn off if the room is bright when the smart car comes. The heating module will push the button of the heater when the temperature is low. It can solve a bid problem for users who live in accommodation because the heater in accommodation can just last 15 minutes to 2 hours, it is very cold in the winter night. There is the modular design (MD) to add more modules in further.

The whole system framework and the different modules which be used will be introduced in the Design part. The mechanical implement and program for these three devices will be explained in the Implement part respectively.

Acknowledgements

During the one-year project life in De Montfort University, I received many bits of help from a lot of people that we would like to say thanks to them.

First of all, I would like to say that it has been my honor to be a student of DeMontfort University. and also I would like to thank my final year project supervisor, who is Dr. Chong. Dr. Chong provided the guidance, support, and encouragement to me throughout the whole duration while I was doing and preparing my final year project (FYP).

I also want to thank the teachers who are in the Queens Building Labs. I always do my project in the Labs, when I have problems or I need some material, they always try their best to help me. Especially thanks for the teacher in Mechanical Lab, he helped me cut my board and polish it.

Last but definitely not least, I would like to thank my parents and friends. My parents support me to study in the UK. My friends always stay with me and give me support.

From the bottom of my heart THANK you all very much.

Contents

List of figures

List of table

1 The Introduction

The aim of this project is to design an intelligent household system which can be used to solve some problem and have fun.

The main part of the system is a smart car, it is controlled by computer or Android. And there is a camera module in this smart car which is used to transmit back the image to master. It can be a member of this family. Adults can monitor other rooms of the house by the smart car and children can play with it. The other parts of the system is a lighting module and heating module. They are in one room of the house. When the smart car comes into this room, the light sensor and the temperature sensor in it will monitor the light and temperature of this room. If the room is dark, the smart car will control the lighting module to open the light, if the room is cold, the smart car will control will control the heating module to open the heating. And in the night, the heating module can hold the heating to keep warm for this room, if the temperature is higher than the maximum value, the smart car will send a message to the heating module to turn off the heating. And the heating module is attached to the heating button, it will push the button automatically.

This intelligent system is based on Arduino. There are three Arduino UNO used for control centers respectively in smart car, lighting module and heating module. And the program is written by using Arduino software.

2 The background and literature research

2.1 The background

Now more and more household appliances become intelligent. The advances in technology make people's lives more convenience. For example, the Roomba which is published by iRobot company is been used to clean the floor for a personal house; the intelligent television and fridge. Those household robot is very popular in the young families and in the big cities' flats. By the development of the city and more young people will have their own house and families, the intelligent household appliances will have a wonderful prospect.

To develop a robot for cleaning is difficult to compete with Roomba because it has been a product for more than ten years and experienced many times improvements. So this project is committed to creating a member of the family. It is suitable for the young family with babies or pets. When they sit on the sofa and watch the television in the sitting room or bedroom, they can control the smart car to monitor the baby or another place in the house, like the personal garden.

The lighting module is placed in several rooms of the house. At night, when the little baby walks to other dark room, they cannot open the light by himself, so the parents can control the smart car to follow the baby to open the light of the dark room. It also can keep the room light when master has left the house to keep the thief away.

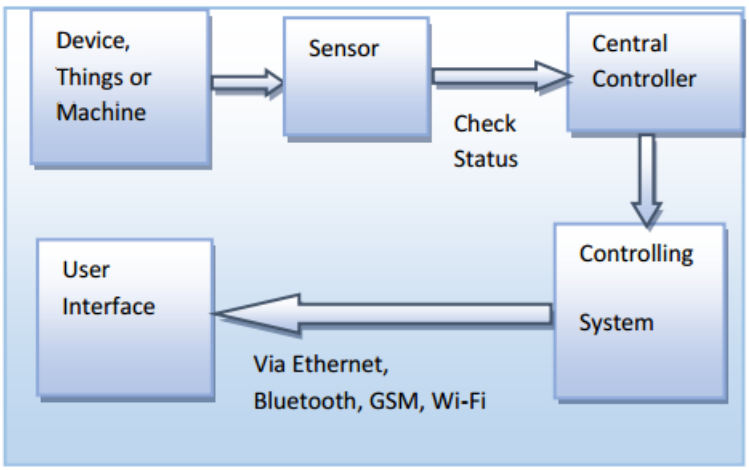
The heating module is been designed for the accommodation. The heating in the accommodation just can last 2 hours mostly. when you push the button once, twice, 3 times or 4 times, it can last fifteen minutes, thirty minutes, one hour and two hours respectively. In the winter night, people always be awakened by the cold. So the heating module is used to help the master to push the heating button at night.

These two modules are the first two ideas when this system becomes a product, there

are more modules will be designed for the consumers. And they are all modular design, so the different function modules can be published or bought separately.

2.2 Literature research

“Smart home is the place where everything is connected & controlled. Simply where electronics talk & live as neighbors.”[1] The objective of developing the intelligent household system is to make the home smart. It is smart because it can help us to solve the problems, create a comfortable environment for us and save energy, time and money.[1] It has a control unit to control the lighting, HVAC (warming, ventilation and aerating and cooling), machines, and different frameworks, to give enhanced accommodation, solace, better energy saving, productivity, and security. It can be remotely controlled by the web, mobile phone or any other electronics.[2] For different intelligent household systems, the basic architecture of Remote Home Automatic is figure.1. The device can use the sensor to check the status of the environment and send the data to the central controller. The controlling system will feedback the data to the user, the user can make the command by the parameters. Sometimes the controlling system will make the command automatically. In this program, WIFI technology will be used.[3]



(Fig.1 Basic block diagram of Home Automation)[4]

To compare WIFI and Bluetooth. Bluetooth can just transmit from one point to another point, it is not suitable to create a whole system. And the distance of Bluetooth transmission is just 10m. The distance of Bluetooth4.0 decreases to 50m. But the distance of WIFI transmission is 100m. It is more suitable for remote control. The transmission speed between them are both enough. The command in the system is very short, it does not need high speed and wide bandwidth. So comparing these specification. WIFI is been chosen.

The using of wireless technologies can provide four advantages that the wired network cannot achieve. Firstly, it can reduce the installation costs, the cabling is not necessary. Secondly, it is internet connectivity, the devices can be controlled from anywhere in the world with using mobile phones. Thirdly, it is expandable, when new or changed requirements, an extension of the network is necessary. Fourthly, it is easy to add devices to create an intelligent home security system and built-in security.[4] So the connection between devices is 2.4G wireless.

Now there are many smart home product categories. They can help masters to control everything from lights and temperature to locks and the security. For example the Amazon Echo (199.99GBP). It is a Bluetooth speaker powered by Alexa, Amazon's handy voice assistant. It works with some other smart home devices directly. So the users can use theirs voice to control the different smart home devices by Alexa, like lights, television, air condition, rice cooker.[5] The advantages of Amazon are good handy voice assistant and the cooperation of other smart home companies. But the cost of this system is to much. The Alexa is the central controller. The users give it the command, it analyses it and send the command to other smart home devices. All of the devices are must smart and can connect to Alexa.

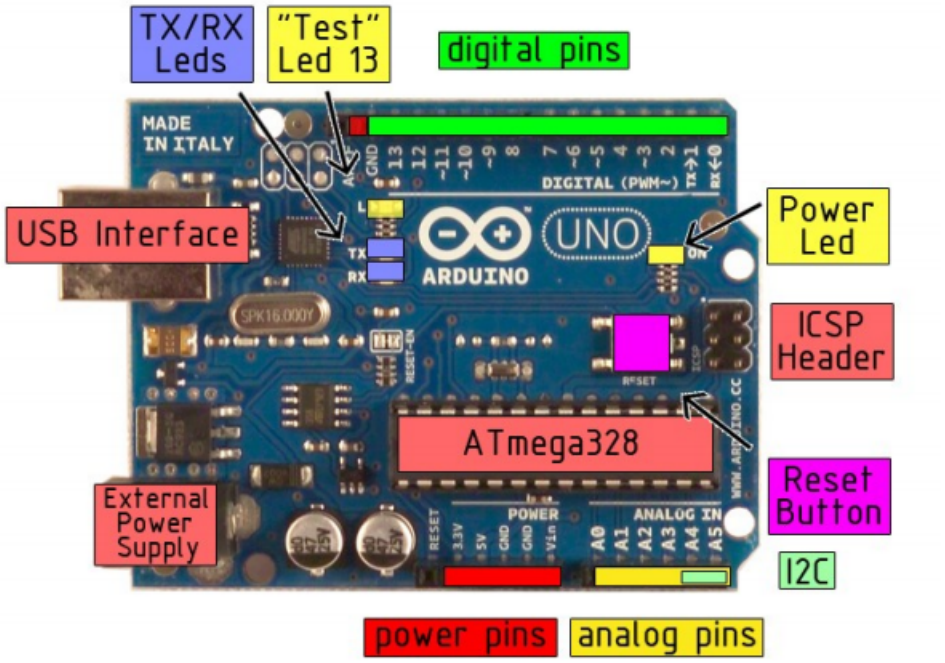
There are a lot of smart home devices. But a few whole smart home system are marketed. And they all need to be installed when the home is decorated. And it is very

expensive.

This project wants to develop a cheaper smart home system which is more suitable for some special situations. So the Arduino is been chosen to become the central controller of this system. It is cheap and has many matched modules.

The development of technology is very fast. After a few years some modules will be not up-data and not suitable for the environment. So the modular design and modular installation are necessary. The users can change the old module of the system by themselves.

Arduino is an electronic prototype platform with open source, it includes hardware (Arduino Board) and software (Arduino IDE). The Arduino board uses Atmel AVR microcontroller and some I/O ports to control the circuit. The program is been written in IDE in the computer, and then the program will be upload to the Arduino board. The development environment is similar to Java and language C. To compare with other microcontroller, Arduino is very cheap and modern. The using of Arduino is convenient and the accessories are abundant. So it is suitable for this project.



(Fig.2 Arduino board)[5]

In Fig.2, it is Arduino UNO is a microcontroller board which is based on the ATmega328. The board includes 14 digital pins, 6 analog pins and some power pins, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.The USB port is been used to upload program and support power for Arduino board. The technical specifications of Arduino UNO in this project are below:

Summary	
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommend	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootl
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

(Table.1 Technical specifications of Arduino UNO)[6]

3 The Specification & Design

3.1 Aim and objectives

The aim of this project is to design an intelligent household system which can be used to solve some problem for users and make users have fun.

The following are the objectives of the project:

1. Review the literature concerning artificial intelligent and smart system for household and Arduino applications. Etc.

2. Develop a basic vehicle, and write a program for it to move automatically.

3. Design some new functions(such as vision and autonomous routing on) for the basic vehicle and test these functions.

4. Develop the new modules for this system to make the home more intelligent and connect them to the basic vehicle.

5.Test the modules respectively, then test the whole system and modify it.

3.2 Requirements and Design specification

3.2.1 Function and performance

To implement the aim of this project, now this system has three modules. One is the basic rover module, it is a smart car. It can be controlled to move around and change the head of camera in the smart car by computer or Android through WIFI. It also can measure the temperature and light of the environment. The other one is lighting

module, it can be lighting when the smart car comes to a dark room which this module placed. The third one is the heating module. When the smart car feels cold, it will control the heating module to open the heating automatically.

3.2.2 Cost

This system have three modules. So three Arduino boards are needed and the price for each is 15GBP. The smart car is the most expensive, the chassis and most of the modules are form a robot store in China. The cost is 60GBP. The WIFI module (6 pieces) and sensors of temperature and light are cost 15GBP form Amazon. The lighting module is cheap, it just includes a breadboard and a LED. It can be got from the Electronic Lab. The framework of the heating module is form a discarded chassis, the other modules (timer, servo, motor driver, Arduino shield, battery) cost 15GBP. So the total cost of the project is 135GBP.

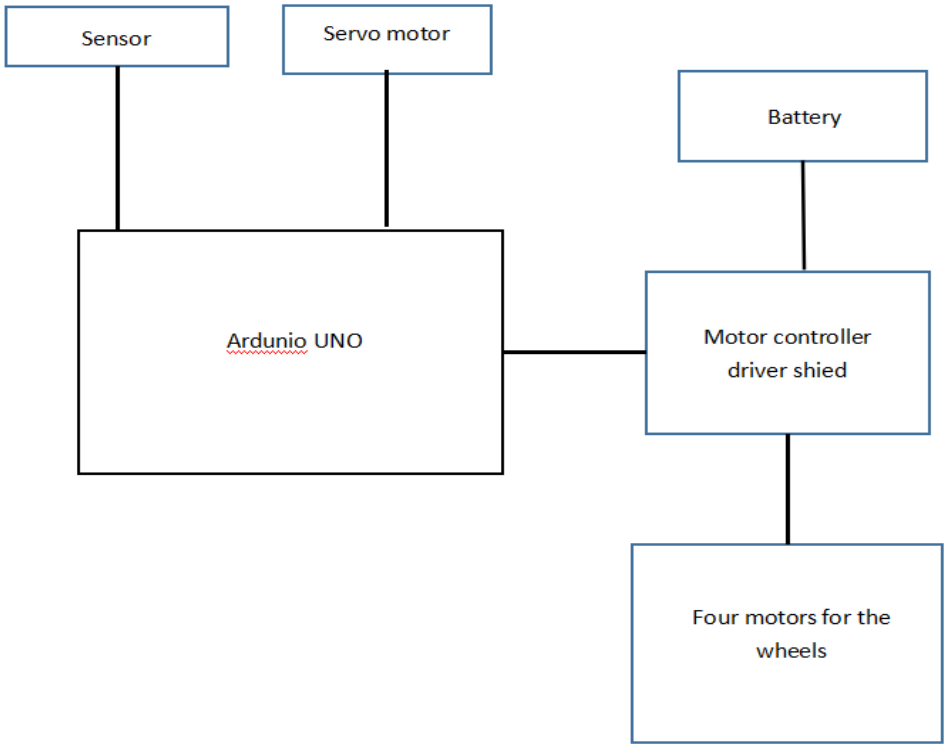
3.2.3 Users

This Intelligent Household system is suitable these families who live in apartment in city. The masters of the home are young man and wife because the smart car need be controlled by mobile phone, young people want to try new things and can adopt it quickly. If they have children or pets, it is better. Because the smart car can play with children and pets and monitor them to make sure they are safety.

3.3 The design of the project

3.3.1 The design of smart car

The Fig.3 is the original system of the smart car. It just includes an Arduino board, sensor, servo motor, motor controller and four motors. The program is written for it, so the smart car can move automatically and when it detects the obstacles in front of the car, it will change its direction.



（Fig.3 the system of the smart car)

The sensor is HC-SR04. It is an Ultrasonic Ranging Module and provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm[7]. And the picture of the real product is below:

(Fig.4 HC-SR04)

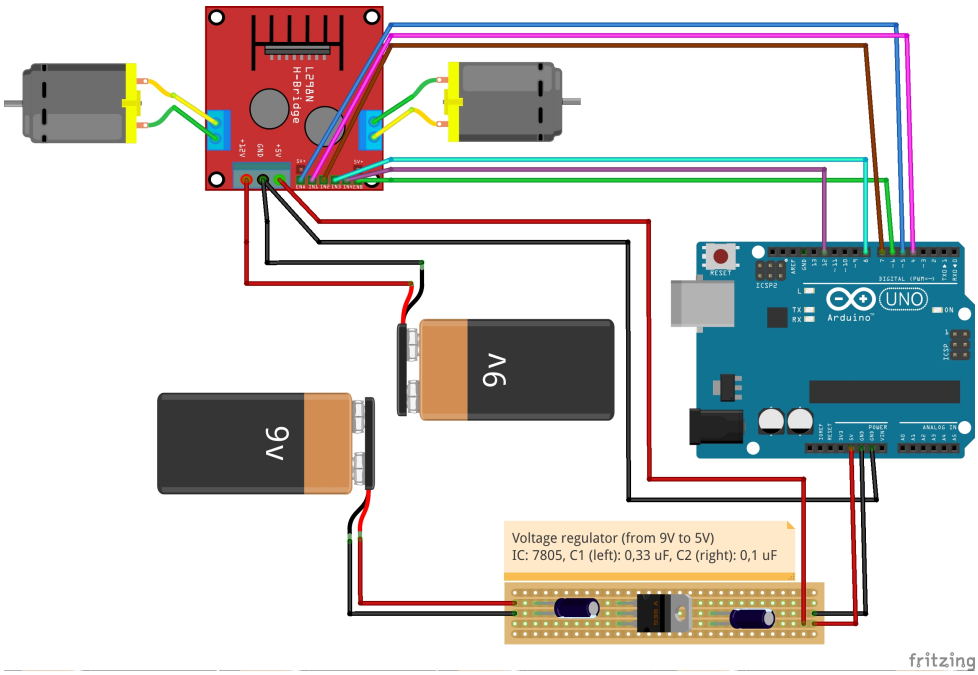
This sensor module is been attached in the front of the car and used to elude obstacle. When it detects an obstacle, it will feedback to Arduino, and Arduino will control the motors for wheels to change the direction to elude obstacle. The servo motor is been used to placed the sensor, so the sensor can detect more range.

