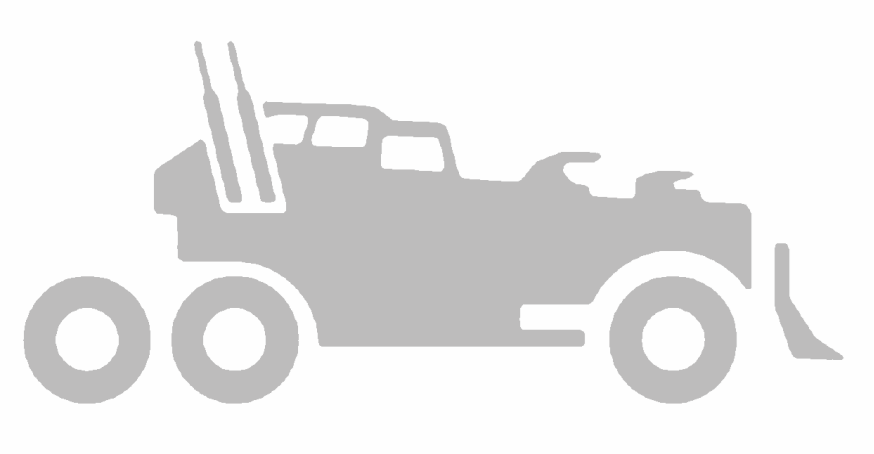
电子科技大学格拉斯哥学院
Glasgow College, UESTC

Team Design Project and Skills

2020-21, semester II

Lab Book



Team : Mad Max (No.37)

Supervisor

Dr Wasim Ahmad

UESTC 3010 TDPS – Team Information Form

Team Number		37				
Team Name		Mad Max				
No	**UESTC ID**	**UoG ID**	**Full Name**	**Role/Responsibility**	**Degree program**	**Gender**
1	2018190606034	2429391X	Zhangchen Xu	Team Coordinator + HC-12 Comm	CE	Male
2	2018190502024	2429497X	Ziyi Xie	Image Group Coordinator + Route Tracing	IE	Male
3	2018190607005	2429400Y	Shubo Yang	Image: Tracing + Ultrasonic Sensors	CE	Female
4	2018190505002	2429592Y	Jingluan Yang	Image: Beacon Recognition	IE	Female
5	2018190505005	2429595H	Qinlin Han	Image: Colour Recognition	IE	Female
6	2018190505006	2429596L	Xinyue Li	Car: Wiring+Claw	IE	Female
7	2018190602001	2429207Y	Yuchen Yao	Car: PID Algorithm	CE	Female
8	2018190606004	2429361Z	Weizhe Zhao	Car: L298 Driver + Gyroscope	CE	Female
9	2018190502030	2429503L	Ziyang Long	Car Group Coordinator + PCB design	IE	Male
10	2018190505036	2429626F	Chenyang Fan	Car: Speed Feedback	IE	Male

Note: To achieve high efficiency, our group is divided into two sub-teams: **Car Group** and **Image Group**. The work of the Car Group is to control the car movement, covering the work of assembling the car, wiring, driving circuit design and manufacturing, PID algorithm, etc. The Image group is responsible for tracing, color and beacon recognition, communication and control of the claws. Each group has a coordinator who is responsible for coordinating the work within the group and communicating with the other group. The team leader is responsible for coordinating the overall project progress.

Mad Max – Project Overview

Overall Project Plan

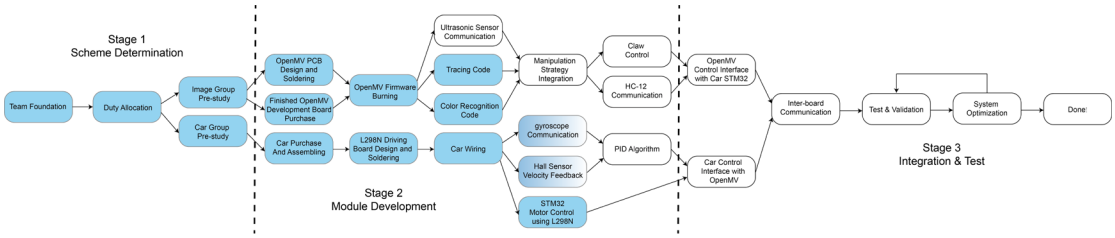
We divided the overall project plan of the project into three stages as shown in the figure below.

Stage 1 is Scheme Determination. In this stage, we make work division and allocate our duties as we mentioned before, and we also do some preliminary research within two small groups. At the end of the stage, the overall system structure and the solutions for patios is determined.

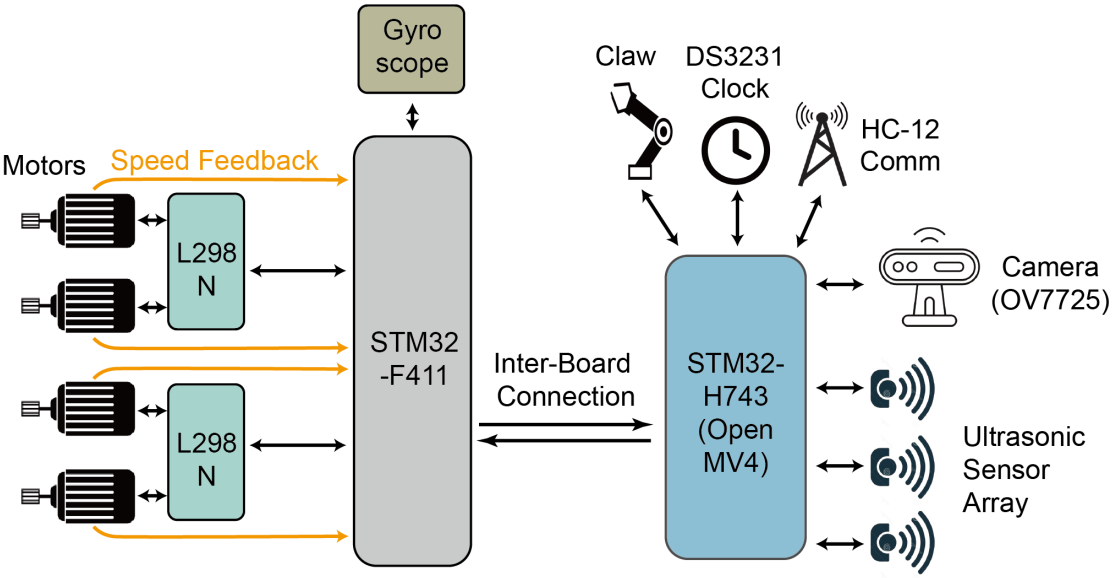
Stage 2 is Module Development. In this stage, we design and make the modules needed in this project. We also draw and fabricate the motor driver board and OpenMV board. Together with existing models, we write the tracing, color recognition as well as motor control codes. The manipulation strategy for Image Group and PID algorithms for Car Group are also developed here.

The Stage 3 is Integration & Test. In this stage, both teams design software interfaces for inter-board communication. Then, we start our test & validation process of the whole system. During this time, the algorithms and codes are continuously optimized through field testing.

Until 2021.6.13, we've finished all part of our project.



System Structure



We used two major boards control the car. One is a STM32F411 development board which is responsible for controlling the car movement and adjusting car's position. Motors with Hall velocity sensor and two L298N driver modules are connected with this board, and there is also a gyroscope module which provides position for PID algorithm. We can regard these modules together as a motion controller.

The other board is a STM32H743 development board with OpenMV4 firmware, aiming at providing operation instructions to the system, and it's like a direction controller (or a wheel) that send direction order to the car. The OpenMV4 is connected to an OV7725 camera and processes image data for tracing and color detection. Furthermore, it collects distance information via ultrasonic sensor array for synthetical decision-making and is responsible for communication with laptops via a HC-12 communication module. In terms of inter-board communication, we use UART, an asynchronous serial communication protocol.

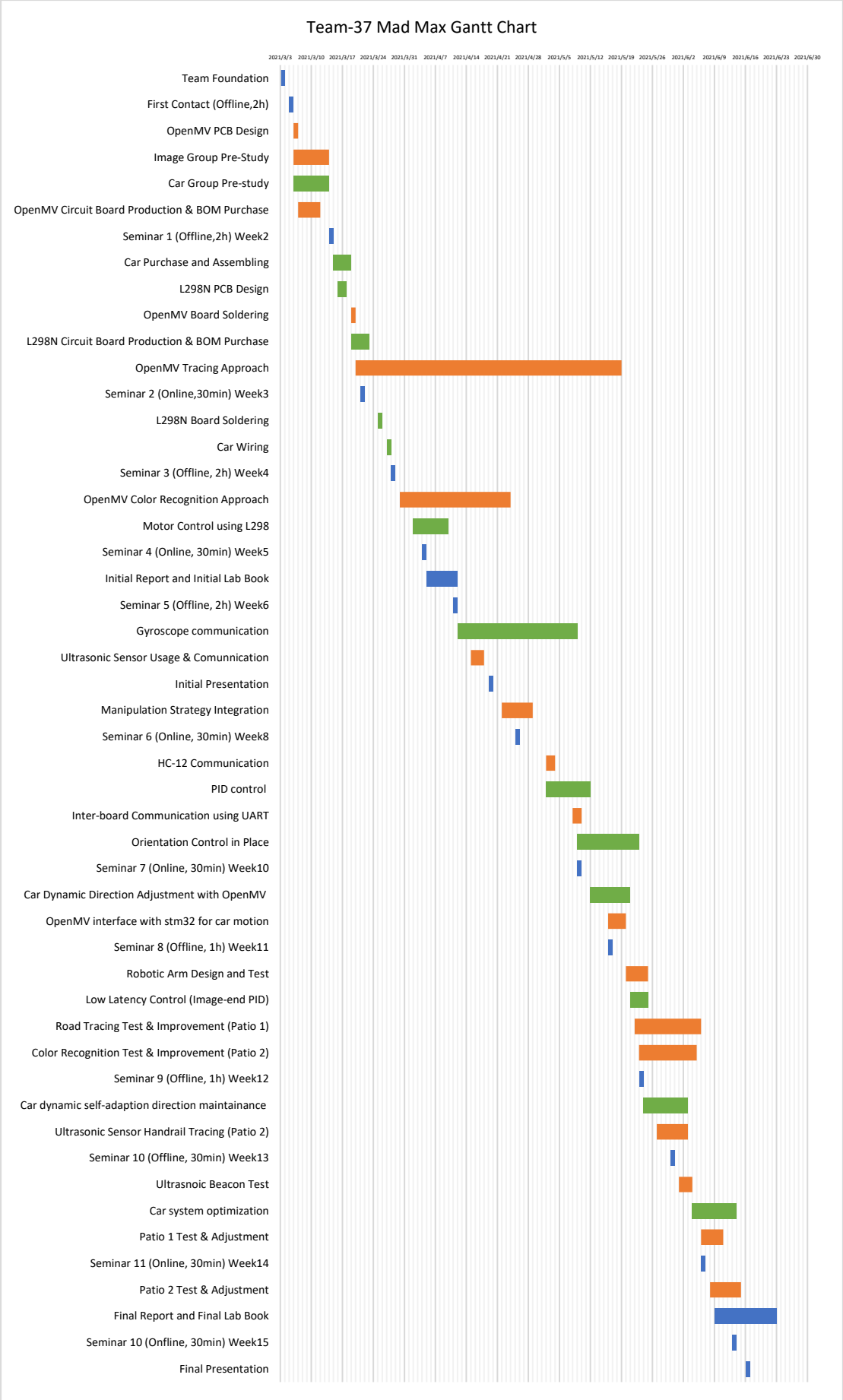
Speaking of software, we choose microPython as project's embedded development language, since it is a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library and is optimized to run on microcontrollers as well as in constrained environments.

Mad Max – Time Arrangement

Here we demonstrate the time arrangement of our project until 2021.6.16.

Mission	Start Time	Days	Group
Team Foundation	2021/3/3	1	All
First Contact (Offline,2h)	2021/3/5	1	All
OpenMV PCB Design	2021/3/6	1	Image
Image Group Pre-Study	2021/3/6	8	Image
Car Group Pre-study	2021/3/6	8	Car
OpenMV Circuit Board Production & BOM Purchase	2021/3/7	5	Image
Seminar 1 (Offline,2h) Week2	2021/3/14	1	All
Car Purchase and Assembling	2021/3/15	4	Car
L298N PCB Design	2021/3/16	2	Car
OpenMV Board Soldering	2021/3/19	1	Image
L298N Circuit Board Production & BOM Purchase	2021/3/19	4	Car
OpenMV Tracing Approach	2021/3/20	60	Image
Seminar 2 (Online,30min) Week3	2021/3/21	1	All
L298N Board Soldering	2021/3/25	1	Car
Car Wiring	2021/3/27	1	Car
Seminar 3 (Offline, 2h) Week4	2021/3/28	1	All
OpenMV Color Recognition Approach	2021/3/30	25	Image
Motor Control using L298	2021/4/2	8	Car
Seminar 4 (Online, 30min) Week5	2021/4/4	1	All
Initial Report and Initial Lab Book	2021/4/5	7	All
Seminar 5 (Offline, 2h) Week6	2021/4/11	1	All
Gyroscope communication	2021/4/12	27	Car
Ultrasonic Sensor Usage & Comunnication	2021/4/15	3	Image
Initial Presentation	2021/4/19	1	All
Manipulation Strategy Integration	2021/4/22	7	Image
Seminar 6 (Online, 30min) Week8	2021/4/25	1	All
HC-12 Communication	2021/5/2	2	Image
PID control	2021/5/2	10	Car
Inter-board Communication using UART	2021/5/8	2	Image
Orientation Control in Place	2021/5/9	14	Car
Seminar 7 (Online, 30min) Week10	2021/5/9	1	All
Car Dynamic Direction Adjustment with OpenMV	2021/5/12	9	Car
OpenMV interface with stm32 for car motion	2021/5/16	4	Image

Seminar 8 (Offline, 1h) Week11	2021/5/16	1	All
Robotic Arm Design and Test	2021/5/20	5	Image
Low Latency Control (Image-end PID)	2021/5/21	4	Car
Road Tracing Test & Improvement (Patio 1)	2021/5/22	15	Image
Color Recognition Test & Improvement (Patio 2)	2021/5/23	13	Image
Seminar 9 (Offline, 1h) Week12	2021/5/23	1	All
Car dynamic self-adaption direction maintainance	2021/5/24	10	Car
Ultrasonic Sensor Handrail Tracing (Patio 2)	2021/5/27	7	Image
Seminar 10 (Offline, 30min) Week13	2021/5/30	1	All
Ultrasnoic Beacon Test	2021/6/1	3	Image
Car system optimization	2021/6/4	10	Car
Patio 1 Test & Adjustment	2021/6/6	5	Image
Seminar 11 (Online, 30min) Week14	2021/6/6	1	All
Patio 2 Test & Adjustment	2021/6/8	7	Image
Final Report and Final Lab Book	2021/6/9	14	All
Seminar 10 (Onfline, 30min) Week15	2021/6/13	1	All
Final Presentation	2021/6/16	1	All


Team-37 Mad Max Gantt Chart
2021/3/3
2021/3/10
2021/3/17
2021/3/24
2021/3/31
2021/4/7
2021/4/14
2021/4/21
2021/4/28
2021/5/5
2021/5/12
2021/5/19
2021/5/26
2021/6/2
2021/6/9
2021/6/16
2021/6/23
2021/6/30
Team Foundation
First Contact (Offline,2h)
OpenMV PCB Design
Image Group Pre-Study
Car Group Pre-study
OpenMV Circuit Board Production & BOM Purchase
Seminar 1 (Offline,2h) Week2
Car Purchase and Assembling
L298N PCB Design
OpenMV Board Soldering
L298N Circuit Board Production & BOM Purchase
OpenMV Tracing Approach
Seminar 2 (Online,30min) Week3
L298N Board Soldering
Car Wiring
Seminar 3 (Offline, 2h) Week4
OpenMV Color Recognition Approach
Motor Control using L298
Seminar 4 (Online, 30min) Week5
Initial Report and Initial Lab Book
Seminar 5 (Offline, 2h) Week6
Gyroscope communication
Ultrasonic Sensor Usage & Comunnication
Initial Presentation
Manipulation Strategy Integration
Seminar 6 (Online, 30min) Week8
HC-12 Communication
PID control
Inter-board Communication using UART
Orientation Control in Place
Seminar 7 (Online, 30min) Week10
Car Dynamic Direction Adjustment with OpenMV
OpenMV interface with stm32 for car motion
Seminar 8 (Offline, 1h) Week11
Robotic Arm Design and Test
Low Latency Control (Image-end PID)
Road Tracing Test & Improvement (Patio 1)
Color Recognition Test & Improvement (Patio 2)
Seminar 9 (Offline, 1h) Week12
Car dynamic self-adaption direction maintainance
Ultrasonic Sensor Handrail Tracing (Patio 2)
Seminar 10 (Offline, 30min) Week13
Ultrasnoic Beacon Test
Car system optimization
Patio 1 Test & Adjustment
Seminar 11 (Online, 30min) Week14
Patio 2 Test & Adjustment
Final Report and Final Lab Book
Seminar 10 (Onfline, 30min) Week15
Final Presentation

Week 0 – Group First Meeting

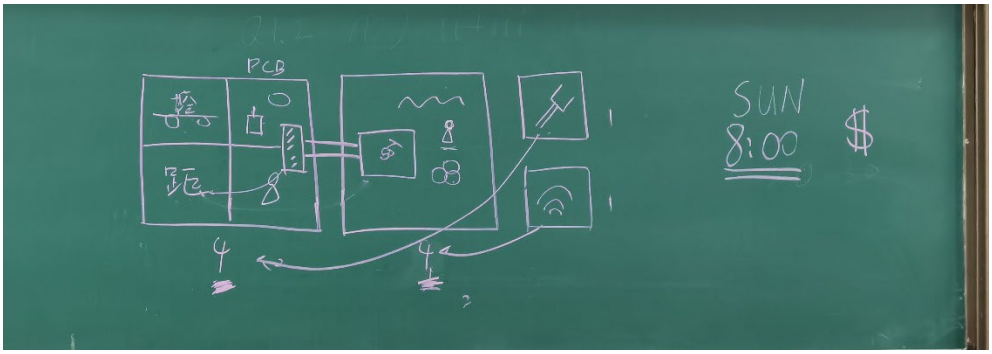
Title	Group First Meeting and Field Research		
Group	All	**Week**	Week 1
Recorder	Zhangchen Xu	**Date**	2021.3.5

On March 5, 2021, the members of this group have their first meeting at the UESTC's Liren Building. The main purpose of this meeting is to familiarize the group members and to have a preliminary plan for the TDPS project program. After the meeting, the group conducted a field study of the site.

At first, the 10 members determine the name of this group as **Mad Max** by show of hands. What a nice name!

We then studied the division of labour in the group. We first define 5 major modules needed for the project, that is: Tracking Module, Positioning Module, Colour Recognition Module, Positioning Module and Communication Module.

After analysis and discussion, we think the group can be divided into **image group** and **car group**, which deal with colour, image recognition and control car movement respectively. Then according to everyone's interest, we divided the 10 group members into two groups equally and elected a coordinator in each group to coordinate the work.



Next, we set up a **weekly meeting routine** on every Sunday. During the next week, the teams will research possible options and compiling relevant information. At the meeting next Sunday, we will reach a consensus on the overall solution.

After the meeting, the group members made a visit to the field together and exchanged their views.

Title	Why We Choose STM32		
Group	Car Group	**Week**	Week 1
Recorder	Yuchen Yao	**Date**	2021.3.6

In this project, microcontrollers are involved to conduct data processing for image processing and motor driving. These two high computational tasks are distributed to separate microcontrollers for the sake of faster processing speed. Since Raspberry Pi and Arduino are banned for this project, stm32 series microcontrollers are chosen as competitive candidates.

stm32 family offers products combining very high performance, real-time capabilities, digital signal processing, low-power / low-voltage operation, and connectivity, while maintaining full integration and ease of development. They are based on ARM cortex and possess varies of functional modules and interfaces for communication including SPI, USART, AD/DA and IIC, which can communicate with MU6050 Gyroscope. Among the series, STM32F411CE is picked out for its high-performance access line and large flash memory size (512 Kbytes). Therefore, it is expected that the intelligent car can exhibit steady performance with real-time response under the control of STM32F411CE.

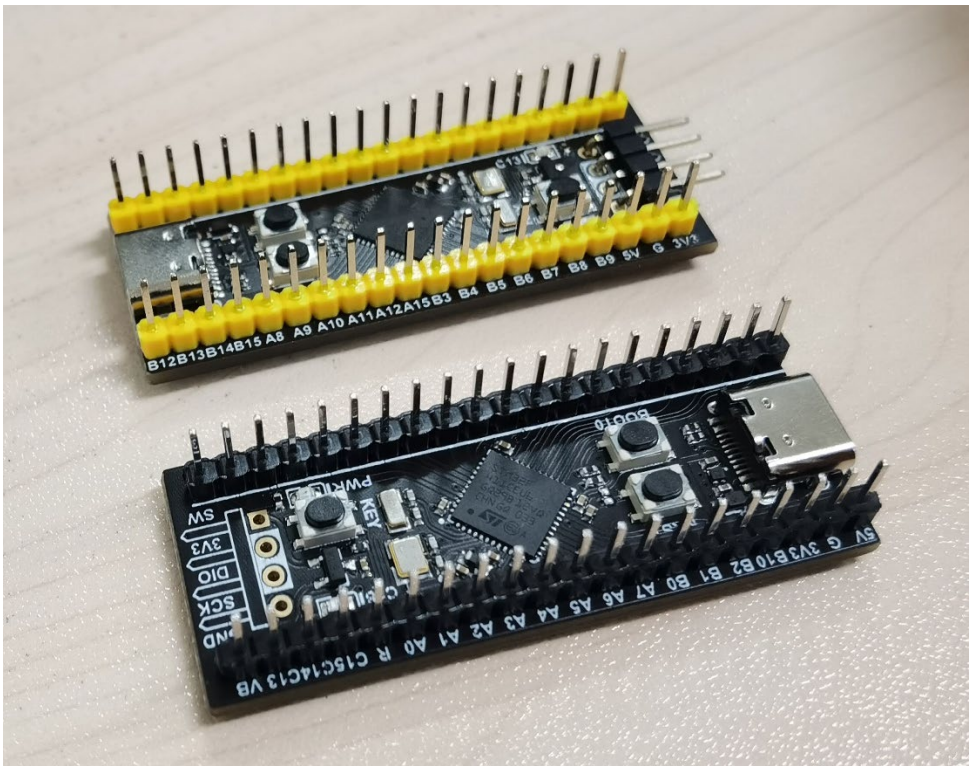


Fig: STM32F411 Minimal System

Title	Patio 1 and Patio 2 Initial Plan		
Group	All	**Week**	Week 2
Recorder	Jinluan Yang Qinlin Han	**Date**	2021.3.9

Patio 1 has three tasks, where the first task is to follow the path, the second task is to find the bridge and cross on top of it and the third task is to find the gate, go through and stop. In this patio, we mainly use edge detection and color (beacon) recognition techniques. Figure 1 below shows the route and beacon position of Patio 1, where beacon 1 is used to enable the car to turn right and run onto the bridge, and beacon 2 is used for stopping the car. Figure 2 shows complete flow chart of Patio 1.