There are four motors to drive the four wheels. The technical specifications of the motors are below:

Model	HC01-48
Standard	Double shaft
Reduction ratio	1:48
no-load (3V)	125 rpm
the speed installing a 66mm wheel (3V)	26 m/minutes
no-load (5V)	208 rpm
the speed installing a 66mm wheel (5V)	44 m/minutes
Speed	Medium
Torsion	0.8kg/cm

(Table.2 The technical specifications of motors)

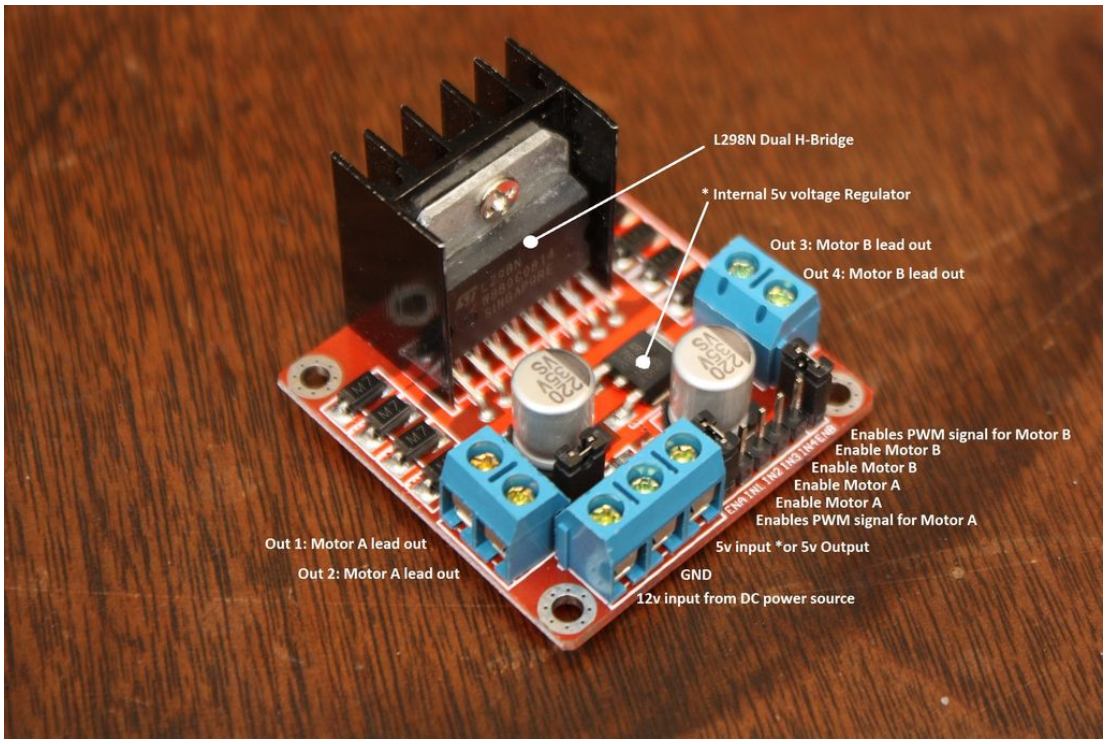
The remaining modules can be connected like Fig.5:



(Fig.5 Arduino controls motors)[8]

L298N is been used for the motor controller module. L298N Dual H-Bridge Motor Controller module is very popular and basic in the motor controller. H-Bridge's are typically used in controlling motors speed and direction. An H-Bridge is a circuit that can drive a current in either polarity and be controlled by Pulse Width Modulation (PWM)[9].

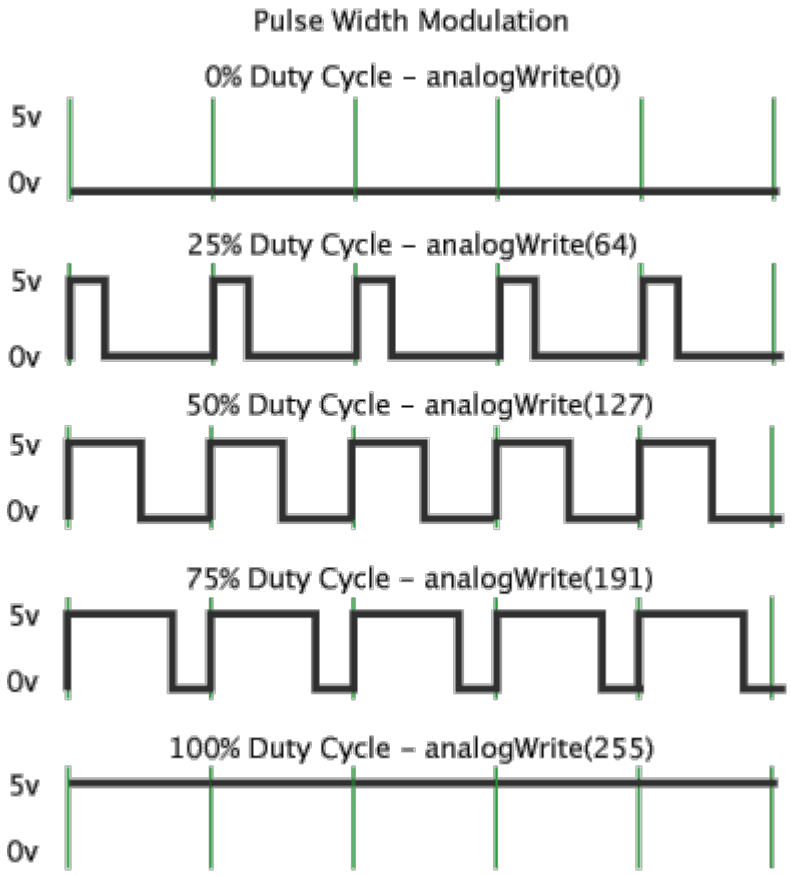
The picture of the real product is below:



(Fig.6 L298N Dual H-Bridge Motor Controller module)[10]

The functions of motor controller module: Firstly, It has H-Bridge, and H-Bridge can control the speed and direction of motors. Secondly, it can input a higher voltage than Arduino UNO for the motors. So it can drive more powerful motors.

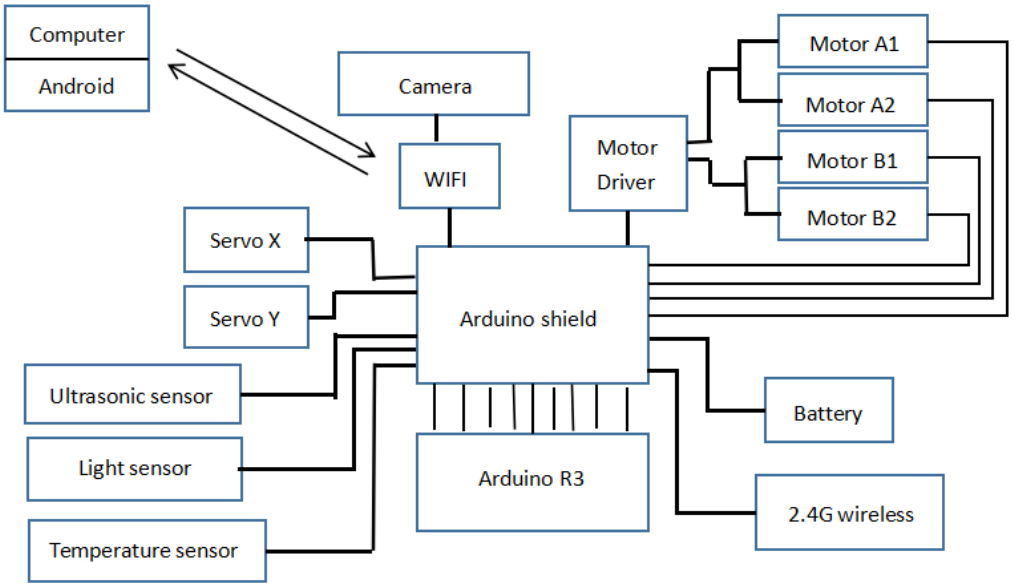
PWM is a technology way to simulate analog output with a digital pin by using the square wave which has different duty cycles. Digital pin can just output 0 and 1. 1 is full (5 Volts), 0 is off (0 Volts). If the voltage between 0 and 5 Volts wants to be got, we can change the width of the high pulse in every period. It is showed in Fig.7.



(Fig.7 The typical waves of PWM)[11]

The green line is a time period. It equals 1 divides the PWM frequency. And the analogWrite() is between 0 and 255. For example, when the wave is 50% duty cycle, $\frac{255+1}{2}-1=127$, so it equals analogWrite(127), and it is 2.5 Volts. This technique can be used to control the brightness of light, the speed of the motor and the angle of the servo.

By the development, the figure of the smart car system is below:



(Fig.8 Smart car system)

The line which connects different modules in this figure is not a wire, it includes data wires, power wire, and ground wire. The structure of the smart car like car chassis, different modules and the control application in computer or Android are bought from a robot store. AR-293D shield is been chosen, because it includes the 293D motor controller chip, Bluetooth port, WIFI port, two servos port, several infrared ports, ultrasonic port and 5V stabilized voltage chip, so it can power for Arduino board and L298N Motor Controller is not necessary. The Battery module for the smart car is two 3.7 Volts 18650 lithium Ion battery with 3000mAh each. So it can power 7.4 Volts for the Arduino shield. And the 5V stabilized voltage chip which is in Arduino shield will transmit the voltage from 7.4V to 5V to power the Arduino UNO and other modules. The two servos in the smart car are used to control the move of camera, they can make the camera turn right and left, turn up and down. The camera is not heavy, so the servos do not need strong power, but it need light weight to keep the car's balance. So Tower pro SG90 is been used for this smart car. This servo has high-strength transparent ABS shell and internal high precision nylon gear set. Those material make the servo just 9 gram, but the output torque of it reaches 1.6kg/cm. The sheet which is below shows more technical specifications of this servo.

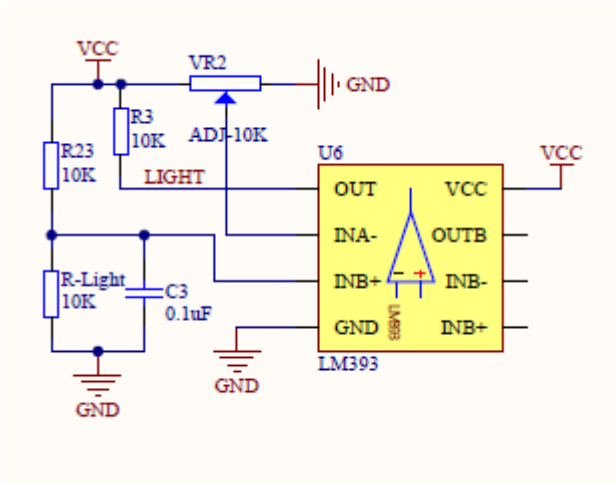
Technical specifications	
Working voltage	4.8V
Torque	1.8kg/cm (4.8V)
Speed	0.1s/60° (4.8V)
Working temperature	0~+55℃
Dead bandwidth	1μs
Dimensions	23*12.2*29mm
Weight	9g

(Table.3 Technical specifications of SG90)[12]

The camera supports for 720P video, it connects to WIFI module by a USB port and sends back the live images to computer or Android by WIFI. The WIFI module and computer, Android applications are bought with the car chassis, the WIFI and applications are not open sources, they are used as a method to connect the smart car and the master. It is not the main part that this project focuses on.

The lighting sensor and temperature sensor are working for the lighting module and the heating module respectively. LM393 is been chosen to become the lighting sensor. It is based on a voltage comparator, the photodiode connects in the two sides of R-Light, the photodiode will change it's the value of resistance when the light changes, then the change of resistance will be translated to the voltage signal which inputs INB+. This voltage signal will be compared with the reference voltage in INA-. When the voltage in INB+ > voltage in INA-, the output of voltage comparator will output high level. When the voltage in INB+ < voltage in INA-, the output of voltage comparator will output low level and the LED will light. If there is no light, the value of photodiode is very high. The voltage of point between photodiode and R23 will increase, the voltage in INB+ will be more than the voltage in INA-. The output of voltage comparator will output high level. If there is bright, the value of photodiode is very low. The voltage of point between photodiode and R23 will decrease, the voltage

in INB+ will be less than the voltage in INA-. The output of voltage comparator will output low level. The potentiometer VR2 is used to adjust the voltage of INA-, it's aim is to adjust the light sensitivity.



(Fig.9 Circuit of LM393)

The temperature sensor is AM2302. It can be used to measure temperature and humidity. In this project, the data of humidity is not necessary. And the technical specifications of temperature measurement is below:

Technical specifications (temperature)					
Parameter	condition	Minimum	Typical	Maximu	unit
Distinguishability			0.1		℃
			16		Bit
Repeatability			+ (-) 0.2		℃
Precision				<+(-)0.5	℃
Scale range		-40		+80	℃

Response time	1/e(63%)	6		20	S

(Table.4 Technical specifications of AM2302)[13]

The 2.4G wireless module is been used to connect the smart car to the lighting module and the heating module. The module in the smart car is sending a message, it receives the message in lighting or heating module. The model of wireless is nRF24L01. It can work between 2.4GHz and 2.5GHz in ISM spectrum and can send or receive the message. It includes frequency generator, mode controller, power amplifier, crystal modulator, modulator, and demodulator. Some features of nRF24L01 are showed:

Features of nRF24L01	
Low working voltage	1.9V-3.6V
High speed	2Mbps, because of the short transmission distance.
Many frequency points	125 points
Small volume	The 2.4GHz antenna is inside. Volume is 15*29mm.
Low power consumption	When working in answer model, fast transmission and start time reduce the
Low cost of apply	nRF24L01 integrates high-speed signal processing of RF protocol.

(Table.5 features of nRF24L01)[14]

The short transmission distance is also the advantage of nRF24L01 for this project. Because the temperature and light measured must be near the heating module and the lighting module, if the distance is too long, the data is useless. And it can control several rooms because the signal cannot across the wall.

Now, those different modules should be connected to Arduino UNO one by one. So the pins in Arduino UNO must be arranged firstly. The connections of the pins for all modules are below:

Smart car		
Module	Pin in Module	Pin in Arduino UNO
Motor 1	+	A0
	-	GND
Motor 2	+	A1
	-	GND
Motor 3	+	A2
	-	GND
Motor 4	+	A3
	-	GND
Servo X	Signal(yellow)	D10
	VCC(red)	5V
	GND(black)	GND
Servo Y	Signal(yellow)	D6
	VCC(red)	VCC
	GND(black)	GND
Ultrasonic sensor (HC-SR04)	VCC	5V
	ECHO	D9
	TEIG	D8
	GND	GND
WIFI	TTL: TX	TX
	RX	RX
	GND	GND
	VCC	5V
Light sensor (LM 393)	D	A5
	VCC	5V
	GND	GND

Temperature sensor (DHT AM2302)	DHTPIN	D2
	VCC	5V
	GND	GND
2.4G wireless (nRF24L01)	VCC	3.3V
	GND	GND
	CE	D4
	CSN	D7
	MOSI	D11
	MISO	D12
	SCK	D13
(A0-5 means analog pin number 0 to 5, D0-13 means digital pin number 0 to 13)		

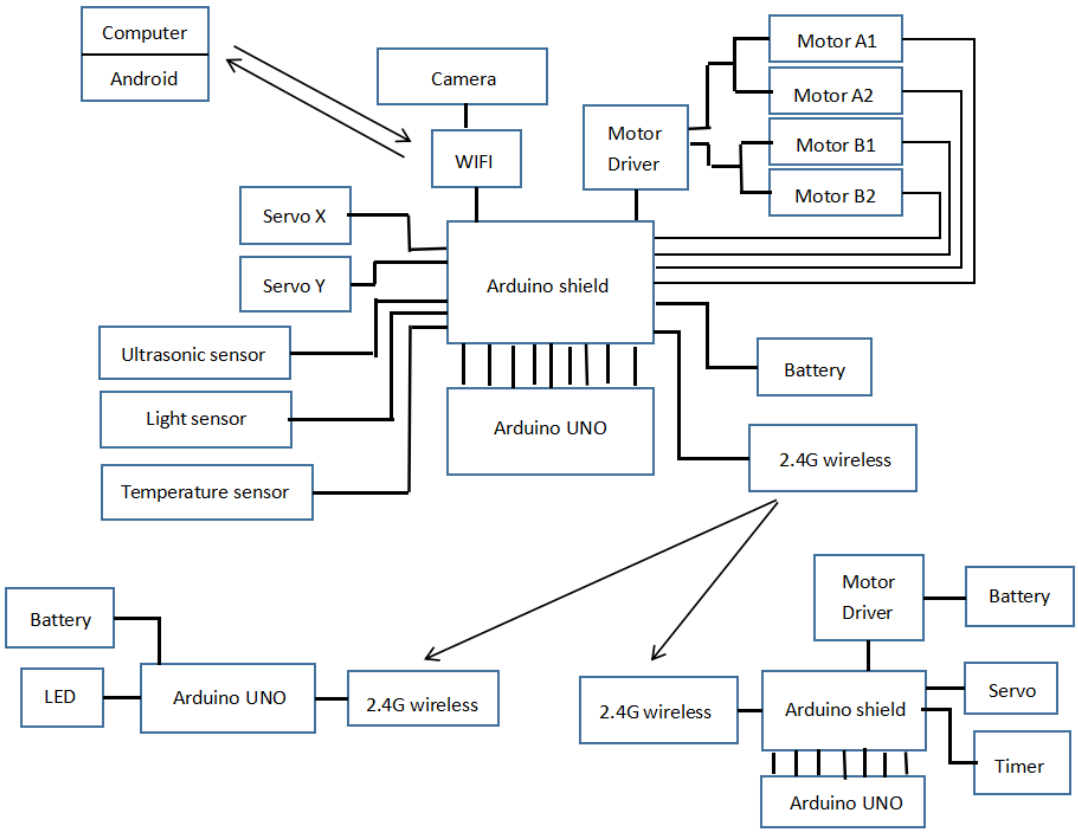
(Table.6 The connections of pins in smart car)

The choice for four motors changes from digital pins to analog pins. The first reason is that analog pins can output analog number to change the voltage to control the speed of motors and it does not need PWM. The second reason is that digital pins are not enough and analog pins have more space.

Now the smart car has those functions: it can be controlled to move front and back, turn right and left; the camera can be controlled to turn up and down, right and left, and it can transmit back the live images to master; it can detect the distance of obstacle to avoid it; it can measure the temperature, light and send message to the heating module and the lighting module by 2.4G wireless.

3.3.2 The design of the whole system and the system diagram

For the whole system, the Fig.10 is below:



(Fig.10 The whole system)

3.3.3 The design of the lighting module

The lighting module is simple, it just includes Arduino UNO, 2.4G wireless module (nRF24L01), LED and Battery. The 2.4G wireless module is the same model of smart car, but it is receiver and transmitter respectively in the lighting module and smart car.

The battery is 5V to power for Arduino UNO. And the LED is placed on the breadboard to simulate an electric lamp. The connections of the several parts in the lighting module is below:

The lighting module		
Module	Pin in Module	Pin in Arduino UNO
LED	+	D6
	-	GND
2.4G wireless (nRF24L01)	VCC	3.3V
	GND	GND
	CE	D4
	CSN	D7
	MOSI	D11
	MISO	D12
	SCK	D13
(D0-13 means digital pin number 0 to 13)		

(Table.7 The connections of pins in the lighting module)

3.3.4 The design of the heating module

For the heating module, in renting accommodation, the inside system of the heater cannot be changed. So to control the heater, a machine need to be made to push the heating button. It is a challenge for this project especially in mechanical design and the precision control. In the theory design, the servo is been used to push the button, Tower pro-MG995 and Tower pro-SG90 are candidates. The technical specifications of Tower pro-SG90 have been written before in this report, and the technical specifications of Tower pro MG995 are below:

Technical specifications	
Working voltage	3.0V-7.2V
Torque	13kg/cm

Speed (no load)	0.17s/60° (4.8V) 0.13s/60° (6.0V)
Working temperature	-30~+60 Degrees Celsius
Dead bandwidth	4μs
Dimensions	40.7*19.7*42.9mm
Weight	62g

(Table.8 Technical specifications of MG995)[15]

The test and compare of the two servos will be written in the next part of the report. The timer module is used to get the precise time which is suitable for the period of heating. After once pushing, the heater will last 15 minutes; twice, it will last 30 minutes; three times, one hour; four times, two hours. If pushing the button when the heater is working, the heater will turn off. So the timer controller must be precise. DS3231 is been chosen. The technical specifications of it are below:

Technical specifications	
Dimensions	38*22*14mm
Weight	8g
Working voltage	3.3V~5.5V
Time chip	DS3231
Clock precision (0~40℃)	2ppm; 1 minutes every year.
Memory chip	AT24C32 (32K)
Inside cell	CR2032 (3V)

(Table.9 Technical specifications of DS3231)[16]

The ds3231 module includes two calendar alarm clocks, it can generate the clock for the second, minute, hour, day, week, month and year. It uses IIC to transmit and the maximum speed is 400KHz (when the voltage is 5V). This motor driver module is just used to power for the Arduino board. It is L298N. The battery is a box with four 1.5V AA batteries in series. So it is 6V. The connections of the heating module are

below:

The heating module		
Module	Pin in Module	Pin in Arduino UNO
Servo	Signal(yellow)	D3
	VCC(red)	5V
	GND(black)	GND
Timer (DS3231)	SCL	A5
	SDA	A4
	VCC	5V
	GND	GND
2.4G wireless (nRF24L01)	VCC	3.3V
	GND	GND
	CE	D4
	CSN	D7
	MOSI	D11
	MISO	D12
	SCK	D13
(A0-5 means analog pin number 0 to 5, D0-13 means digital pin number 0 to 13)		

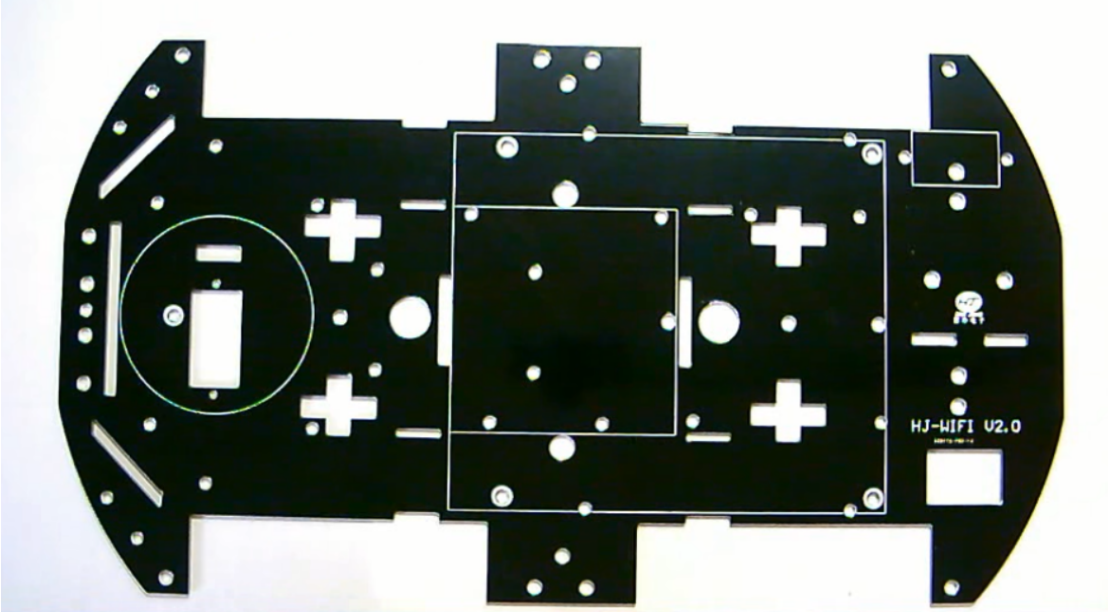
(Table.10 The connections of pins in the heating module)

4 The implementation

4.1 Smart car

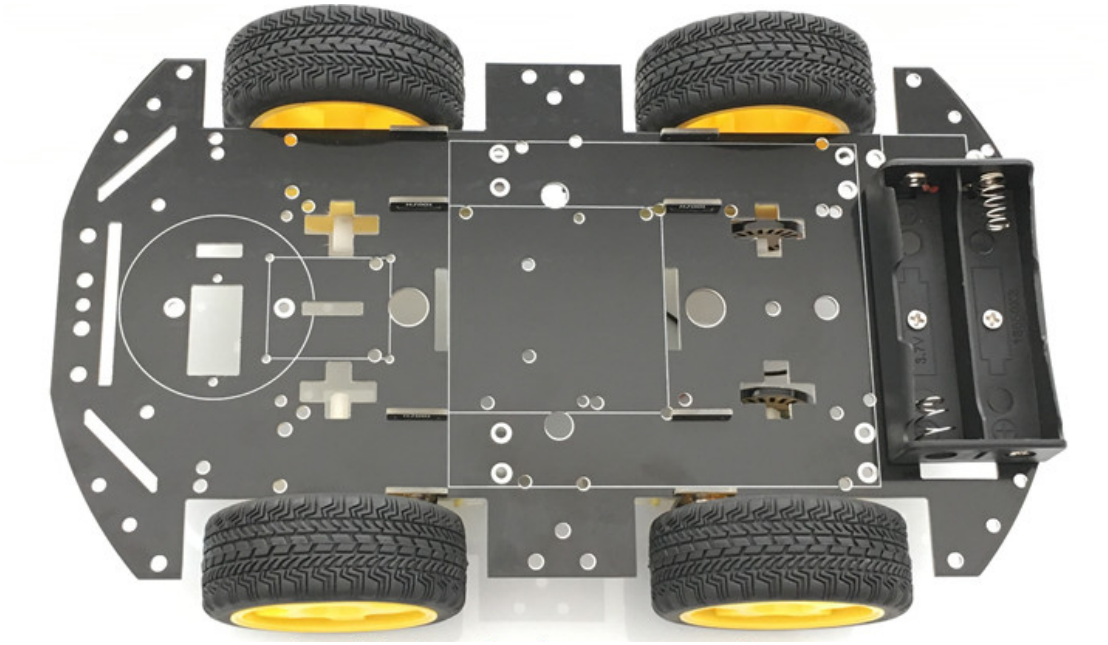
4.1.1 The mechanical implementation of smart car

The basic frame of the smart car is been given by the robot store. So this project will focus on the layout of the different modules in the car and the optimization of the smart car. The chassis of the car is in figure.11. The length is 260mm, the width is 140mm, the thickness is 2mm. There are many holes reserved to place the sensors and Arduino board.



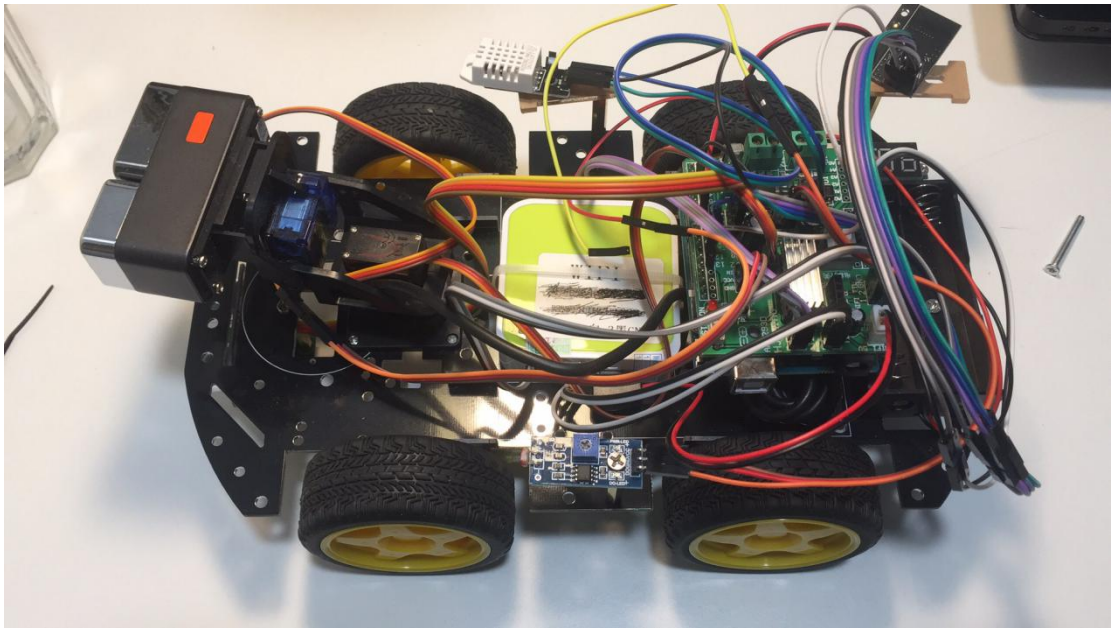
(Fig.11 The chassis of the smart car)

The four motors are placed on the downside of the chassis, each of them is fixed to the chassis by two “T” holder (35mm*15mm*3mm) and two M3 screws. And the wires of the four motors go through the holes to the upside of the chassis. The wires of two motors in the right side are one group, and the left side is another group. Then the four wheels will be locked in the motors respectively. And the battery box is placed in the stern of the chassis, now it looks like a vehicle in figure.12.



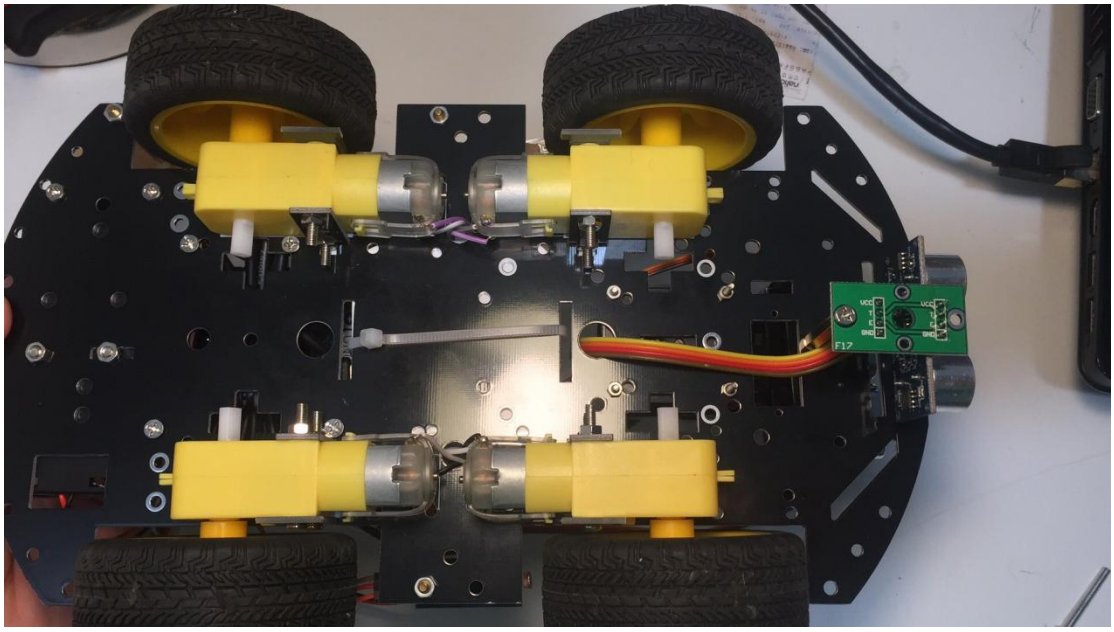
(Fig.12 The rover)

After placing all the modules to the chassis, the wires of different modules will be connected to the Arduino shield with the table.6. Figure.13 shows the real smart car with all the modules in the design figure.7. There is a voltmeter near the battery box to monitor the supply voltage. The Arduino board is in the back of the chassis, it is been risen up by four metal screws and covered with the Arduino shield. Rising up the Arduino board has three advantages, firstly there is more space to place the wires; secondly, it is easy to connect the wires to Arduino shield; thirdly it is good at heat dissipation. The WIFI module is in the middle of chassis. In the front of the chassis, there is a double-shaft head. It is made up of two SG90 servos and a frame. The ultrasonic module is also in the front of the chassis because it needs to detect the barrier in front of the car. But it is on the downside of chassis, because of the lack of space. Finally, the lighting sensor, temperature sensor, and wireless module are placed in the two sides of the chassis. They are also been risen up by iron sticks. It can reduce the interference of the measurement of the sensors by smart car and also can reduce the interference of the wheels' moving by the sensors.