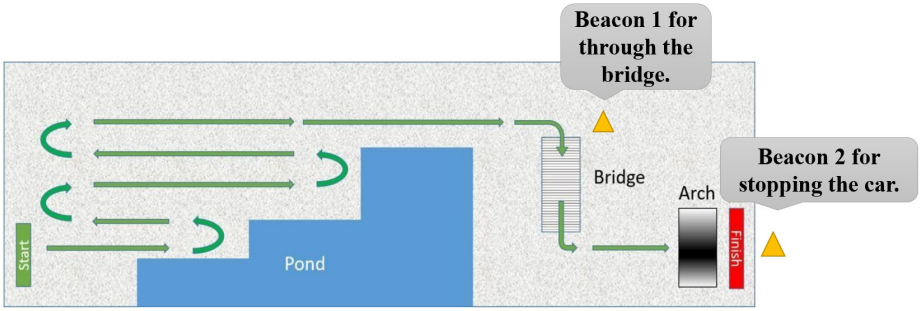


Figure 1 Route and Beacon of Patio 1

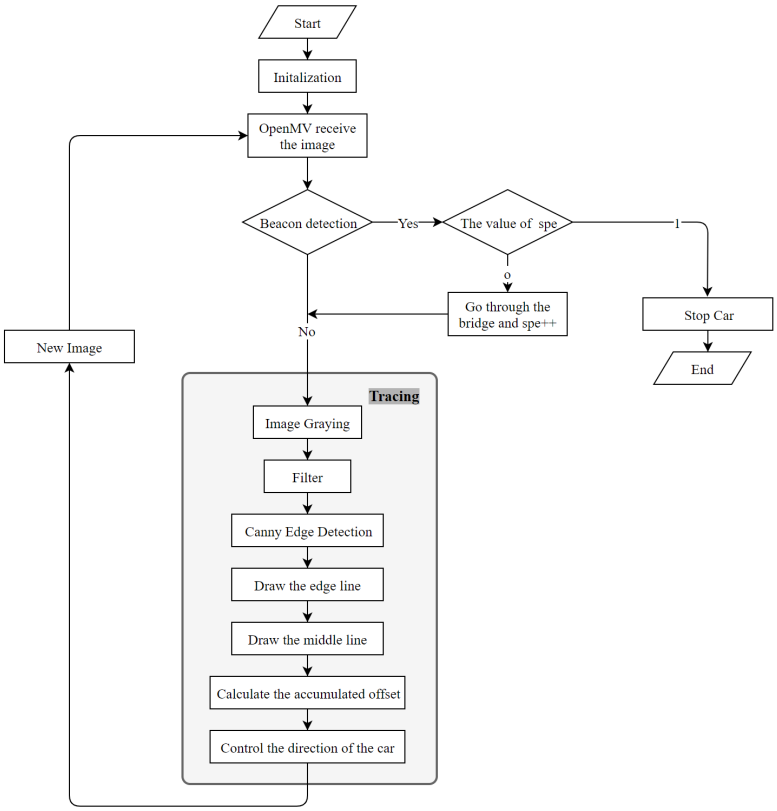


Figure 2 Flow Chart of Patio 1

Patio 2 also has three main tasks. The mission of task 1 is to detect a particular color and automatically move in a specific direction. Task 2 is to automatically move from the red line in task 1 to a position near the lake and release an item to the lake. Task 3 requires car to return back to planter area and transmit a message to a laptop. After receiving reply message from the laptop, the car needs to keep moving until the destination.

In this patio, we use more techniques, e.g., tracing, color recognition, distance detection, robotic arm control and HC-12 communication. As shown on Figure 3, beacon 1 is set near the planter for the car to turn right and align with the feeding port, and beacon 2 is set in the feeding port for control precision on a smaller scale. Figure 4 shows complete flow chart of Patio 2.

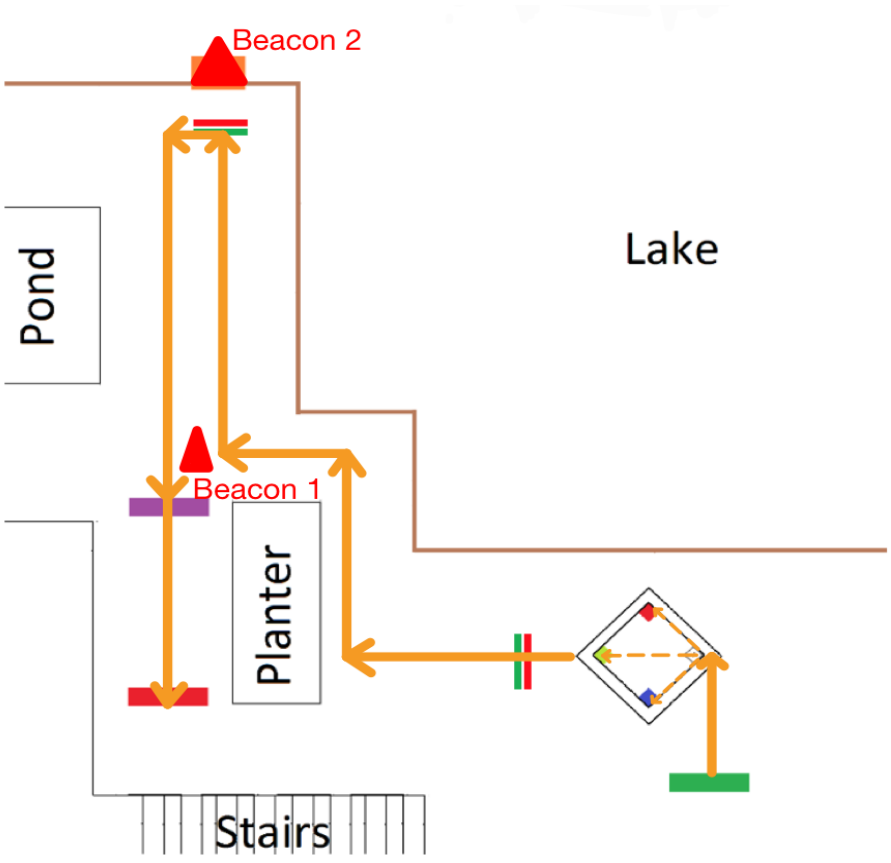


Figure 3 Route and Beacon of Patio 2

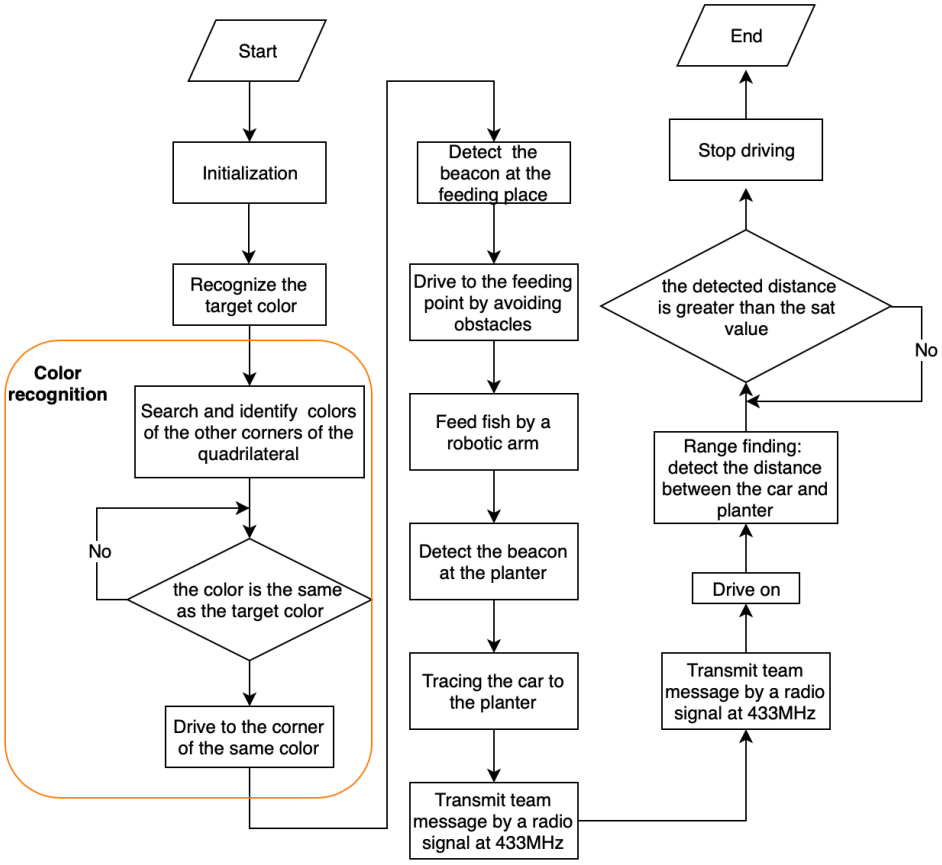


Figure 4 Flow Chart of Patio 2

Title	Drive Scheme of the Robot Car		
Group	Car Group	**Week**	Week 2
Recorder	Weizhe Zhao	**Date**	2021.3.13

In terms of drive scheme of the robot car, we evaluated two-wheel drive and three-wheel (with one omni-directional wheel), and finally chose **4WD (four-wheel drive)** scheme. There are three main reason for the choice. First, as the car is expected to drive across the cobblestone pavement in Patio 2, 4WD is preferred for its outstanding off-roading performance, with better traction on rough roads than its 2WD counterparts. Second, 4WD cars are able to make sharp turns, since both the front and rear wheels can be designed to take part in turning. Moreover, 4WD is switchable to 2WD when necessary.

4WD scheme has some shortcomings, though, such as higher cost and energy consumption. However, the cons are almost immaterial when compared with the pros. Consequently, we decided to follow the 4WD scheme.

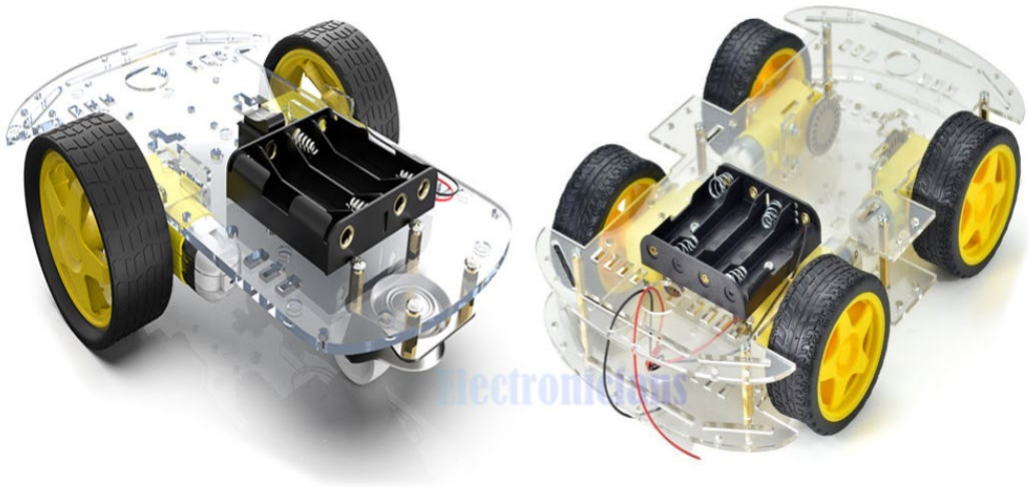
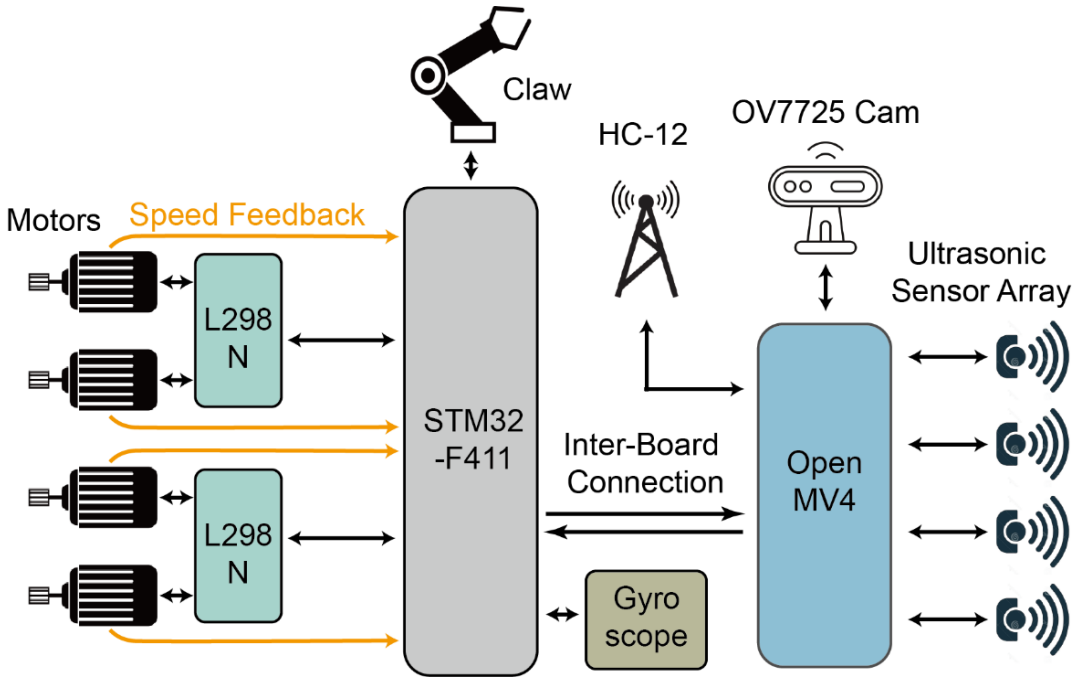


Figure: 4WD vs 2WD

Week 1 – Group Seminar 1

Title	Group Seminar 1: System Overall Structure		
Group	All	**Week**	Week 2
Recorder	Zhangchen Xu	**Date**	2021.3.14

On March 14, 2021, Mad Max members met at A1-305 in the main building for the first offline seminar. The main topic of this seminar is the overall system architecture of the trolley. After discussion, we agreed on **OpenMV** as the technology for image processing, and **4WD scheme** as the vehicle structure. The motion of the car will be controlled by a Stm32 developer board. The system structure is demonstrated in the figure below (Note that we omit power supply and power cables).



For our system, here's an analogy. We can divide the whole system into two separate subsystems (i.e., direction controller and motion controller). The OpenMV board and its peripheral components are like a direction controller that controls the direction of the trolley by sending direction instructions, while the STM32F411 board acts as a motion controller which receives direction instructions and apply different signals to the motor driver to perform different motions (e.g., turn left or turn right).

From the perspective of software, we choose **microPython** as project's embedded development language, which is a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library and is optimized to run on microcontrollers and in constrained environments. In our system, both STM32F411 development board and OpenMV4 support microPython,

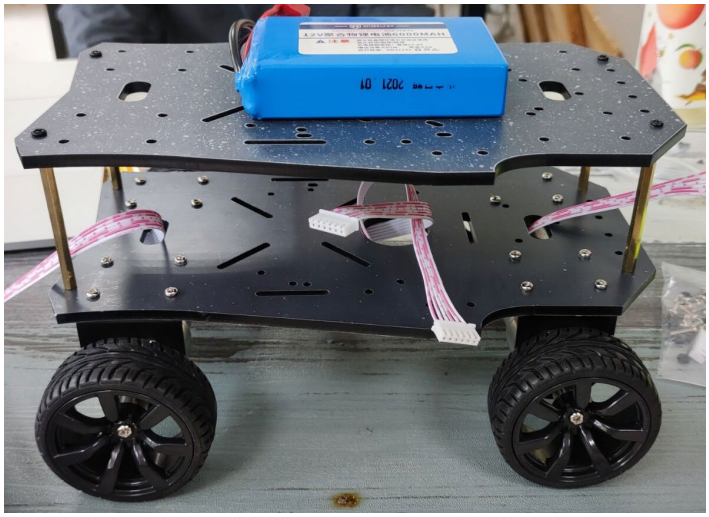
so choosing microPython can unify the programming language in the project and improve efficiency.

In this seminar, we also financially qualify the price of each component.

Title	How to Build the Frame of the Car?		
Group	Car Group	**Week**	Week 3
Recorder	Chenyang Fan	**Date**	2021.3.19

Initially, I need to get the screws, screwdrivers, wheels, motor shaft and some other components ready to build the car.

The following step is using M3x12 screws and M3 nuts to connect the motor bracket and the acrylic base plate. Then, I utilized M3x4 screws to connect the motor and the motor bracket (the motor shaft is connected at the end far away from the acrylic base plate). After that, M3x6 screws were used to connect the motor and the coupling while I need to ensure that the screws is aligned with the cut surface of the motor shaft Finally, M4x6 screws and M4 washers were utilized to connect the wheels and couplings, and the frame of a car with four-wheel drive has been installed.



Title	OpenMV PCB Design and Soldering		
Group	Image Group	**Week**	Week 1+2
Recorder	Ziyi Xie	**Date**	2021.3.19

After deciding on using OpenMV as our microprocessor, we discussed the timetable of the designing and testing of OpenMV.

I searched the Internet for the layout of OpenMV (figure 1) and mastered the skill of drawing PCB. Then I modified the circuit and added another path for supplying power to improve its stability. (figure 2) What is more, our team logo was designed by me and painted on the PCB which could be seen in figure 3 and 4. After doing some validations and verifications, these PCB designs were sent to the PCB printer factory.