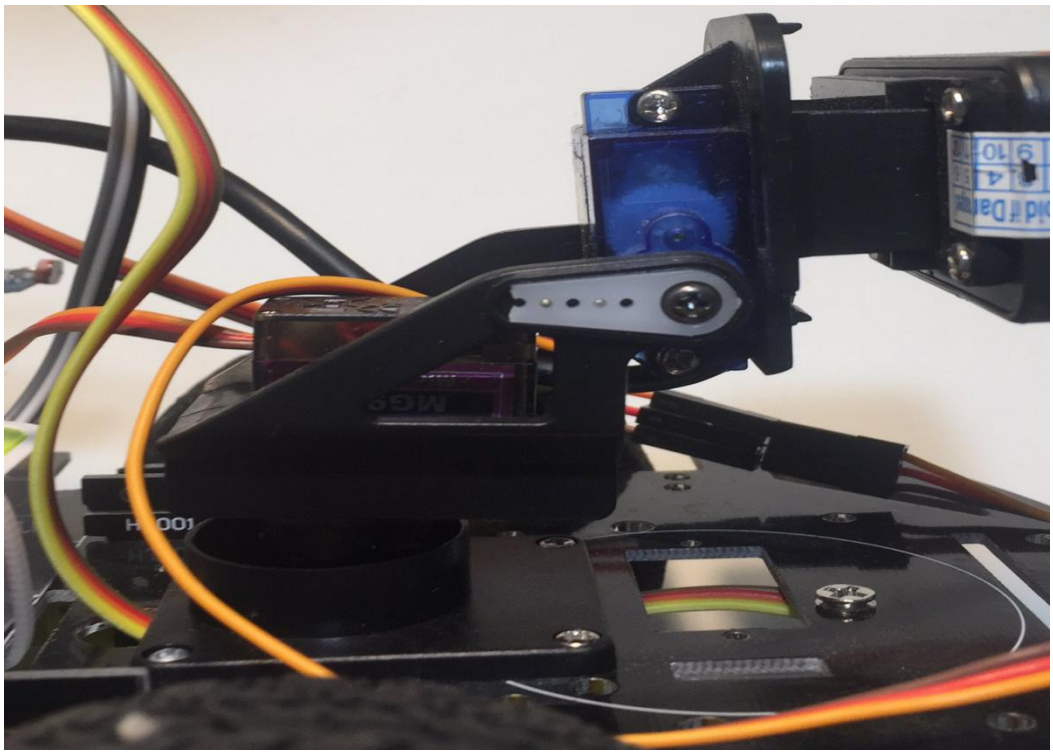
(Fig.13 The real smart car)

The downside of the smart car is been shown in figure.14. There are just ultrasonic sensor and motors placed. It is very clear to increase the ground clearance and reduce the destruction for the modules when the smart car catches barriers or wades. The volume of the smart car is 260mm*145mm*120mm.



(Fig.14 The downside of the real smart car)

For the double-shaft head module, there are two servos. The top one is used to control the head turn up and down, the range is 0~180°. The bottom one is used to control the head turn right and left, the range is also 0~180°. When installing it, there is a problem for the double-shaft head module. The frame is not suitable for the rocker arms of SG90 servos. The rocker arms are too large to put into the frame. So the file is been used to polish the rocker arms. And the rocker arm is fragile, it should be treated carefully. The double-shaft head module is in figure.15, the pointy end of the white rocker arm is been polished.

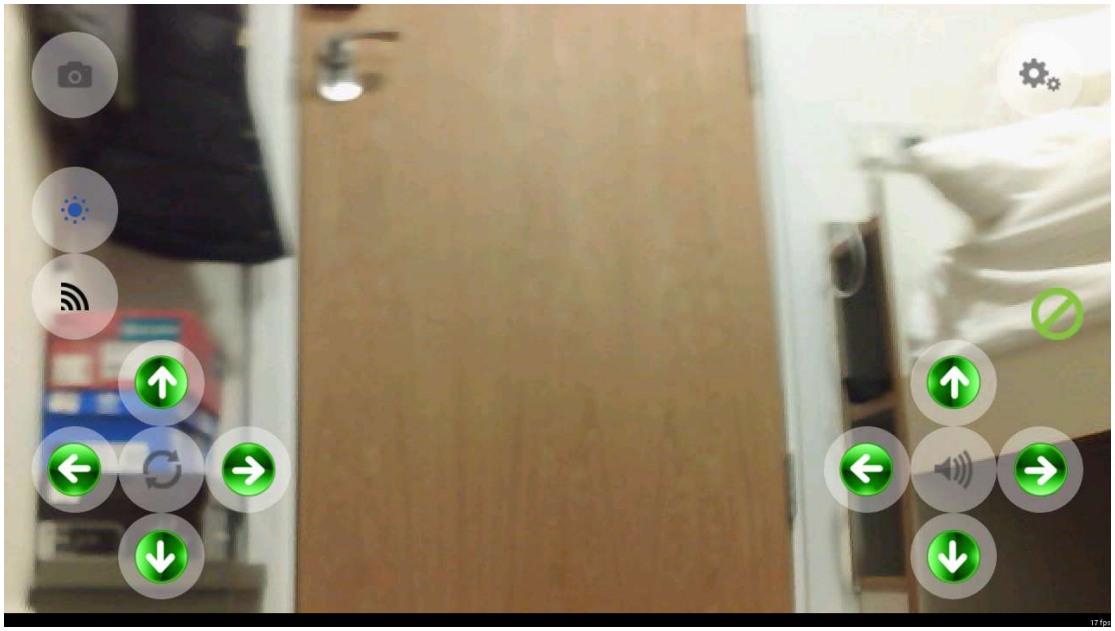


(Fig.15 The double-shaft head)

4.1.2 The interface of the WIFI control application

This application for WIFI control is from the robot store and it is not open source. The interface of the control system in Android is shown in figure.16. The background figure in the interface is the current image of the camera in the smart car. It will

transmit the image to the controller in real time by WIFI. The five buttons in the lower left of the interface are used to control the head turn up and down, turn right and left, and reset the original statue. The three buttons which are above the five buttons are image shoot, turn on, off the light and turn on, off the radar. There are another five buttons in the lower right of the interface, they are used to control the smart car move front and back, turn right and left, and the horn. The button in the upper right corner is used to set some parameters. But the light and horn have been removed from the smart car, so these two buttons are useless.



(Fig.16 the interface of control system in Android)[17]

4.1.3 The program of the smart car

4.1.3.1 The introduction of Arduino IDE

The program which controls the whole smart car system is been written in the Arduino IDE (Integrated Development Environment). It includes code editor, code compiler, code debugger and Graphical User Interface (GUI). It can be used to write, analysis, compile and upload the code. The interface of Arduino IDE is been shown in figure.17. The top layer of the interface is the menu bar. The second layer of the interface is the toolbar. The five buttons on the left of the tool bar are compiling, uploading, new program, opening program and saving program. The right one in the right of the toolbar is the serial monitor. The large space with white background in the middle of the interface is been used to write and modify the code. The oblong space with the black background in the bottom of the interface is the status bar. The result of the compiling and upload will be displayed on it.

(Fig.17 The interface of Arduino IDE)

4.1.3.2 The initialization and setup() of smart car program

The whole smart car program is attached in Appendix 9.1. In the smart car program. Firstly, the library files must be added at the beginning of the program. There are *Servo.h, DistanceSRF04.h, SPI.h, Mirf.h, nRF24L01.h, MirfHardwareSpiDriver.h* and

DHT.h. *Servo.h* is used for the servo, it uses PWM to control the moving angle of servo by the output of the digital signal. *DistanceSRF04.h* is used for the ultrasonic sensor, it can calculate the time between the ultrasonic is sent from the sensor and the ultrasonic is received by the sensor after reflecting by a barrier. Then it can transmit the time to the distance between the sensor and barrier. *SPI.h, Mirf.h, nRF24L01.h and MirfHardwareSpiDriver.h* are all used for the 2.4G wireless module. These libraries are used to initialize the wireless module and set the different parameters of the wireless module, like checking the channel of the transmission and setting the waiting time for the receive. *DHT.h* is used for the temperature sensor. It sets the measuring period and transmits the digital signal to the degree centigrade.

Then the global variables are been defined follow the libraries. Those variables include the pin numbers of some ports and the initial values and limit values of motors and servos. The left program can be divided into three parts: *setup(), loop()* and some functions. The context in the *setup()* will be run once when the Arduino board is powered up or reset. The context in the *loop()* will be run circularly after the *setup()* finished. The several functions will be called in the *loop()*.

In *setup()*, the input and output of pins and some modules' pins are been defined. The code for the initial set of the 2.4G wireless module is below:

Serial.begin(9600); //initialize the serial communication and set the Baud rate as 9600. The serial monitor can be used to monitor the serial print. It is helpful to debug the program.

Mirf.cePin = 4 ; //set the CE pin in wireless module is connected to digital pin 4 in Arduino UNO.

Mirf.csnPin = 7; //set the CSN pin in wireless module is connected to digital pin 7 in Arduino UNO.

Mirf.spi = &MirfHardwareSpi;

Mirf.init(); //initialize nRF24L01

*Mirf.setRADDR((byte *)"Sen01");* //set the identifier (Send01) of the sender.

Mirf.payload = sizeof(unsigned int); //set the number of bytes in once transmission. In this project, an integer will be transmitted. sizeof(unsigned int) is equal 2 bytes.

Mirf.channel = 3; //set the number of the channel (3). the range is 0 to 128. And the number of channels are must the same between sender and receiver.

Mirf.config(); //configure the device.

Serial.println("I'm Sender..."); //print "I'm Sender" in the serial monitor.

4.1.3.3 The functions of smart car program

There are nineteen functions in this program. All the names of these functions is been shown in table.11.

The functions' names of smart car's program	
void getSerialLine()	void processCommand(String input)
String getValue(String data, char separator, int index)	void servo_test(void)
void servo_right(void)	void servo_left(void)
void servo_down(void)	void servo_up(void)

void servo_center(void)	void servo_Vertical(int corner)
void servo_Horizontal(int corner)	void qian(void)
void hou(void)	void you(void)
void zuo(void)	void ting(void)
void SetEN()	void SendStatus()
void SendMessage(String data)	

(Table.11 The functions' names of smart car's program)

The void which is front of the name of function means there is no return value in this function. *String getValue* means the type of the return value is a string. The brackets which are behind the name of the function is used to declare the formal parameters. The formal parameters are used to receive and call the parameters when the function is transmitted. If the void is in the brackets, it means there is no actual parameter transiting to this function.

For the function *void getSerialLine(), it is used to get the*

```
void getSerialLine()
{
    while (serialIn != '\r')            //while statement includes the first
```

three if statements, these if statements work's condition is that serialIn is not equal '\r'. And '\r' is been used become the space character between two commands.

```
    {
        if (!(Serial.available() > 0))          //if there is no serial message
```

available, this function will return. If the serial message is available, the following code will work.

```
{
    return;
}
serialIn = Serial.read();                    //the serial message will be put into variate serialIn.
if (serialIn != '\r') {
    if (serialIn != '\n'){                   //the two if statements used to make sure the message which is in serialIn is not ‘\r’ and ‘\n’.
        char a = char(serialIn);             //the cerialIn is transmitted to character type and put into a.
        strReceived += a;                    //put the single character into the character string.
    }
}
}
if (serialIn == '\r') {                      //the while statement finishes, this if statement is used to reset the commandAvailable and serialIn when the serial message is space character.
    commandAvailable = true;
    serialIn = 0;
```

```
    }
}
```

The fucntion *void processCommand(String input)* is been used to identify the command and execute the corresponding operation. Another function getValue is been called to arrange the received parameters in one message in this function processCommand. Then a lot of if, else if statements are been used to identify the command. For example:

```
if (command == "MD_Qian")
    {
        qian();
    }
    else if (command == "MD_Hou")
    {
        hou();
    }
```

If the command is “MD_Qian”, the function qian() will be called. If the command is “MD_Hou”, the function hou() will be called. After these if statements for command. There is another if statement in the bottom of this function which judge whether the command has been received. In this if statement, the functions SendMessage and SendStatus are been called to feedback the statuses of the motors, servos, speeds, radar and lights by serial communication. If there is no command has been received, the statuses will not been sent to controller. It can reduce the waste of bandwidth.

In function *String getValue(String data, char separator, int index).* It is a function with return value and three formal parameters. The code for this function is below:

String getValue(String data, char separator, int index) //the formal parameters are data (string type), separator (character type), index (integer type).

{

int found = 0; //define a local variable found =0, and it is integer.

int strIndex[] = {0, -1 }; //define a local integer one dimensional array strIndex[], strIndex[0]=0, strIndex[1]=-1.

int maxIndex = data.length() - 1; //define a local integer maxIndex, it is the length of the formal parameter data minutes one. Because the parameter i in the next for statement start at 0, and the length of data is start at 1.

for (int i = 0; i <= maxIndex && found <= index; i++){ //for statement, if i <= maxIndex and found<=index, the statement will continue to work, and i will add one when for statement loops once.

if (data.charAt(i) == separator || i == maxIndex){ //the if statement is been included in the for statement. If the number i character of data is a separator or i is equal to maxIndex, the following three statements will be executed.

found++; //the local variable adds one.

strIndex[0] = strIndex[1] + 1;

strIndex[1] = (i == maxIndex) ? i + 1 : i; //if i == maxIndex, strIndex[1]=i+1, else strIndex[1]=i.

```
        }
    }
    return found>index ? data.substring(strIndex[0], strIndex[1]) : "";
```
//this statement is used to return the value. If found>index, data.substring(strIndex[0], strIndex[1]) will be returned, else “” will be returned.

```
}
```

The aim of this function is to divide the parameters in one message. When the message is been transmitted for the controller, it may includes one, two or three parameters. And one parameter is corresponding to one command. For example, to control the smart car to turn right and change the speed of left motors, change the speed of right motors, there are three commands and they can be transmitted in one massage.

void servo_test(void), void servo_right(void), void servo_left(void), void servo_down(void), void servo_up(void), void servo_center(void), void servo_Vertical(int corner) and *void servo_Horizontal(int corner)* are all used to control the servos in head. Servo_test is called in the setup() to test the servos. Servo_right, servo_left, servo_down and servo_up are used to control the head turn right and left, up and down. Servo_center controls the head move to the original status (looking forward). Servo_Vertical and servo_Horizontal are used to transmit the step of moving to the destination angle and output the operation to servo by PWM. Servo_Vertical is used to control servoY (the head turns up and down), and servo_Horizontal is used to control servoX (the head turns right and left). The code for Servo_Vertical will be explained as a example.

```
void servo_Vertical(int corner)
```
//the integer formal parameter corner is representative of the destination angle after this moving.

```
{
    int cornerY = servoY.read();                    //the present angle of servoY
```
will be read, and an integer variable cornerY is equal to this angle.

```
    if (cornerY > corner) {                         //if the present angle is bigger
```
than the destination's, the angle will be decreased. The following for statement will be executed.

```
        for (int i = cornerY; i > corner; i = i - servoStep) {          //the variable i
```
starts at cornerY, if i > corner, the for statement will work again, and every time the i=servoStep. The servoStep is a constant which is defined in the front of the program. It is four. And this for statement always work twice, because the corner = cornerY - servoStep.

```
            servoY.write(i);                        //output the angle i to
```
servoY by PWM.

```
            servoYPoint = i;                        //the present angle of
```
servoY will be put into servoYPoint.

```
            delay(50);                              //delay 50ms to wait
```
the servo finish the work.

```
        }
    }
    else {                                          //if the present angle is
```
not bigger than the destination's, the angle will be increased. And the following for statement is almost the same of above one.

```
        for (int i = cornerY; i < corner; i = i + servoStep) {
```

```
            servoY.write(i);
            servoYPoint = i;
            delay(50);
        }
    }
    servoY.write(corner);                    //output the destination angle
to ensure the servoY is in the current status.
    servoYPoint = corner;                    //the present angle of servoY
will be put into servoYPoint.
}
```

void qian(void), void hou(void), void you(void), void zuo(void), void ting(void) and *void SetEN()* are used to control the moving of smart car. There are four status functions :Moving front, moving back, turning right, turning left and stopping. SetEN is used to called in the five functions to output the speed and status to motors.