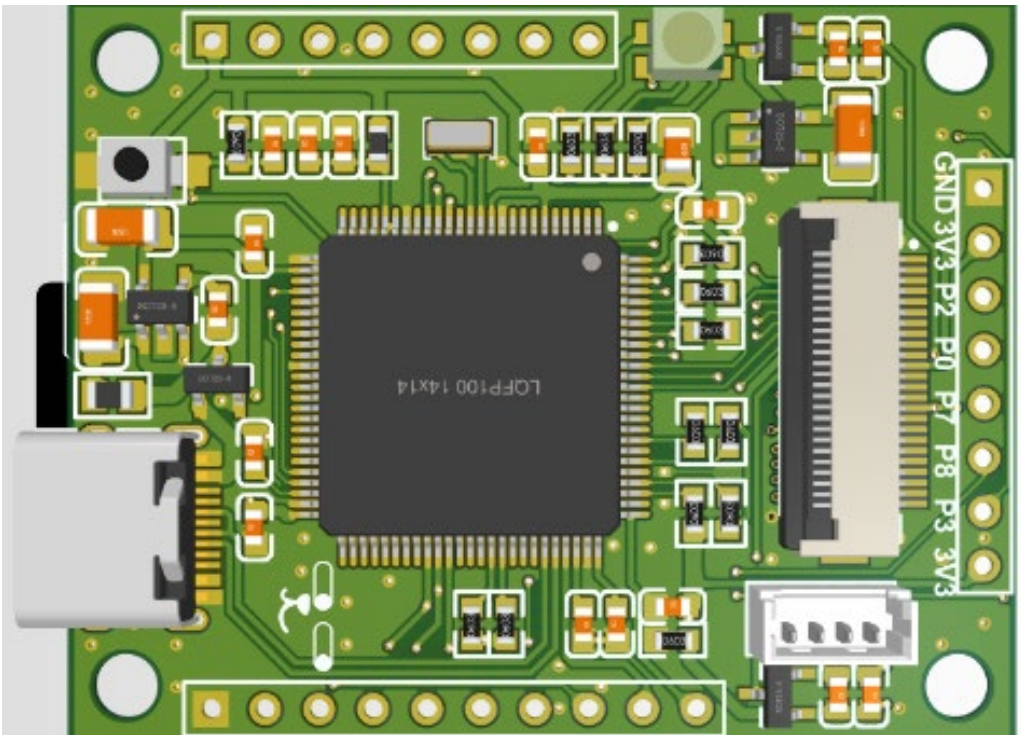


Figure 1 Layout of OpenMV

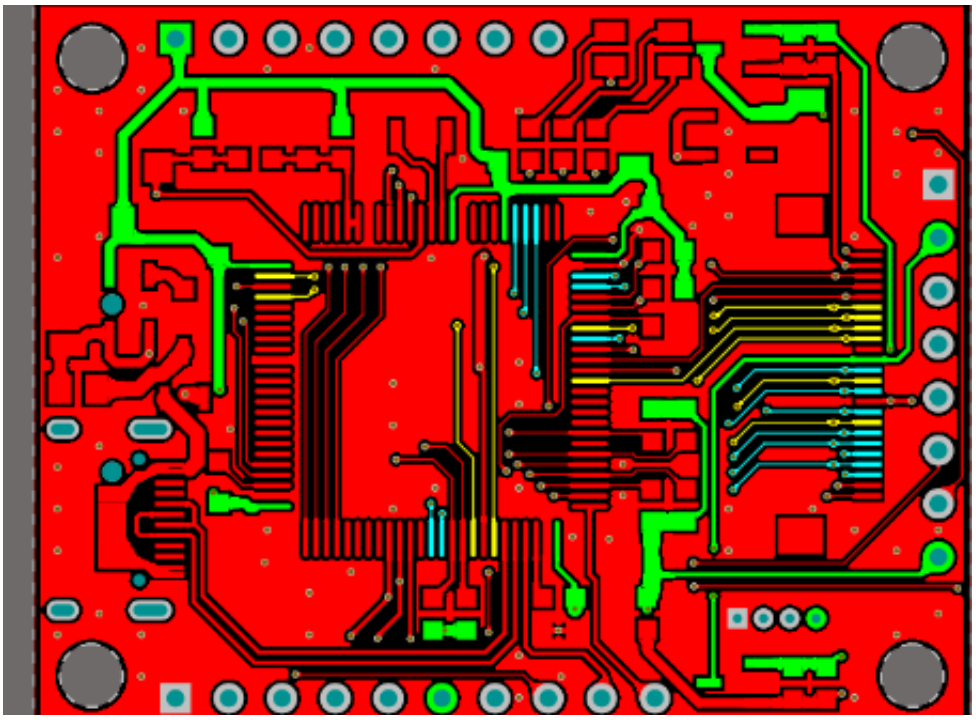


Figure 2 PCB design of OpenMV

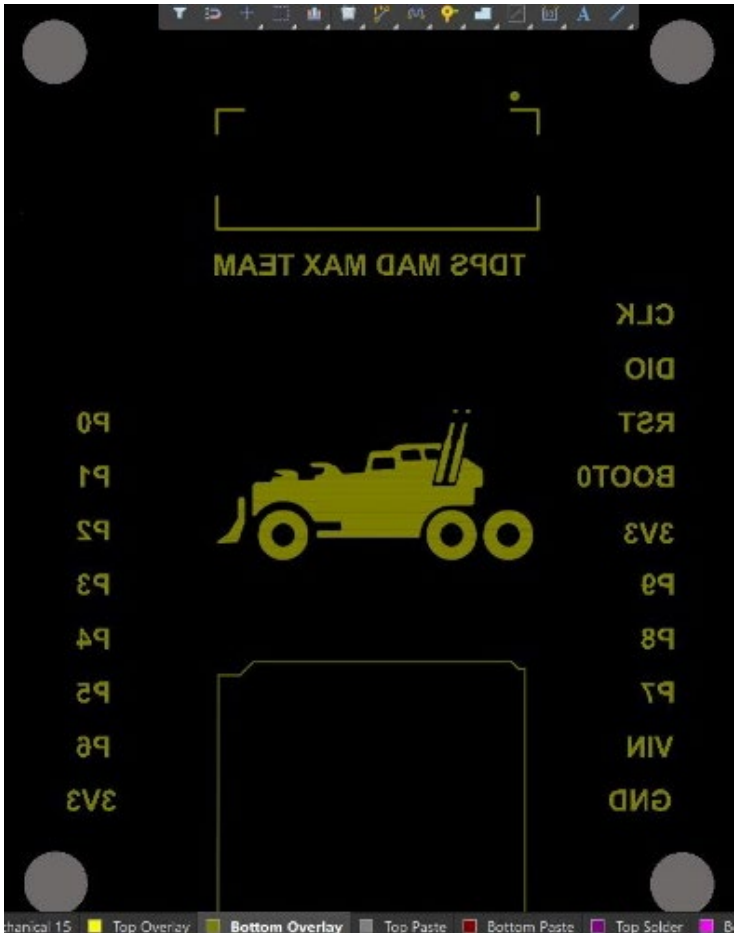
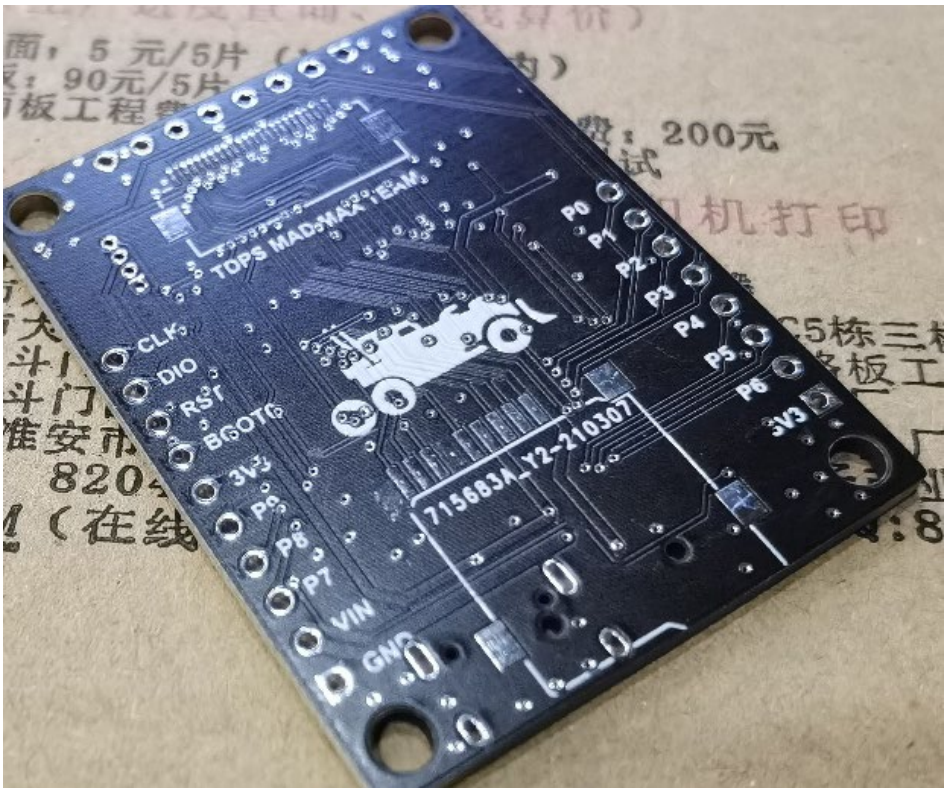


Figure 1 Logo

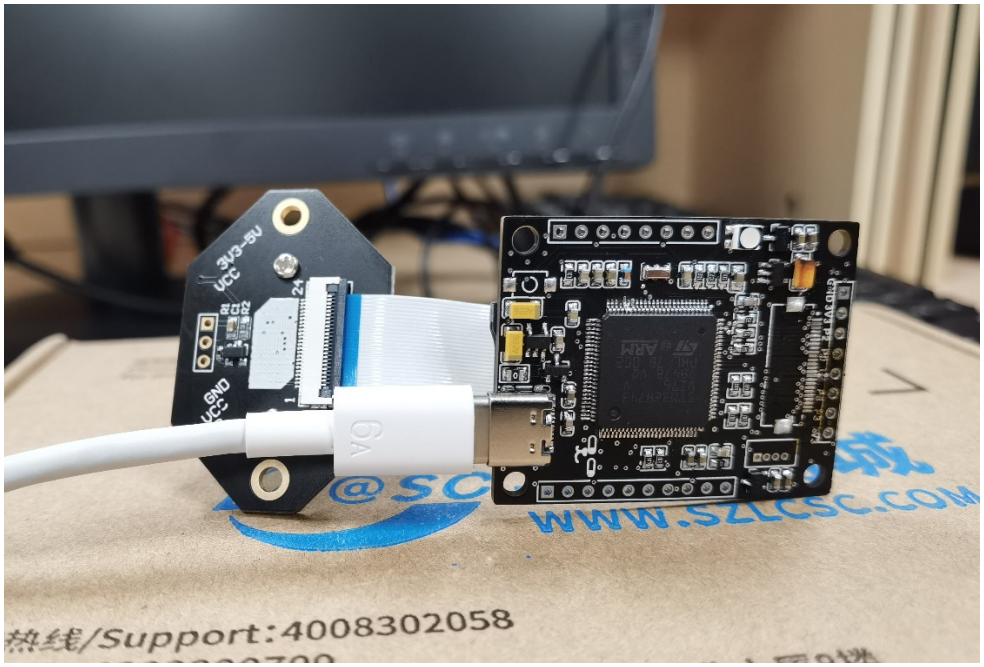
Having received the printed PCB of OpenMV, I started to solder all the components and did some testing like measuring some resistance and voltages between two pins. But unfortunately, the ground level voltage was 5V which meant our circuit had some problems. I didn't figure out the exact problem of PCB since all resistors and capacitors are in the right position and functions well. But I thought there might be some imperfections in the processing of soldering.

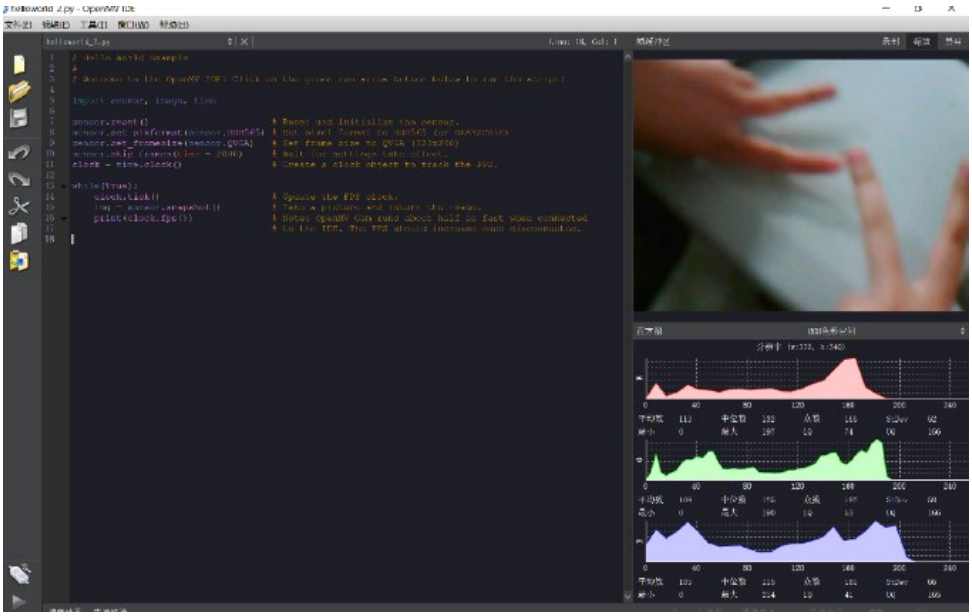


In order to follow the Gantt chart and offer hardware of OpenMV for image processing, I decided to start two plans at the same time. Plan A was still designing and soldering another OpenMV. Plan B was to buy a cheap demo board which uses the same chip as OpenMV and then transformed this demo board into OpenMV by bootloader.

The result was shown below: Plan A the LED of OpenMV did not flash when connected to the power source.

Plan B succussed. When I connected our microcontroller to the IDE, it could be identified. It also run he demo program successfully.


3V3-5V
VCC
GND
VCC
WWW.SZLCSC.COM
热线/Support:4008302058



Demo Code

```
# HelloWorld
import sensor, image, time
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(time = 2000)
clock = time.clock()

while(True):
    clock.tick()
    img = sensor.snapshot()
    print(clock.fps())
```

Title	Brain Storm of Image Processing		
Group	Image group	**Week**	Week 3
Recorder	Jinluan Yang	**Date**	2021.3.20

Question 1: The position of the camera?
Plan A: the direction of camera is down.
By this method, it is easy to obverse the whole path which is benefit to control the direction of the car, however the car has to wait for the graph upload and the OpenMV send the instruction to the microprocess. As a result, the car cannot operate smoothly. What is more, there is the limitation about the colour recognition in the patio 2.
Plan B: the direction of camera is forward.
If the camera is facing forward, more information could be involved in one graph because of the longer vision of the camera. The most important part of this plan is prediction, we should predict a long distance route for the car. Thus, when the car runs based on the past instruction, the OpenMV could receive and process the new image and send the command.

Question 2: the position of the beacon.
Beacon could be used in the special position, for example, the bridge and the gate. (Patio 1) When the car receive the signal from the beacon, it means the OpenMV should give a feedback to interrupt to the tracing block. For instance, the OpenMV find the beacon which means the car should across the bridge, and then it will send a signal about turning left.

Question 3: the type of beacon.
Plan A: the colour beacon.
This kind of beacon can be recognized and distinguished easily, the programme achieves conveniently as well. However the range of this signal is limited, and it cannot transmit the signal active.
Plan B: the ultrasonic beacon.
The ultrasonic could be recognize in a long distance and it is easy to distinguish by different frequency while this function require the other block to achieve so this plan has low execution.

Week 2 – Group Seminar 2

Title	Group Seminar 2: Progress Report		
Group	All	**Week**	Week 3
Recorder	Zhangchen Xu	**Date**	2021.3.21

On March 21, 2021, the group held a 30-minute group seminar (No.2) at the Tencent conference. In this online meeting, the image group showed how to use OpenMV and performed a demo. The car group also showed their built cars through photos.

Next week's tasks are as follows: The car group will solder the designed L298N module and conduct wiring experiments. The image group will start the tracing and colour recognition research. Hope there will be good results~

Title	How to Connect Modules Together		
Group	Car Group	**Week**	Week 4
Name	Xinyue Li	**Date**	2021.3.27

The object of this lab is to connect modules with wires so that each module can operate smoothly. Initially, we refer to the graph of the micro control unit in order to understand the function of each pin. Then we look through the user manual of each module to comprehend how to connect modules in a correct way.

There are mainly six modules, which are Ultrasonic distance measuring module HC-SR04, Driver module, Motor module, Motion Processor and Power module.

We adopt 4 HC-SR04 modules in our design. Firstly, we have to understand the basic working principles of these modules.