void SendStatus() and *void SendMessage(String data) are used to feedback the statues to controller by serial communication.*

4.1.3.4 The loop() of smart car program

The loop() is the main part of the program. After the initialization of the parameters, the loop part will be executed circularly by the Arduino microcontroller. The statements in loop() can be divided into four parts. The first part is used to check the status of radar, and judge whether use it. The code is below:

```
if (radar == true)                                    //the if statement will be
```
executed if radar is true. And the radar will be turned on and turned off in function processCommand.

```
    {
        distance = Dist.getDistanceCentimeter();      //measure the distance
```
between the sensor and barrier and put it into variable distance.

```
        if (distance <= 5 & distance > 1){            //judge whether the
```
distance between the sensor and barrier is 1~5, if so, the following statements will be executed to avoid the accident.

```
            hou();                                    //call the function hou(),
```
the smart car will move back.

```
            delay(100);                               //delay 100ms to wait the
```
operation of moving back finish.

```
            ting();                                   //call the function ting()
```
to stop the smart car.

```
            distance = 0;                             //reset the distance. make
```
the variable distance equal to 0.

```
        }
    }
```

The second part of the loop() is calling functions to get the message for the controller and do the command. The code is just several statements because the details are in the functions. The code is below:

getSerialLine(); //call the function getSerialLine to get the message for controller, and put it into variable strReceived. If the message has been got, the commandAvailable will be changed to true.

if (commandAvailable) { //the commandAvailable is true, it means the message has been put into strReceived, then the following statements can be executed.

processCommand(strReceived); //call function processCommand to execute the command in the message, and the function getValue is called in this function to divide the parameters in the message to commands.

strReceived = ""; //reset the strReceived.

commandAvailable = false; //reset the commandAvailable.

}

The third part of the loop() is used to get the current temperature and the current light by the temperature sensor and lighting sensor in the smart car, then these parameters will be translated to a message. The result of lighting sensor has two statuses 0 and 1. The temperature is been divided into three scopes: less than minimum temperature (mintem=20); between minimum temperature and maximum temperature (maxtem=30); more than maximum temperature. The following table.12 shows the temperature and light statues are corresponding to the message.

The message and its parameters for temperature and light.		
light	temperature	message (adata)
1	<=mintem	1
1	>mintem and <maxtem	3
1	>=maxtem	5
0	<=mintem	2

0	>mintem and <maxtem	4
0	>=maxtem	6

(Table.12 The message and its parameters for temperature and light)

The fourth part of the loop() is used to send the message to receivers (the lighting module and the heating module) by the 2.4G wireless module. This message is been created in the third part of loop(). The code is below:

//the 2.4G wireless module nRF24L01 can just send the data by the single byte, and the size of data is Mirf.payload. So the data which need to be transmitted must be split to the single byte. Mirf.payload is been defined in the beginning of the program, it is 2.

byte data[Mirf.payload]; //define a one-dimensional array data[], the type is the byte. It is data[2].

//variable adata is been defined in the setup(), the type of it is unsigned int, so it has two bytes. Before transmitting it, it must be split. It can be split to high 8 bits and low 8 bits.

data[0] = adata & 0xFF; //0xFF is 11111111, so the low 8 bits of adata can be saved in data[0].

data[1] = adata >> 8; //adata is right shifted 8 bits. The low 8 bits of it have been discarded, and the high 8 bits are saved in data[1].

*Mirf.setTADDR((byte *)"Rec01");* //set to transmit the data to "Rec01". The 2.4G wireless modules in the lighting module and the heating module are both "Rec01", so they can receive the data simultaneously. Then the two modules will extract the parameter which they need.

Mirf.send(data); //send the data byte by

byte.

while(Mirf.isSending()) { //while loop, it will print "wait" in serial communication circularly until the data has been transmit.

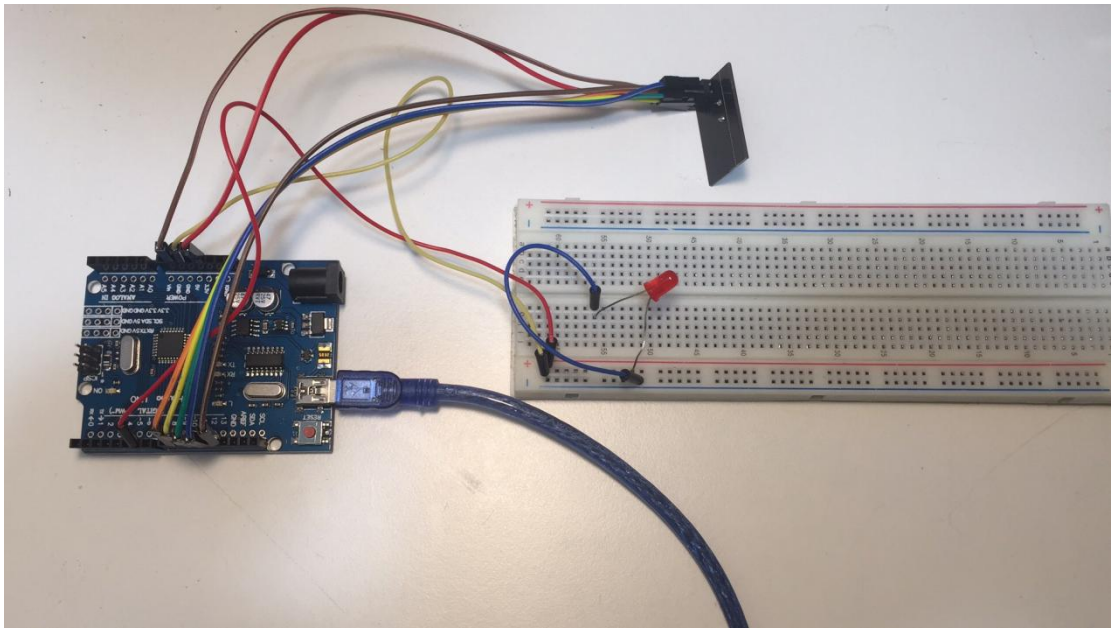
Serial.println("wait");

}

4.2 The lighting module

4.2.1 the mechanical implementation of the lighting module

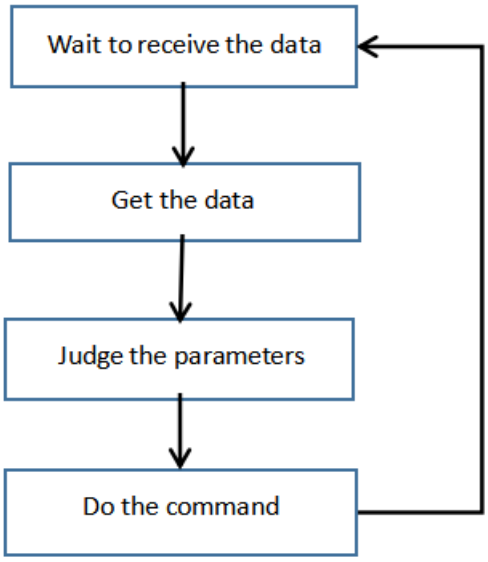
In the actual product, the lighting module is connected to the lamp or light of the room in the house. In this project, to have a convenient presentation, the light emitting diode (LED) is been used to simulate the light in the room. Then the breadboard is necessary to place the LED. The mechanical implementation of the lighting module is very simple, placing the LED on the breadboard and connecting the wires in table.7 are all the work. The most important part of the module is the wireless module. It needs to hold and wait to receive the signal of the wireless module in the smart car. The USB interface will be connected to the computer to power the Arduino board. Then the Arduino board can power LED by a digital pin. There is a notice that the long leg of LED is +, the short leg is connected to GND. The whole lighting module is been shown in figure.18.



(Fig.18 The lighting module)

4.2.2 The program of the lighting module

The whole lighting module program is attached in Appendix 9.2. The program is almost focusing on the 2.4G wireless module. When the data has been received, the parameters in the data will be judged to make sure which command can be executed. In setup(), the definition of the wireless is almost the same of smart car's. The difference is that *Mirf.setRADDR((byte *)"Rec01");* the identifier of the receiver is "Rec01" to distinguish from the sender.



(Fig.19 the processing sequence of loop() in the lighting process)

Figure.19 shows the processing sequence of the loop(). The whole code of loop is below:

```
void loop()

{

    byte   data[Mirf.payload];                    //define a byte
```

one-dimensional array data[]. The size is Mirf.payload. It has been defined in setup() (Mirf.payload = sizeof(unsigned int);). So Mire.oayload is 2, the array data[] is data[2].

```
    if(Mirf.dataReady())                          //use if statement to
```

wait the wireless module is ready to receive the data. If Mirf.dataReady() is 1, then the following statements will be executed.

```
    {
```

Mirf.getData(data); *//get the message and use array data to save the message. The message has been split to array data[1] and data[0] when transmission, so the receiver also use array data[1] and data[0] to collect the message.*

adata = (unsigned int)((data[1] << 8) | data[0]); //define a unsigned integer adata, it is 2 bytes. Then data[1] is left shifted 8 bits and or data[0] bit by bit. It will make up a 16 bits integer which is saved in adata.

Serial.println(adata); //print the value adata in serial monitor to check the program.

if(adata ==1 || adata == 3 || adata == 5) //use if statement to judge the parameters in the data. When adata is equal 1 or 3 or 5, it means the parameter of light is 1. It means the environment near the smart car is dark, so the lighting module needs to turn on the light.

{

usefuladata = 1; //use variable usefuladata to record the status of the light. usefuladata =1 means the light is turning on.

digitalWrite(lightPin, HIGH); //output high level to lightPin (digital pin 6), turn on the light.

}

else if (adata ==2 || adata == 4 || adata == 6) //When adata is equal 1 or 3 or 5, it means the parameter of light is 0. It means the environment near the smart car is bright, so the lighting module needs to turn off the light.

```
    {
        usefuladata = 0;                    //use variable
```
usefuladata to record the status of the light. usefuladata =0 means the light is turning off.

```
        digitalWrite(lightPin, LOW);        //output low
```
level to lightPin (digital pin 6), turn off the light.

```
    }
        Serial.println(adata);              //print the value
```
adata in serial monitor to check the program.

```
}
delay(200);                                 //delay 200ms to
```
wait the operation has done. It is also used to have an interval when checking the data in serial monitor.

```
}
```

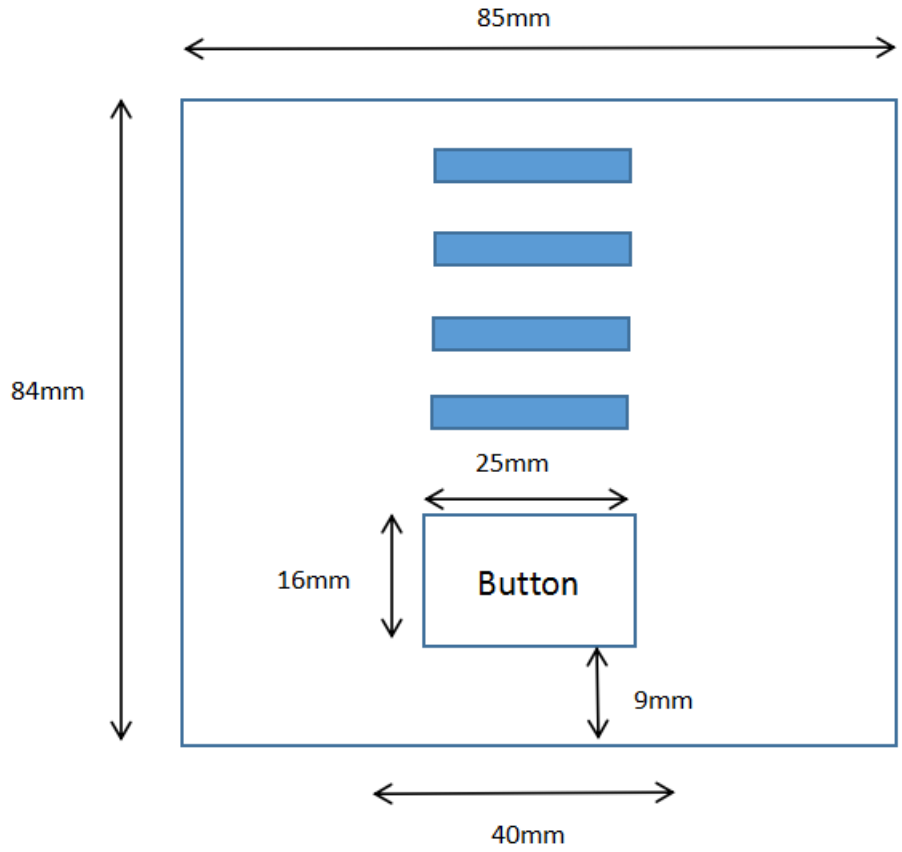
4.3 The heating module

4.3.1 The mechanical implementation of the heating module

The switch for the heater in the accommodation is shown in figure.20. The front side of the switch is almost a square. The height is 84mm and the width is 85mm. The middle of the front side is a protruding half column. And there are four indicator lights and an oblong button in the column. The thickness of the switch is 15mm, but the thickness in the middle is 22mm.

(Fig.20 The real heater switch)

The detailed physical parameters of the switch are shown in figure.21. 40mm is the width of the column in the middle of the switch.

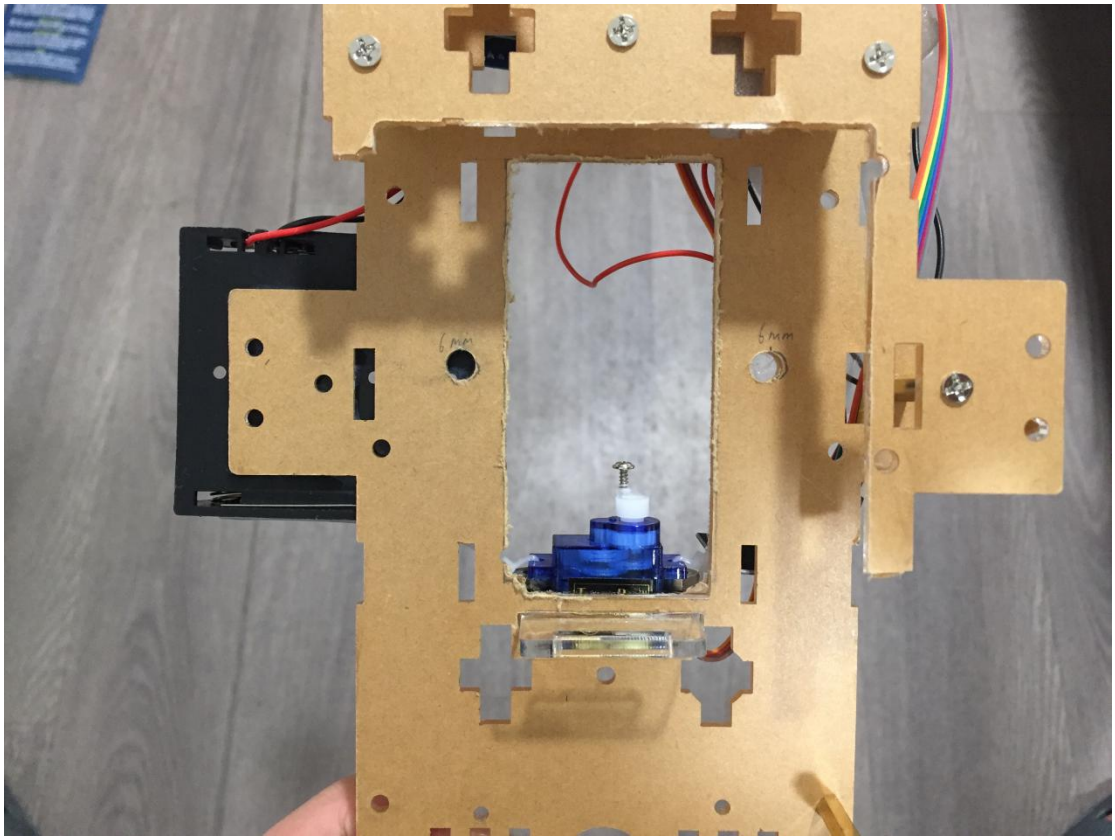


(Fig.21 The size of the switch)

The frame of the machine is two abandoned car chassis, the volume of the chassis is 270mm*150mm*3mm. These two boards are been connected by metal sticks and been fixed by screws. The length of the sticks is 25mm. To suit for the switch, these two boards have been cut. The back one of the two boards is used to trap the four broadsides of the switch. So this board should be cut off a rectangle with 85mm*84mm. The length and width of the rectangle will be reduced 1mm each in the first time cutting. Then comparing the size of the board and switch and using the file to polish the gap in the board are working together. It can make sure the board can be fixed in the switch. The front one of the two boards is been cut off a rectangle with 84mm*40mm to make sure the button and indicator lights are not been blocked by the board. The precision of this cutting is not necessary. But the notice is that the two rectangles which be cut in the two boards must be corresponding. In the below of the rectangle in front board, there is a 4mm*35mm rectangle being cut which is used to

place the servo.