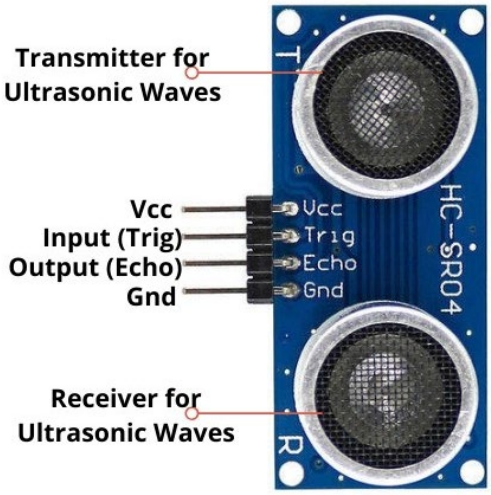


Figure 1

There are 4 pins, including Vcc, Trig, Echo and GND. It is obvious that GND connects to the GND pin of the MCU and Vcc connects to “5V” supplied by the power module (the Ultrasonic distance measuring module generates by 5V). As for the Trig pin, it is used to input trigger signal. The function of Echo pin is to transmit signal echo. The figure 2 below shows how to connect the Ultrasonic distance measuring modules to the MCU.

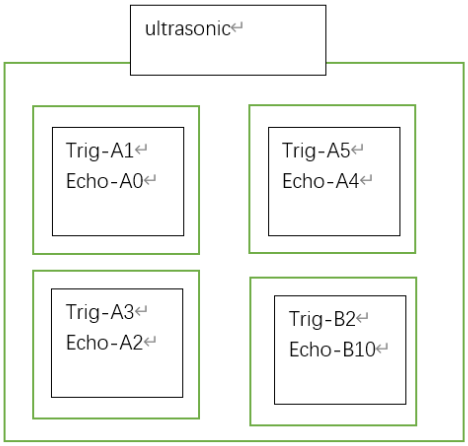


Figure 2

As for the Driver module, we combined 2 basic driver circuit in one driver module to control 4 motors. The Figure 3 and Figure 4 show the structure of the driver module.

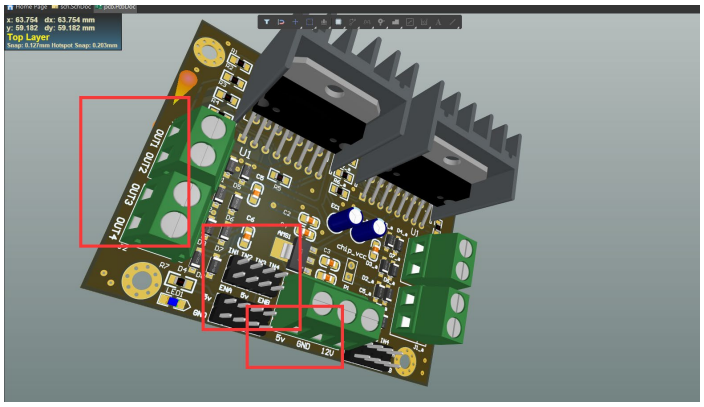


Figure 3

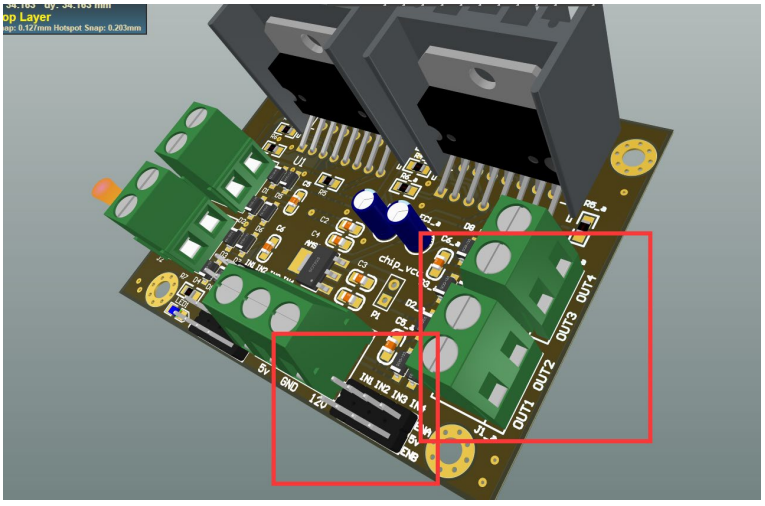


Figure 4

The Vss is connected to the voltage supply Vcc 12 volts and the Vs is connected to the digital signal 5 volts. There are 2 groups of OU pins, OU1 to OU4 are used to input digital signal series which control the direction of current output to the two

motors (M1 M2), the other group is used to control the other motors (M3, M4). Besides, the ENA and ENB pins are used to control the speed of wheels by adjusting the PWM of input. The Figure 5 illustrates how the driver module is connected to other parts.

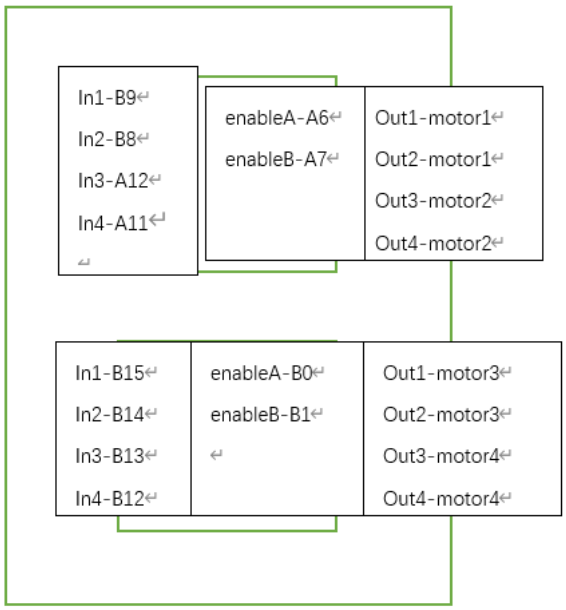


Figure 5

Next, we considered linking MPU-9250 to the design.

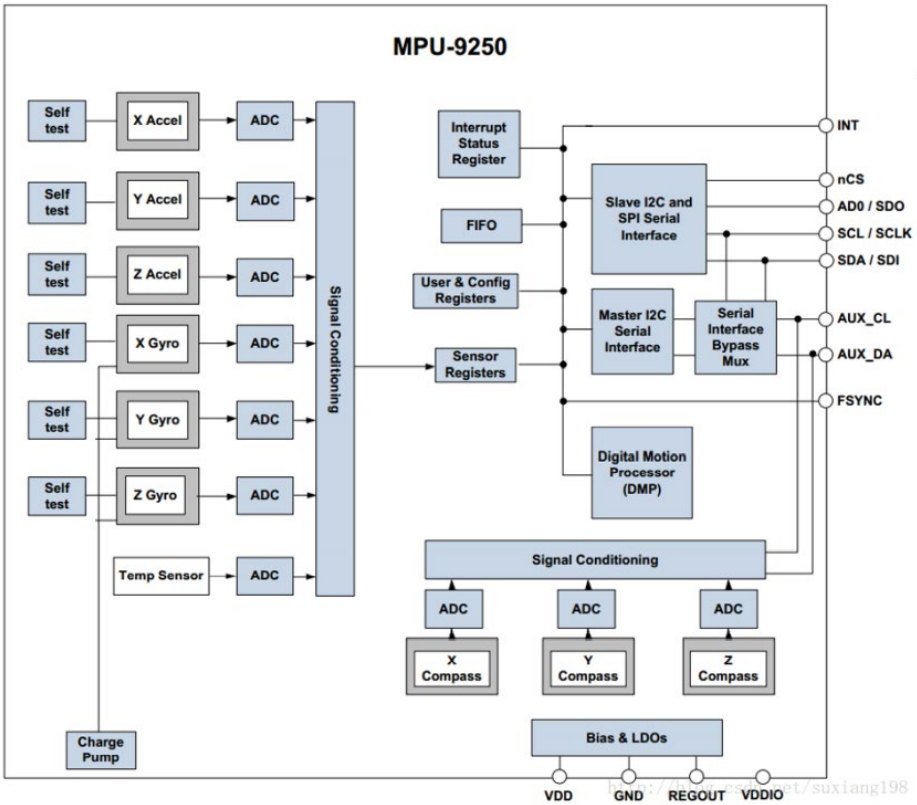


Figure 6 (the structure of MPU9250)

We connected the motion processing module as Figure 7 shown. However, we have difficulty finding out how to connect pin AD0 to the MCU.

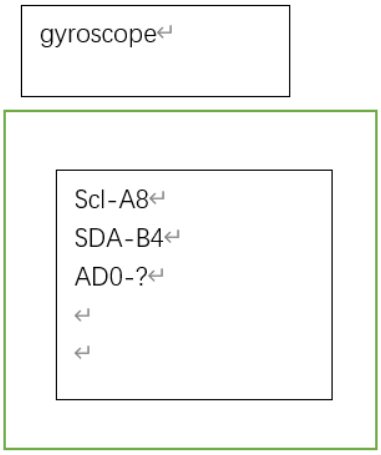


Figure 7

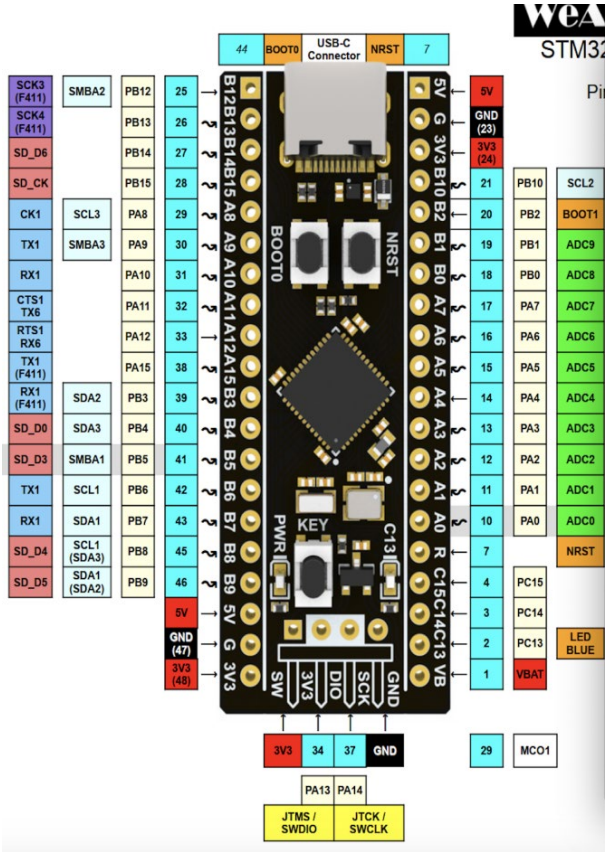
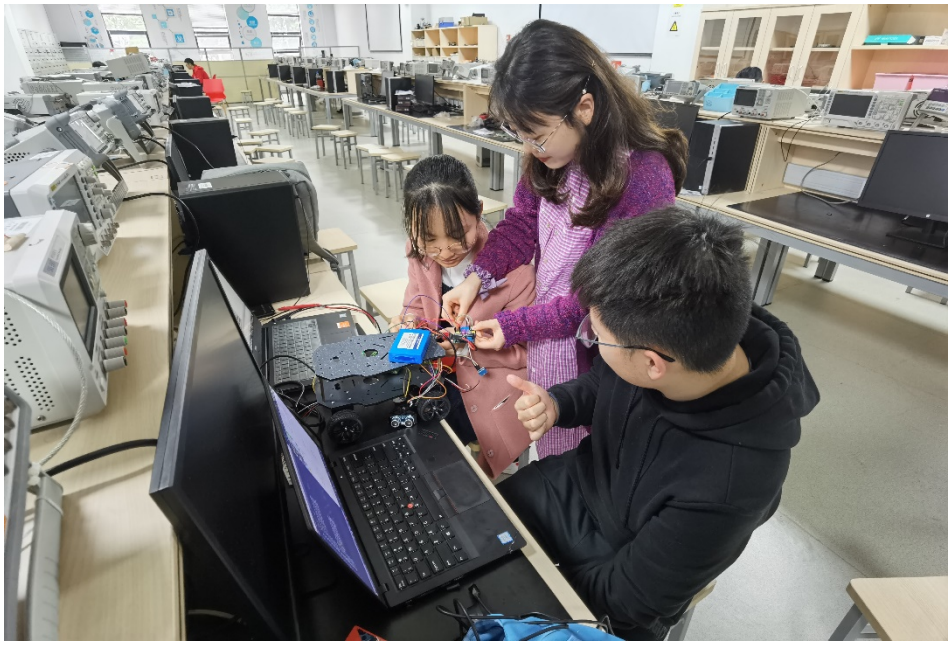
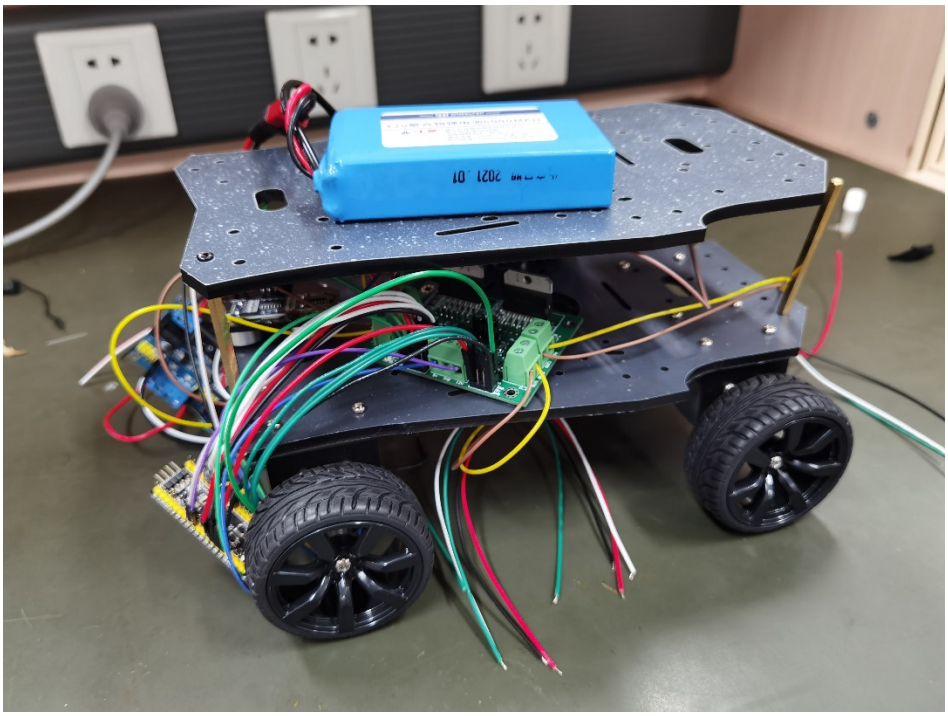


Figure 8(the I/O of MCU)

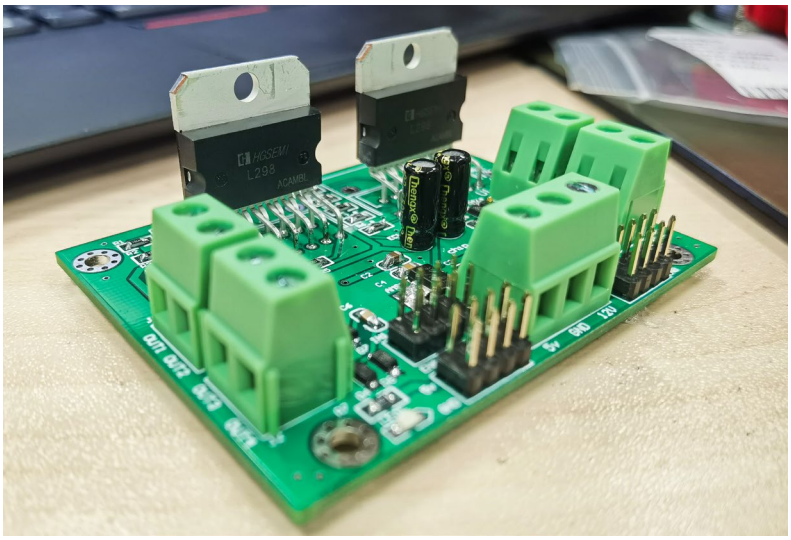
Photos:

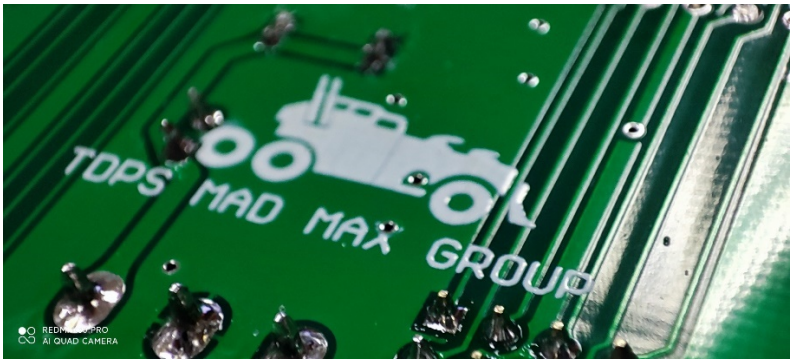




Title	How to Damage Two Pieces of STM32 Board		
Group	Car Group	**Week**	Week 4
Recorder	Ziyang Long	**Date**	2021.3.27

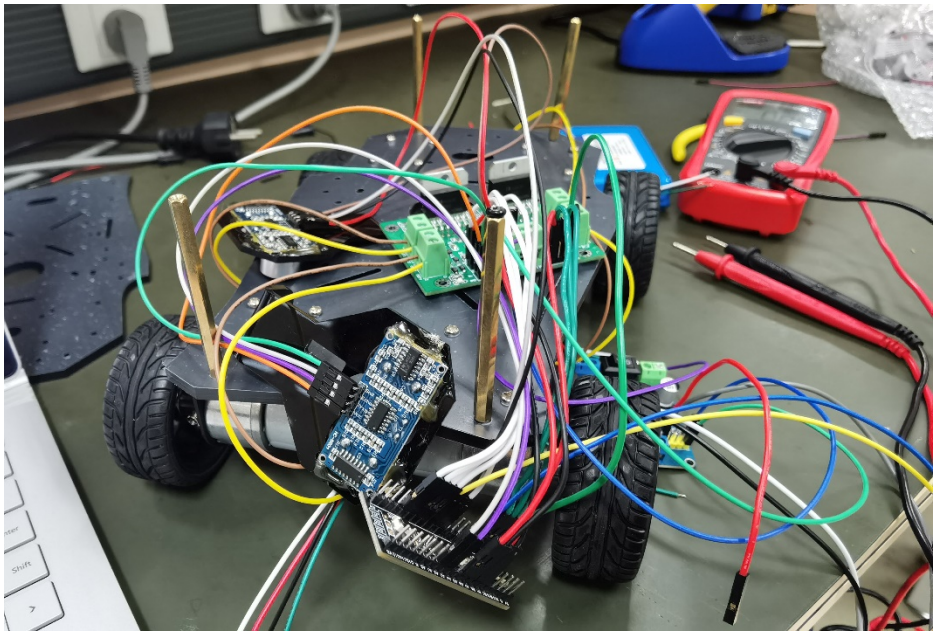
We found a L298N PCB design on the internet. And Just having a sketchy glance of its design, we decide to use that version, which finally cause a big problem.





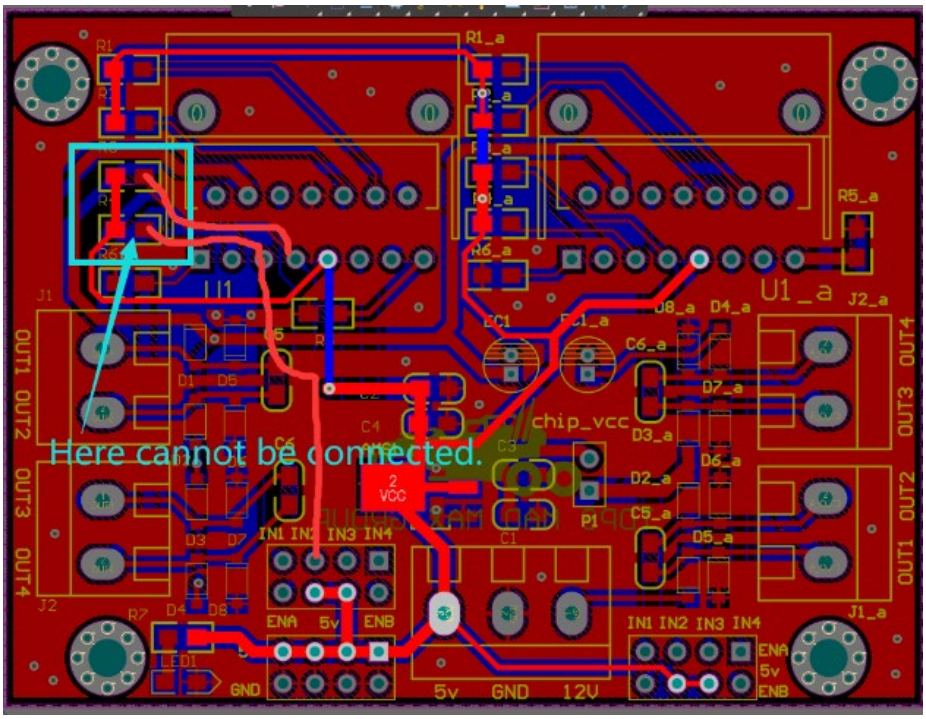
After the soldering of L298N board, we checked its power supply and it was ok at that moment, enhancing my belief of the correctness of that board. And we decide to assemble all components together. At first, everything is fine, including the connection between each module, the correct output of STM32 board, the smooth run of battery and power control module. However, we found the motor did not rotate as we expected. As a result, we come back to check the whole circuit. Suddenly, the power LED of STM32 board die out which deepen our doubts.

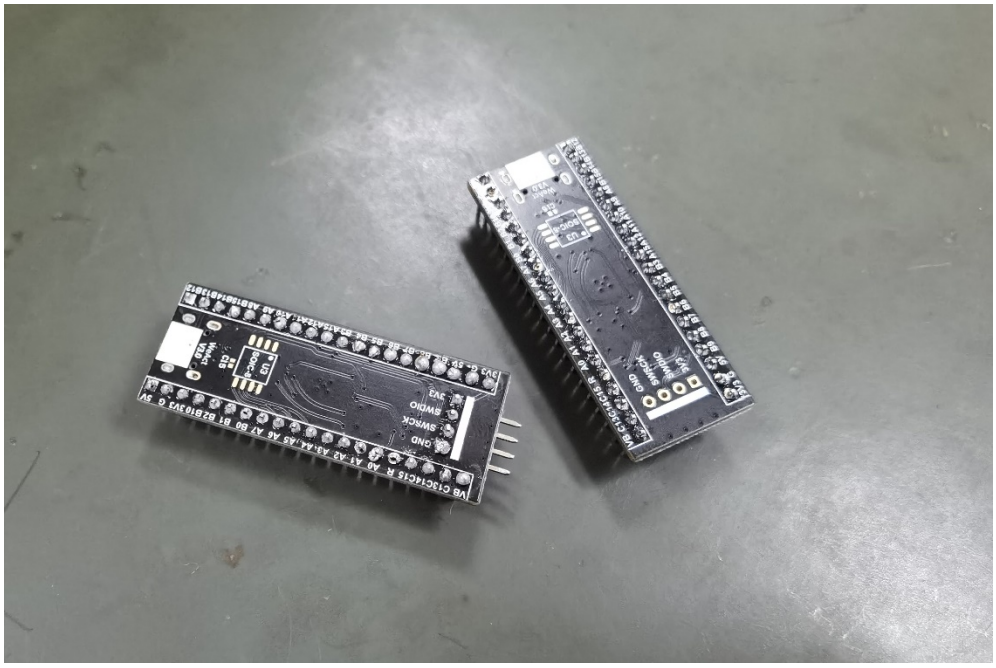
We firstly check whether STM32 board was still ok, and we found we cannot read the data from STM32 board when we use type C line to connect it with laptop. It makes us realize the STM32 board's damage and we just could not find the reason.