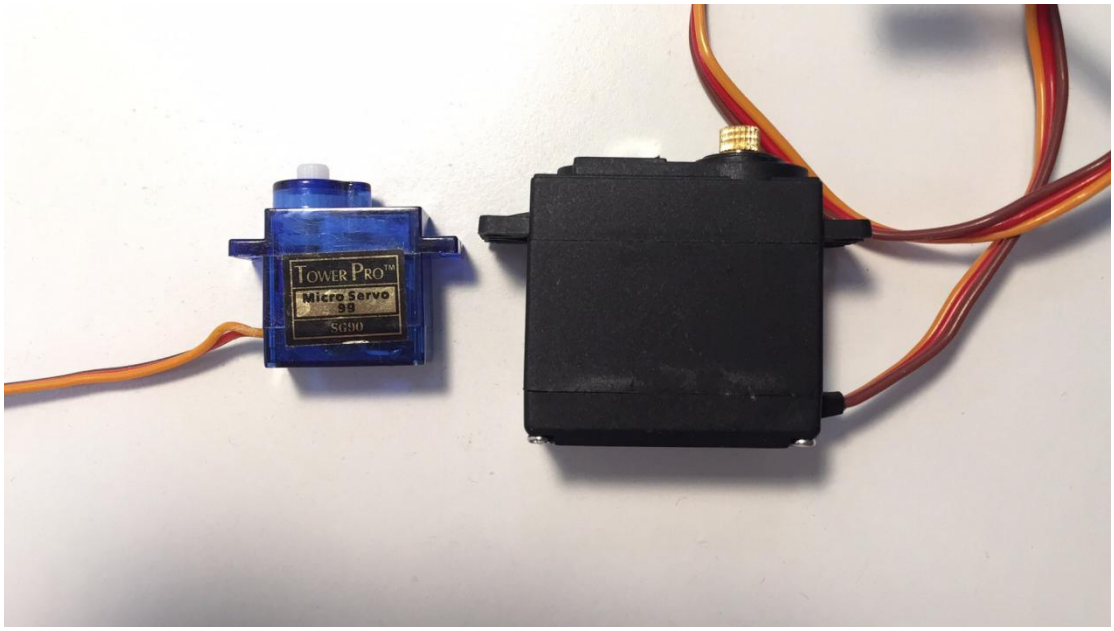
When polishing the oblong gap of the back board, the board has been broken down because the material of board is too fragile and the two sides of the board are too slim after cut of the rectangle. The upper half of the board can also lock the switch. So the bottom half has been discarded. After cutting, these two boards can be seen clearly in figure.22.



(Fig.22 The back of the lighting module)

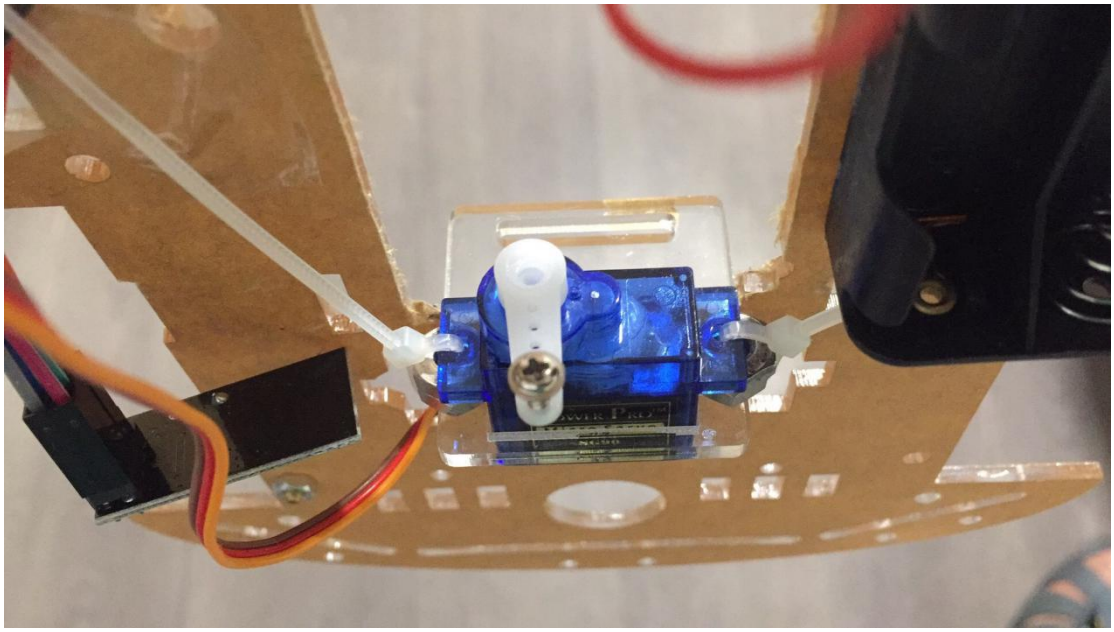
The most important part is the installation of the servo. It needs to push the button correctly. SG90 servo and MG995 servo are been compared to become the servo for the heating module. The real products of the two servos are in figure.23. And the technical specifications are showed in table.3 and table.8. MG995 servo has high torque, but it also has the large volume and heavy weight. SG90 and MG995 servo are both used to push the button, the torque of them are both enough. And the small board

which used to connect the front board and servo just can install the size of SG90. So SG90 is been chosen to become the servo for the heating module.

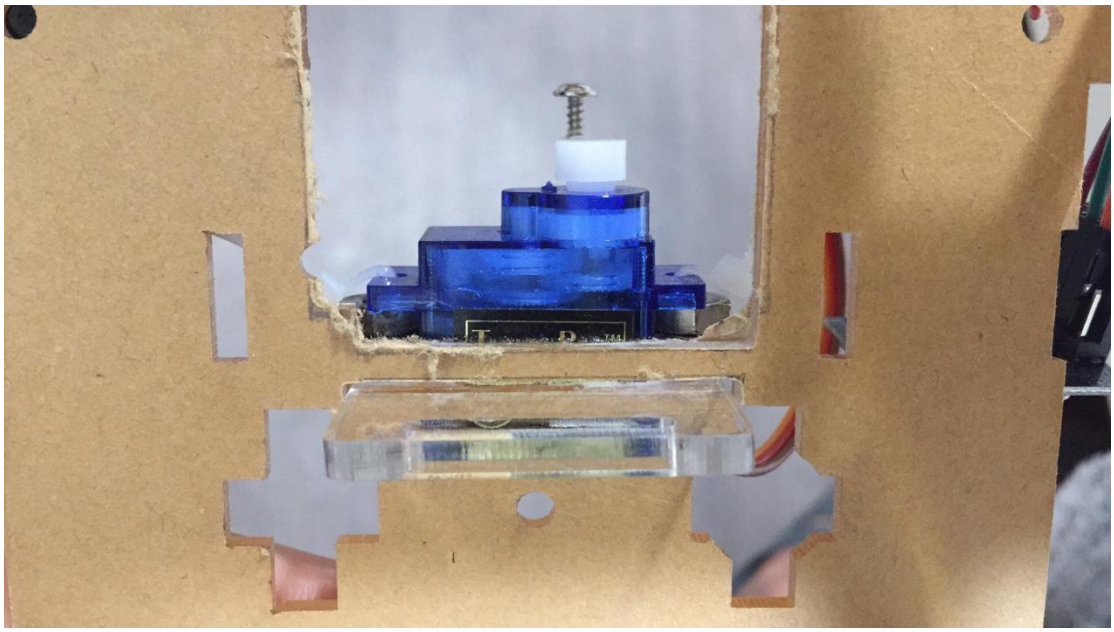


(Fig.23 SG90 servo and MG995 servo)

The figure.24 and figure.25 show the front and back of the servo. The servo is been locked in a small board by two rolling belts. And there are four nuts between the servo and the small board (two nuts each side), it is used to rise up the servo. Because the rocker arm is a litter low to touch the button of the switch. Then the small board with the servo is been pushed into the oblong gap in the front board of the heating module. It is clear in figure.25. After, the super glue is been used to fix the small board and the front board. The screw in the rocker arm was used to touch the button because of the short of the rocker arm. But it is not stable. Sometimes the screw touches the button, but the button has not been pushed and the heater does not work. Now the cuts are been used to rise up the servo, so the rocker arm can touch the button by itself, and it is more stable. The screw in the rocker arm is just a decoration now.



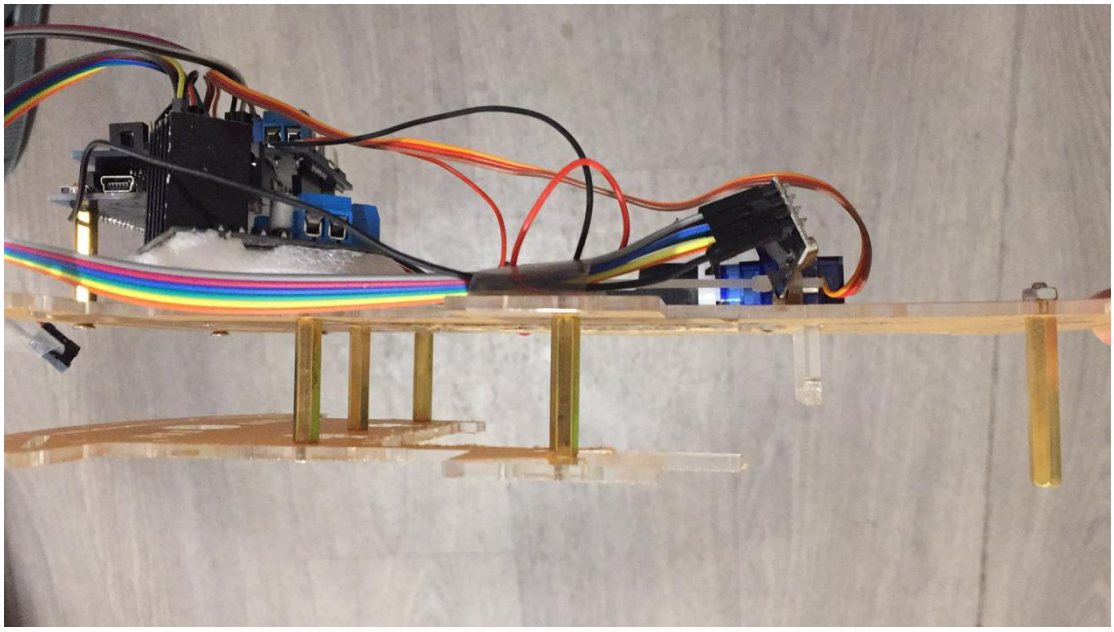
(Fig.24 The front of the servo in heating module)



(Fig.25 the back of the servo in heating module)

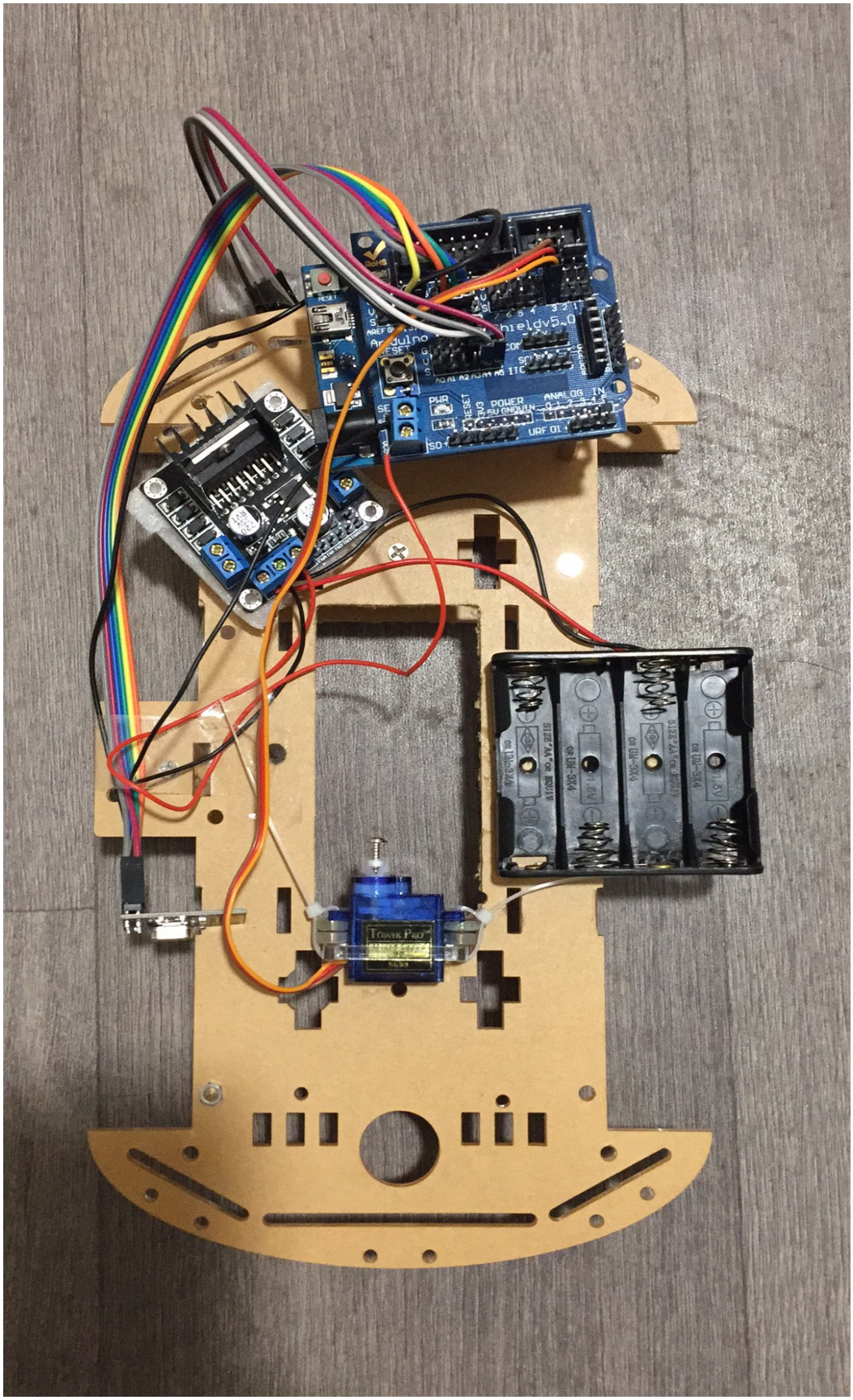
Because the bottom half of the back board has been discarded. The bottom of the front board is not stable. After the servo pushes the button, it is shaky. So another metal stick has been used to fix in the bottom of the front board by strewing, and the other side of the metal stick holds the wall. This metal stick is longer than which used to

connect the two boards because the board has thickness. The length of it is 30mm. In figure.26, there are four metal sticks (25mm) on the left of the figure, there are used to connect these two boards. The metal stick (30mm) which is on the right of the figure is used to hold the wall to keep the machine stable.



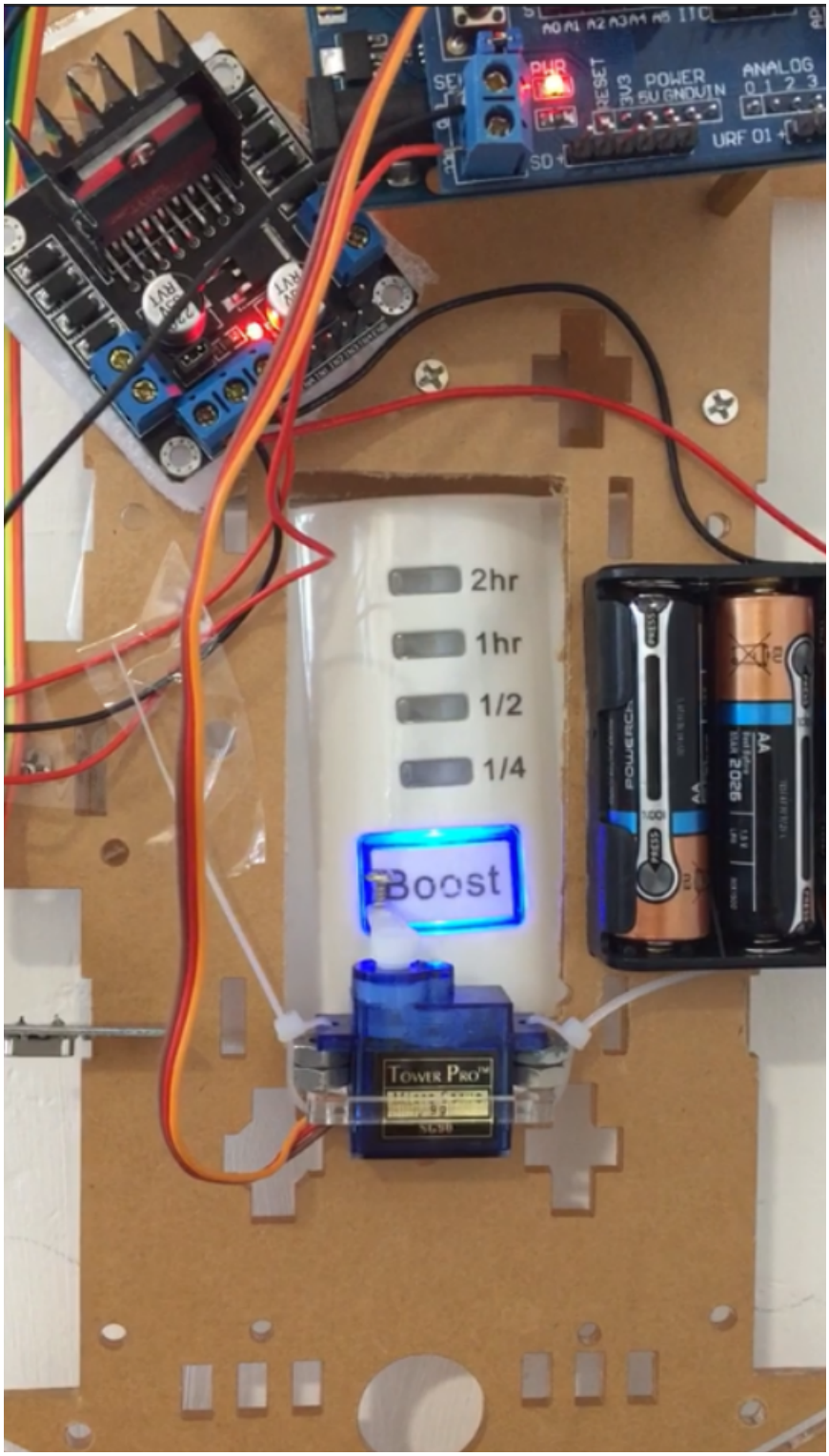
(Fig.26 the metal sticks in heating module)

The finished product is been shown in figure.27. The fixation of different modules (battery, Arduino UNO, Arduino shield, motor control, timer, wireless) can be done at the end of work. The Arduino UNO is also been risen up by three metal sticks. Because there are no screw holes suitable for the motor controller and the back of the motor controller is not smooth, the glue cannot fix it with the board. So a piece of foam plastics is been used to connect the motor controller and board. Then the super glue is been used to fix them. The wireless module is been fixed to the left of the front board by sellotape. The timer module is been fixed to the back of the front board by sellotape. The battery box is been fixed in the right of the front board by glue.



(Fig.27 The heating module)

The heating module is been put in the switch. Figure.28 shows that. When the wireless module receives the signal that the temperature is less than the limit, the servo will spin 180° and return to use rocker arm push the button.



(Fig.28 The work of heating module)

4.3.2 The program of the heating module

4.3.2.1 The initialization and setup() of the heating module program

The whole heating module program is attached in Appendix 9.3. 2.4G wireless module, servo, and timer are included in the program. In setup(), the initialization of wireless module is the same of lighting module's. The initialization of timer RTC_DS3231 is below:

Serial.begin(9600); //initialize the serial communication and set the Baud rate 9600. Then it can be monitored by serial monitor.

delay(3000); //delay 3000 ms to wait the console open.

if (! rtc.begin()) { //if rtc.begin() =0 (means the timer is not ready), the following two statements will be executed. If the timer is ready, the if statement will be skipped.

Serial.println("Couldn't find RTC"); //when rtc.begin() =0, the serial print "Couldn't find RTC", it will be seen in serial monitor.

while (1); //it is a infinite loop to remind controller there are some problems in the timer.

}

if (rtc.lostPower()) { //if the timer lost power, it need to reset the time. The following statements will be executed.

Serial.println("RTC lost power, lets set the time!");

rtc.adjust(DateTime(F(__DATE__), F(__TIME__))); // set the RTC to the date & time this sketch was compiled. This statement sets the RTC with an explicit date & time, for example to set: April 20, 2017 at 3am. you would call: rtc.adjust(DateTime(2017, 4, 20, 3, 0, 0));

}

4.3.2.2The loop() of the heating module program

The processing sequence in the loop() is similar to the lighting module's. Firstly the wireless module is waiting for the data. When it receives the data, the parameter in data will be judged. Then the corresponding command will be executed. The command is complex, so it is written as a function: work() and work2().The part of receiving the data is the same of the lighting module's. The part of the code which is used to judge the parameter in data is below:

if (adata == 1 || adata == 2) //if adata is 1 or 2 (the current temperature is lower than the minimum temperature 20℃), the function work2() will be executed.

{

work2();

}

else if (adata == 3 || adata == 4) //if adata is 3 or 4 (the current temperature is higher than the minimum temperature 20℃ and lower than the maximum temperature 30℃), the function work() will be executed.

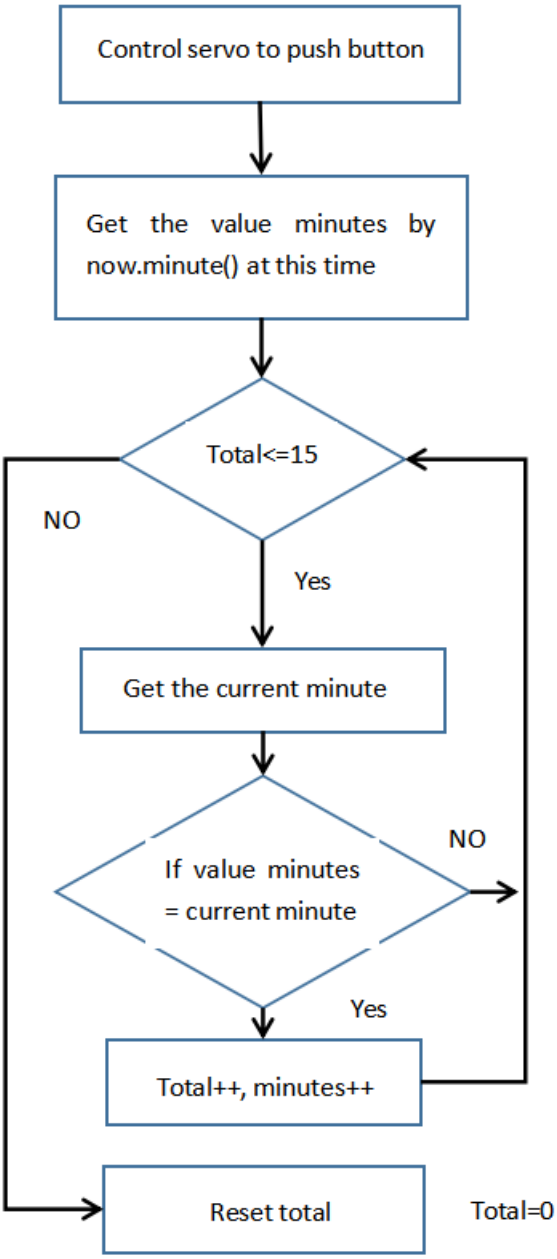
```
    {
      work();
    }
    else if (adata == 5 || adata ==6)                    //if adata is 5 or 6 (the
```
current temperature is higher than the maximum temperature 30℃), there is no command will be executed.

```
    {


    }
  }
```

4.3.2.3 The functions of the heating module program

The difference between work() and work2() is that work() pushes the button once, and the heater will work 15 minutes, work2() pushes the button four times at one time, the heater will work 2 hours. So the work2() is used for lower temperature and work() is suitable for medium temperature. The flow chart of work() processing is shown in figure.29.



(Fig.29 The flow chart of work() processing)

It just extracts the current minute in timer and uses the change of minute to count the number. When counting 15 times, it means the heater turns off. And this function will finish its job and return to main program loop(). The code for the function work() is below:

void work()

```
{
    for(pos = 0; pos < 180; pos += 1)                    //this for statement
```
is used to move the current angle from 0° to 180°, and every time it will add 1°.

```
  {
    myservo.write(pos);                    //output the current value of angle to
```
servo.

```
    delay(15);                             //delay 15ms to wait the servo move to
```
the objective position.

```
  }
  for(pos = 180; pos>=1; pos-=1)           //this for statement is used
```
to move the current angle from 180° back to 0°, and every time it will minus 1°.

```
  {
    myservo.write(pos);                    //output the current value of
```
angle to servo.

```
    delay(15);                             //delay 15ms to wait the servo move to
```
the objective position.

```
  }
        DateTime now = rtc.now();          //get the current time
```
from timer module and save it in the DataTime now. DataTime now includes the current year, month, week, day, hour, minute and second.

```
Serial.print("adata=");
```

```
Serial.println(adata);
```
//print the adata in serial monitor to check the program.

```
minutes = now.minute();
```
//get the current minute from DataTime now and save it in variable minutes.

```
Serial.print("minutes=");
```

```
Serial.println(minutes);
```
//print the variable minutes in serial monitor to check the program.

```
while(total <= 15)
```
//the while loop is used to count the minute's change, it is the minutes that the heater has worked. And it will be count by the variable total. When total > 15, it means the heater is turning off and the function will return to main program loop().

```
{
```

```
Serial.println(total);
```
//print the variable total in serial monitor to check the program.

```
DateTime now = rtc.now();
```
//get the current time from timer module and save it in the DataTime now.

```
if ( minutes != now.minute())
```
//compare the value minutes with the current minute, if there are not equal, it means that one minutes has past, so the following statement can be executed.

```
{
```

```
total = total +1;
```
//total is used to

count the minutes, so after one minute total can add 1.

```
Serial.println(total);                    //print the variable
```

total in serial monitor to check the program.

```
if( minutes == 59)                        //to compare with the
```

current minute, the value minutes need to add 1, but when minute is 59, the next is 0. So if statement needs to distinguish these two conditions. If the value minutes is 59, it will become to 0.

```
{
  minutes = 0;
}
else
{
  minutes = minutes + 1;                  //if the value minutes is
```

not 59, it will add 1.

```
}
delay(300);
}
}
if (total == 16 )                         //the first change of
```

minute is been counted as one minute, but it is must less than one minute. So the total will count from 1 to 16. It can avoid the situation that the next function work() will be

called when the heater is working.

```
        total = 0;                                    //reset the value
```

total for the next work.

```
}
```

The function work2() just uses a for the statement to control the servo move four times and change the total to 60.

5 Test, Results and Discussion

After the three modules have been done, the programs will be uploaded to the Arduino boards respectively. The test for the smart car is been done firstly. The initialized parameters like the status of head, the speed of left and right motors need to be adjusted. Now, the smart car can be controlled by Android mobile phone by WIFI. The control distance between the user and the smart car depends on the effective scope of the WIFI module in the smart car. It can work at about 50 meters between the user and smart car if no walls among the rest. It also can go through two walls, but the control distance will decrease. The head can move as expected and move back to the forward when the user push the button. The speed of right motors and left motors have been adjusted. When pushing moving forward button, it can walk a straight line. It also can turn right and left quickly.

Then, the lighting module and the heating module will be tested. The program for lighting module has been changed to output high level and low level to the LED light circularly to test the light. The program for the heating module has been changed to push the button frequently to adjust the status of the servo. After that, the communication between the three modules has been tested, the three modules are all connected to the computer by USB. The serial monitor will monitor the communication between the three modules. After that, the transmission distance has been tested. It can work at about 10 meters. It can just go through one wall. It is suitable for the environment. Because the sensors in the smart car must measure the data of environment near the lighting module or the heating module. Finally, the whole system has been installed in the accommodation to have a practical test. After several days test. There are several problems being found.

Firstly, after a long time using, the voltage of the battery for the smart car is not stable. Although the Arduino shield has constant voltage system, the servos for the head are also slight shaking. The angle of the servo is been controlled by voltage

using PWM, the constant voltage system cannot clear all the fluctuation of voltage. So the angle of the servo is not stable. It is not a big problem for control and camera.

Secondly, when having a large number of tests for pushing the button, it is failed sometimes. The probability is less than 5% if the device and servo are been adjusted to the suitable status. Because the heating module is made by hand, the precision is limited. It is difficult to increase the success rate of pushing button by using mechanical methods. The software method can be used. For example, reducing the work time for each pushing. In work() function, the heater will just work 15 minutes. So if the pushing operation is failed, it just wastes 15 minutes, then the next work() function will be called to push the button. The second solution is that after few minutes of the pushing operation, the temperature sensor will measure the temperature data. If the temperature has not changed, it means the pushing operation is failed. The interrupt function will be used to restart the operation.

Thirdly, the appearance of the three modules in this system is terrible. If it will become a product for consumers, the appearance needs to be redesigned. It must look friendly and pretty.

6 The future work

Although the intelligent household system has been built successful, if it wants to become a product, there are more future work need to do.

Firstly, the extended modules are just lighting module and heating module, there are more modules need to be developed to make the home more intelligent.

Secondly, the heating module is been designed suitable for the switch in the accommodation. In future, to suit more other sizes of switch, the heating module need to be designed as a device which can change its size to suitable another switch. And the different heating modes can be selected by using a mobile phone.

Thirdly, the appearances of the three modules need to be beautified.

Fourthly, the control application is from HuiJing Electronic company, it is not open sources. So for the future expansion, the Android control application should be developed.

7 Conclusions

This project develops an intelligent household system based on Arduino to help the masters. It is successful. All the objectives can be done on time.

Now the system includes three modules: smart car, lighting module, heating module. The smart car can be controlled to move around and turn around the head of the camera by Android a mobile phone, and it can send back the image to mobile phone by WIFI. The lighting module and the heating module are connected to the smart car by 2.4G wireless. The temperature sensor and light sensor are in the smart car, they measure the data of the environment and transmit to the lighting module and the heating module, these two modules will do the different operation by the parameters in the data. The communication and work between the system is automatic, so it is convenient for the users. And the cost of this system is not high. Many materials are discarded things, like the boards of heating module.

8 References

[1] "What is smart home - Basic Idea".*cctvinstitute.co.uk*. Retrieved 2016-10-29.

[2] Deepali Javale, Mohd. Mohsin, Shreerang Nandanwar and Mayur Shingate. International. Journal of Electronics Communication and Computer Technology (IJECCT) Volume 3. Issue 2 (March 2013).

[3] Ahmed ElShafee, Karim Alaa Hamed," Design and Implementation of a WiFi Based Home Automation System", International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol: 6, No: 8, 2012

[4] Vaishnavi S. Gunge, Pratibha S. Yalagi, Walchand Institute of Technology Solapur. "Smart Home Automation: A Literature Review". International Journal of Computer Applications (0975 – 8887). National Seminar on Recent Trends in Data Mining (RTDM 2016).

[5] ERIC GRIFFITH. (2017) "The Best Smart Home Devices of 2017". http://uk.pcmag.com/digital-home/85/feature/the-best-smart-home-devices-of-2017

[6] "Arduino UNO"

http://digital.csic.es/bitstream/10261/127788/7/D-c-%20Arduino%20uno.pdf

[7] From ELEC Freaks, http://www.micropik.com/PDF/HCSR04.pdf

[8] Dmitri Pahhomov (2015)."Low cost motor controller (H-Bridge) for LYNX robot platform (based on L298N chip)"

https://lynx2015.wordpress.com/2015/07/28/motor-controller/

[9] Reichenstein7,（2014）"Arduino Modules - L298N Dual H-Bridge Motor Controller"

http://www.instructables.com/id/Arduino-Modules-L298N-Dual-H-Bridge-Motor-Controll/

[10] Fig.4 Reichenstein7,（2014）“Arduino Modules - L298N Dual H-Bridge Motor Controller ”
http://www.instructables.com/id/Arduino-Modules-L298N-Dual-H-Bridge-Motor-Controll/

[11] Timothy Hirzel. “Arduino Home”, https://www.arduino.cc/en/Tutorial/PWM

[12] “TowerPro online shop”, http://www.towerpro.com.tw/product/sg90-7/

[13] “The handbook of AM2302” (2015).

https://wenku.baidu.com/view/4af942567375a417876f8f4e.html

[14] “The handbook of nRF24L01 wireless module” (2013).

https://wenku.baidu.com/view/0a1e861a2af90242a895e54d.html

[15] “The handbook of MG995 servo” (2011).

https://wenku.baidu.com/view/ae9c222e2af90242a895e593.html

[16] “The handbook of DS3231 clock module” (2015).

http://www.360doc.com/content/15/0718/03/12109864_485616995.shtml

[17] HuiJing Electronic (2016) http://www.hjmcu.com/

9 Appendices

9.1 Smart car

```
#include <Servo.h>
#include <DistanceSRF04.h>
#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>
#include "DHT.h"

#define DHTPIN 2          // what pin we're connected to
#define DHTTYPE DHT22      // DHT 22    (AM2302)

DHT dht(DHTPIN, DHTTYPE);

DistanceSRF04 Dist;
int distance;
int EN1 = 14;
int EN2 = 15;
int EN3 = 16;
int EN4 = 17;

//int LED1 = 2;
//int LED2 = 4;
int EA = 3;
int EB = 5;

//int LaBa = 12;


uint8_t EN1Status = LOW;
uint8_t EN2Status = LOW;
uint8_t EN3Status = LOW;
uint8_t EN4Status = LOW;

Servo servoX;
Servo servoY;


byte serialIn = 0;
```

```
byte commandAvailable = false;
String strReceived = "";
byte radar = false;
byte light = false;


byte servoXCenterPoint = 94;
byte servoYCenterPoint = 88 ;

byte servoXmax = 170;
byte servoYmax = 130;
byte servoXmini = 10;
byte servoYmini = 10;
byte servoXPoint = 0;
byte servoYPoint = 0;

byte leftspeed = 0;
byte rightspeed = 0;

byte servoStep = 4;

int maxtem = 30;
int mintem = 20;
int lightPin = A5;


void setup()
{

      Dist.begin(9,8);
      servoX.attach(10);
      servoY.attach(6);

      pinMode(EN1, OUTPUT);
      pinMode(EN2, OUTPUT);
      pinMode(EN3, OUTPUT);
      pinMode(EN4, OUTPUT);

   pinMode(lightPin, INPUT);


      servo_test();
      Serial.begin(9600);
      Mirf.cePin = 4 ;
```

```
        Mirf.csnPin = 7;
        Mirf.spi = &MirfHardwareSpi;
        Mirf.init();


        Mirf.setRADDR((byte *)"Sen01");

        Mirf.payload = sizeof(unsigned int);
              Mirf.channel = 3;
        Mirf.config();


        Serial.println("I'm Sender...");

         dht.begin();
}
int tem = 0, lig = 0;
unsigned int adata = 0;
void loop()
{
        if (radar == true)
        {
              distance = Dist.getDistanceCentimeter();
              if (distance <= 5 & distance > 1){
                    hou();
                    delay(100);
                    ting();
                    distance = 0;
              }
        }
        getSerialLine();
        if (commandAvailable) {
              processCommand(strReceived);
              strReceived = "";
              commandAvailable = false;
        }

tem = dht.readTemperature();
lig = analogRead(lightPin);
   Serial.println(lig );
if (lig >= 100)
{
   if ( tem <= mintem )
   {
```

```
        adata = 1;
    }
    else if ( tem > mintem && tem < maxtem)
    {
        adata = 3;
    }
    else if ( tem >= maxtem)
    {
        adata = 5;
    }
}
else if (lig < 100)
{
    if ( tem <= mintem )
    {
        adata = 2;
    }
    else if ( tem > mintem && tem < maxtem)
    {
        adata = 4;
    }
    else if ( tem >= maxtem)
    {
        adata = 6;
    }
}


        byte data[Mirf.payload];


        data[0] = adata & 0xFF;
        data[1] = adata >> 8;
             Mirf.setTADDR((byte *)"Rec01");
        Mirf.send(data);

        while(Mirf.isSending()) {
                Serial.println("wait");
           }

}


void getSerialLine()
```

```
{
        while (serialIn != '\r')
    {
        if (!(Serial.available() > 0))
        {
            return;
        }

        serialIn = Serial.read();
        if (serialIn != '\r') {

            if (serialIn != '\n'){
                char a = char(serialIn);
                strReceived += a;
            }

        }
    }
        if (serialIn == '\r') {
        commandAvailable = true;
        serialIn = 0;
    }

}


void processCommand(String input)
{
    String command = getValue(input, ' ', 0);
    byte iscommand = true;
    int val;
    if (command == "MD_Qian")
    {
        qian();
    }
    else if (command == "MD_Hou")
    {
        hou();
    }
    else if (command == "MD_Zuo")
    {
        zuo();
    }
    else if (command == "MD_You")
```

```
{
    you();
}
else if (command == "MD_Ting")
{
    ting();
}
else if (command == "MD_SD")
{
    val = getValue(input, ' ', 1).toInt();
    leftspeed = val;
    val = getValue(input, ' ', 2).toInt();
    rightspeed = val;
}
else if (command == "DJ_CS")
{
    servo_test();
}
else if (command == "DJ_Shang")
{
    servo_up();
}
else if (command == "DJ_Xia")
{
    servo_down();
}
else if (command == "DJ_Zuo")
{
    servo_left();
}
else if (command == "DJ_You")
{
    servo_right();
}
else if (command == "DJ_Zhong")
{
    servo_center();
}
else if (command == "DJ_CZ_JD")//VerticalRotation
{
    val = getValue(input, ' ', 1).toInt();
}
else if (command == "DJ_SP_JD")//Horizontal rotation
{
```

```
            val = getValue(input, ' ', 1).toInt();
        }
        else if (command == "LED_Status")
        {
            val = getValue(input, ' ', 1).toInt();
            light = val == 0 ? false : true;
        }
        else if (command == "LED_Status_Swich")
        {
            light = light ? false : true;
        }
        else if (command == "Radar_Status")
        {
            val = getValue(input, ' ', 1).toInt();
            radar = val ? true : false;
        }
        else if (command == "Radar_Status_Swich")
        {
            radar = radar ? false : true;
        }
        else if (command == "LaBa_Start"){

        }
        else if (command == "LaBa_Stop"){
        }
        else
        {
            iscommand = false;
        }

        if (iscommand){
            SendMessage("cmd:" + input);
            SendStatus();
        }

}

String getValue(String data, char separator, int index)
{
        int found = 0;
        int strIndex[] = {
            0, -1 };
        int maxIndex = data.length() - 1;
```

```
    for (int i = 0; i <= maxIndex && found <= index; i++){
        if (data.charAt(i) == separator || i == maxIndex){
            found++;
            strIndex[0] = strIndex[1] + 1;
            strIndex[1] = (i == maxIndex) ? i + 1 : i;
        }
    }

    return found>index ? data.substring(strIndex[0], strIndex[1]) : "";
}

void servo_test(void) {
    int nowcornerY = servoY.read();
    int nowcornerX = servoX.read();
    servo_Vertical(servoYmini);
    delay(500);
    servo_Vertical(servoYmax);
    delay(500);
    servo_Vertical(servoYCenterPoint);
    delay(500);
    servo_Horizontal(servoXmini);
    delay(500);
    servo_Horizontal(servoXmax);
    delay(500);
    servo_Horizontal(servoXCenterPoint);
    delay(500);
    servo_center();
}
void servo_right(void)
{
    int servotemp = servoX.read();
    servotemp -= servoStep;
    servo_Horizontal(servotemp);
}
void servo_left(void)
{
    int servotemp = servoX.read();
    servotemp += servoStep;
    servo_Horizontal(servotemp);
}
void servo_down(void)
{
    int servotemp = servoY.read();
    servotemp += servoStep;
```

```
    servo_Vertical(servotemp);
}
void servo_up(void)
{
    int servotemp = servoY.read();
    servotemp -= servoStep;
    servo_Vertical(servotemp);
}
void servo_center(void)
{
    servo_Vertical(servoYCenterPoint);
    servo_Horizontal(servoXCenterPoint);
}
void servo_Vertical(int corner)
{
    int cornerY = servoY.read();
    if (cornerY > corner) {
        for (int i = cornerY; i > corner; i = i - servoStep) {
            servoY.write(i);
            servoYPoint = i;
            delay(50);
        }
    }
    else {
        for (int i = cornerY; i < corner; i = i + servoStep) {
            servoY.write(i);
            servoYPoint = i;
            delay(50);
        }
    }
    servoY.write(corner);
    servoYPoint = corner;
}
void servo_Horizontal(int corner)
{
    int i = 0;
    byte cornerX = servoX.read();
    if (cornerX > corner) {
        for (i = cornerX; i > corner; i = i - servoStep) {
            servoX.write(i);
            servoXPoint = i;
            delay(50);
        }
    }
```

```
        else {
            for (i = cornerX; i < corner; i = i + servoStep) {
                servoX.write(i);
                servoXPoint = i;
                delay(50);
            }
        }
        servoX.write(corner);
        servoXPoint = corner;
}

void qian(void)
{
        EN1Status = LOW;
        EN2Status = HIGH;
        EN3Status = LOW;
        EN4Status = HIGH;
        SetEN();
}
void hou(void)
{
        EN1Status = HIGH;
        EN2Status = LOW;
        EN3Status = HIGH;
        EN4Status = LOW;
        SetEN();
}
void you(void)
{
        EN1Status = LOW;
        EN2Status = HIGH;
        EN3Status = HIGH;
        EN4Status = LOW;
        SetEN();
}
void zuo(void)
{
        EN1Status = HIGH;
        EN2Status = LOW;
        EN3Status = LOW;
        EN4Status = HIGH;
        SetEN();
}
void ting(void)
```

```
{
    leftspeed = 0;
    rightspeed = 0;
    EN1Status = LOW;
    EN2Status = LOW;
    EN3Status = LOW;
    EN4Status = LOW;
    SetEN();
}

void SetEN(){
    analogWrite(EA, leftspeed);
    analogWrite(EB, rightspeed);
    digitalWrite(EN1, EN1Status);
    digitalWrite(EN2, EN2Status);
    digitalWrite(EN3, EN3Status);
    digitalWrite(EN4, EN4Status);
}


void SendStatus(){

    String out = "";
    out += EN1Status;
    out += ",";
    out += EN2Status;
    out += ",";
    out += EN3Status;
    out += ",";
    out += EN4Status;
    out += ",";
    out += leftspeed;
    out += ",";
    out += rightspeed;
    out += ",";
    out += servoXPoint;
    out += ",";
    out += servoYPoint;
    out += ",";
    out += radar;
    out += ",";
    out += light;
    SendMessage(out);
}
```

```
void SendMessage(String data){
      Serial.println(data);
}
```

9.2 The lighting module

```
#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>
int lightPin = 6;
          unsigned int adata = 0, usefuladata = 0;

void setup()
{
   pinMode(lightPin, OUTPUT);
      Serial.begin(9600);

      Mirf.cePin = 9;
      Mirf.csnPin = 10;
      Mirf.spi = &MirfHardwareSpi;
      Mirf.init();
      Mirf.setRADDR((byte *)"Rec01");
      Mirf.payload = sizeof(unsigned int);
      Mirf.channel = 3;
      Mirf.config();

      Serial.println("I'm Receiver...");
}

void loop()
{

          byte data[Mirf.payload];
      if(Mirf.dataReady())          {
            Mirf.getData(data);
            adata = (unsigned int)((data[1] << 8) | data[0]);

            Serial.println(adata);
                  if(adata ==1 || adata == 3 || adata == 5    )
            {
```

```
            usefuladata = 1;
              digitalWrite(lightPin, HIGH);
        }
        else if (adata ==2 || adata == 4 || adata == 6    )
        {
              usefuladata = 0;
             digitalWrite(lightPin, LOW);
        }


            Serial.println(adata);

    }


delay(200);
    }
```

9.3 The heating module

```
#include <Wire.h>
#include "RTClib.h"

RTC_DS3231 rtc;
#include <Servo.h>
#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>


    unsigned int adata = 0, total = 0;
    int minutes = 0;
    int pos = 0;

    Servo myservo;
void setup()
{

    myservo.attach(3);
```

```
    Serial.begin(9600);

      delay(3000);

        if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);
  }
    if (rtc.lostPower()) {
    Serial.println("RTC lost power, lets set the time!");
        rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }

    Mirf.cePin = 9;
    Mirf.csnPin = 10;
    Mirf.spi = &MirfHardwareSpi;
    Mirf.init();
    Mirf.setRADDR((byte *)"Rec01");
    Mirf.payload = sizeof(unsigned int);
    Mirf.channel = 3;
    Mirf.config();
    Serial.println("I'm Receiver...");
}

void loop()
{
    byte data[Mirf.payload];
    if(Mirf.dataReady())
    {
        Mirf.getData(data);
        adata = (unsigned int)((data[1] << 8) | data[0]);

        Serial.println(adata);
   if (adata == 1 || adata == 2)
    {
      work2();

    }
     else if (adata == 3 || adata == 4)
     {
      work();
     }
      else if (adata == 5 || adata ==6)
      {
```

```
        }

      }

}
void work()
{
       for(pos = 0; pos < 180; pos += 1)
  {
            myservo.write(pos);
     delay(15);
  }
  for(pos = 180; pos>=1; pos-=1)
  {
     myservo.write(pos);
     delay(15);
  }
            DateTime now = rtc.now();
                Serial.print("adata=");
              Serial.println(adata);
              minutes = now.minute();
              Serial.print("minutes=");
            Serial.println(minutes);
          while(total <= 15)
          {
            Serial.println(total);
            DateTime now = rtc.now();
            if ( minutes != now.minute())
            {
              total = total +1;
              Serial.println(total);
              if( minutes == 59)
              {
                minutes = 0;
              }
              else
              {
                minutes = minutes + 1;
              }
              delay(300);
            }
          }
        if (total == 16 )
```

```
            total = 0;
  }

void work2()
{
  for(i=1; i<5; i++)
    {
         for(pos = 0; pos < 180; pos += 1)
    {
                myservo.write(pos);
        delay(15);
    }
    for(pos = 180; pos>=1; pos-=1)
    {
        myservo.write(pos);
        delay(15);
    }
  }
                DateTime now = rtc.now();
                     Serial.print("adata=");
                   Serial.println(adata);
                   minutes = now.minute();
                   Serial.print("minutes=");
                Serial.println(minutes);
              while(total <= 60)
              {
                Serial.println(total);
                DateTime now = rtc.now();
                if ( minutes != now.minute())
                {
                   total = total +1;
                   Serial.println(total);
                   if( minutes == 59)
                   {
                      minutes = 0;
                   }
                   else
                   {
                      minutes = minutes + 1;
                   }
                   delay(300);
                }
              }
            if (total == 61 )
```

```
        total = 0;
}
```