Someone advised that we should try another STM32 board, and we do as her saying. Although the power LED initially work fine, it still extinguished after a few minutes. Every car group member was sorrow about what we confronted. Finally, after checking other part of module, we focus on the problem in L298N.

I use Altium designer to scrutinize every details of that PCB design and found that the input pins were wrongly connect to the 5V power supply line, which probably cause the reverse breakdown of STM32 board. And it is too obscure to notice.





Two Dead STM32 Boards

Week 3 – Group Seminar 3

Title	Group Seminar 3		
Group	All	**Week**	Week 4
Recorder	Zhangchen Xu	**Date**	2021.3.28

On March 28th, 2021, the mad max group met at A1-305 in the main building for the third offline seminar.

In this seminar, the car group reflected on the mistake of burning two STM32s and received encouragement from everyone. The image group share their preliminary breakthrough in tracking: the ability to recognize one of the edges of the road.

After this setback, in order not to delay the progress, the car team decided to purchase the finished L298N module first, and then replace the components with the homemade motor drive after passing the debugging. The image group decided to continue on tracing on both sides of the road.

It is worth noting that although Xinyue Li broke her foot this week, she still came all the way to A1-305 in the main building to meet with the group. Great!

Title	Multiple Tracing Approach Comparison using OpenMV		
Group	Image Group	**Week**	Week 5
Recorder	Ziyi Xie; Shubo Yang	**Date**	2021.3.30

Tracing approach 1: Binarization + Focus

Purpose:

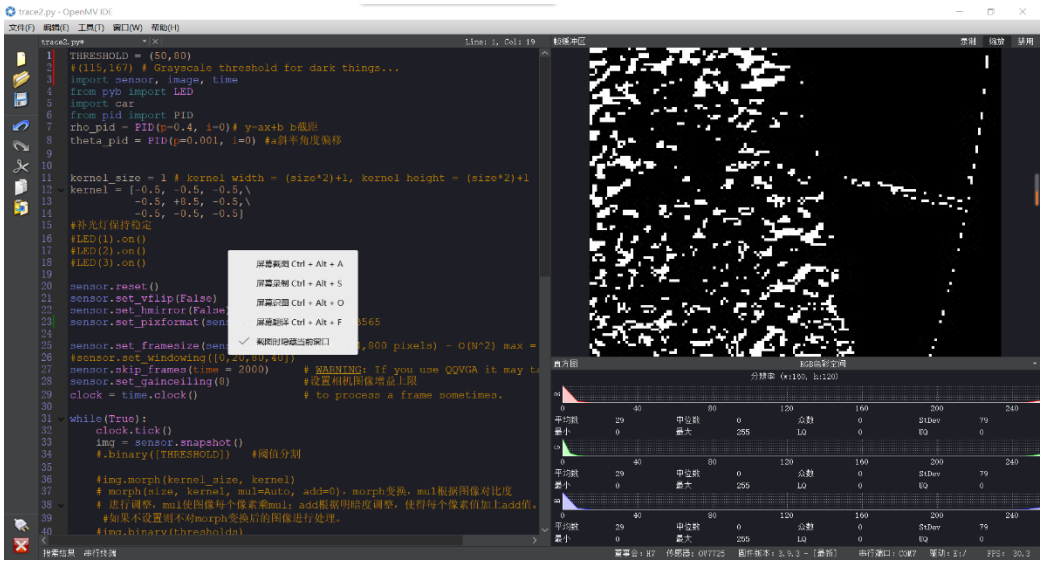
We try to combine binarization and focus to identify the trace of the path and give instructions for the car, e.g., turning left or right. By calculating the focuses of the binarized image, we can derive the focus of the black pixels and the white pixels. By comparing the two focuses, the instructions that our car should take can be consequently determined.

Procedure:

On the website of OpenMV codes, we searched for functions to binarize the images. “img.binary()” can binarize the images with its threshold carefully set. We write a python program including the “img.binary()” and test it with images. The focuses are then calculated.

Result:

Using “img.binary()”, the images are binarized according to different thresholds. We adjust the thresholds and choose the best one that has the clearest edges and traces. The focuses are then calculated. However, because the rough pebbles on the road, the black and white pixels are crossed and the edges are hard to distinguish from the binarized images. Additionally, we find that the performance is affected by the light and shades, so this solution is not robust against environmental changes.



Conclusion:
The performance of binarizing and finding the focuses of the black pixels and the white pixels is affected by several factors, including weather, roughness of the road, lighting conditions, and shadows of environmental objects. Thus, this approach is not the best option.

Tracing approach 2: Binaryzation + Regression
Purpose:
We try to combine binarization and regression to identify the trace of the path and give instructions for the car, e.g., turning left or right. By calculating the regression line of the binarized image, we can derive the trace that our car should follow, the instructions that our car should take can be consequently decided.

Procedure:
On the website of Openmv codes, we searched for functions to binarize the images. "img.binary()" can binarize the images with its threshold carefully set. We write a python program including the "img.binary()" and test it with images. The program also contains a function "img.draw_line()" that returns the parameters of the line regression of the binarized image.

Result:
Using "img.binary()", the images are binarized according to different thresholds. We adjust the thresholds and choose the best one that has the clearest edges and traces. The regression line is then determined by the "img.draw_line()". Using the parameters of the line, we can decide the instructions for the car. However, as in the approach of combining binarizing and focuses, the pixels in the binarized image are crossed, and the edge between the black and white pixels is not clear.

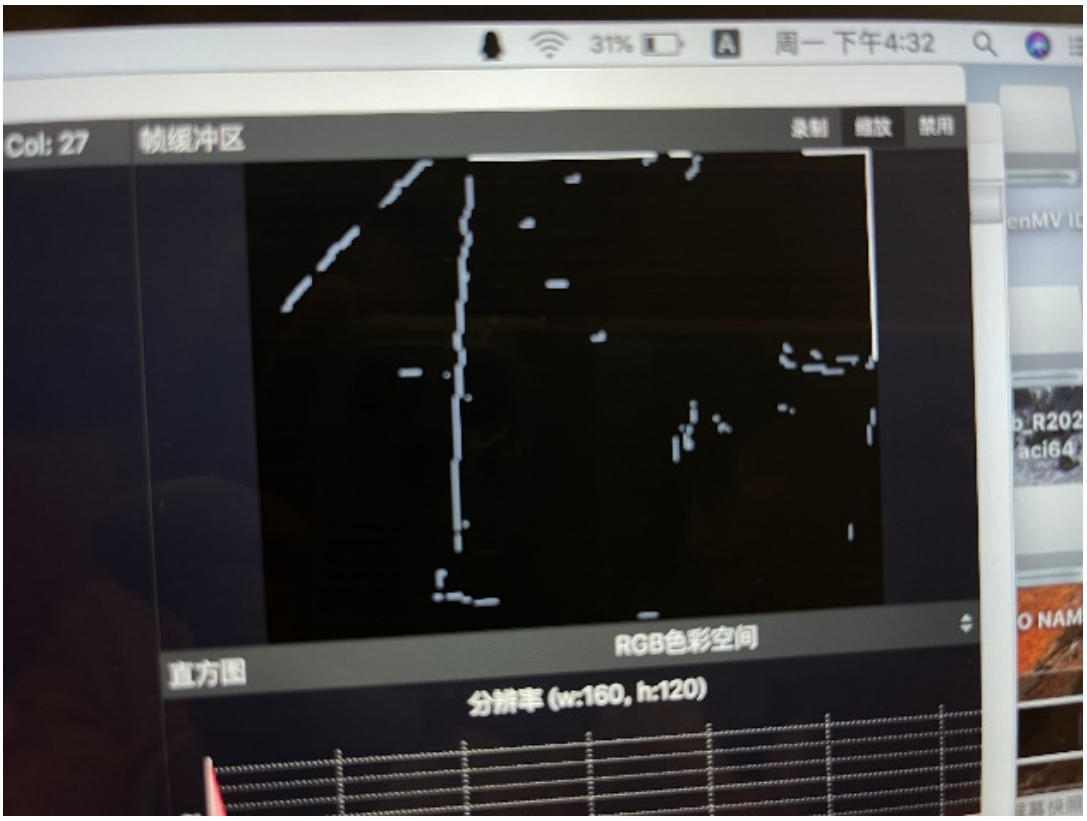
Conclusion:
The performance of binarizing and finding the regression line is not satisfactory due to the similar reason of the binarizing and finding focuses. This is because the binarized images have many disadvantages which make it difficult to detect the trace.

Tracing approach 3: Edge Detecting
Purpose:
We try to utilize edge detecting to identify the trace of the path and give instructions for the car, e.g., turning left or right. Once the edges between other obstacles and the main path are identified, the trace that our car should follow can be immediately determined.

Procedure:
On the website of Openmv codes, we searched for some functions to detect edges. There is a function "img.find_edges()" that can detect edges in a fast speed, with the Canny operator used. We write a python program including the "img.find_edges()"

and test it with normal edges such as the classroom, gaps in the keyboard, and in the outdoor environment of the car. The edges are detected successfully, and the threshold is adjusted to get a better detection.

Result:
We find that the thresholds of the classroom, keyboard and the outdoor path are different. We set several sets of thresholds and also modify the distance between the ground and our camera to find the optimal angle. After a few trails, the edges of the path can be recognized individually.



Conclusion:
The performance of using Canny operator is sufficient to detect edges. We modify the thresholds to get an optimal performance with edges clearly detected.

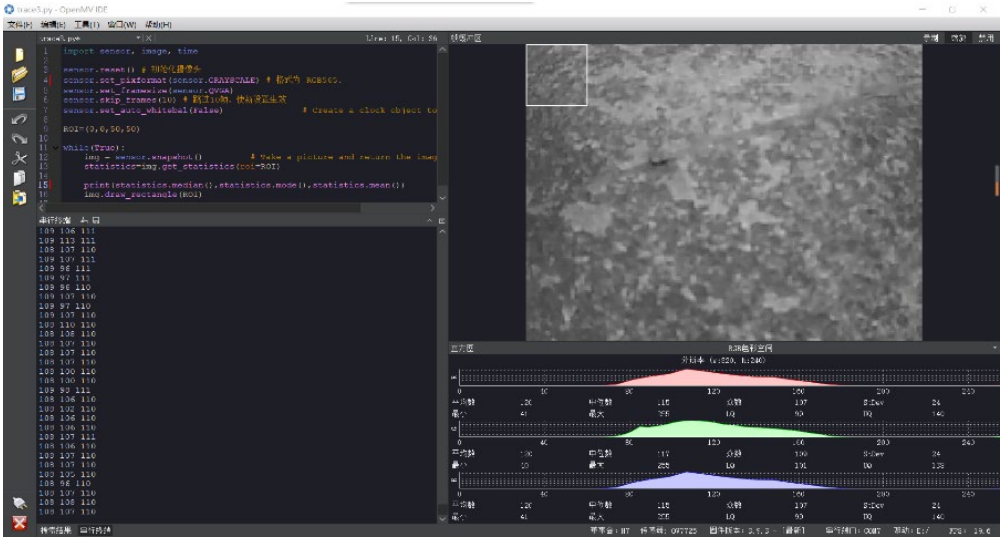
Tracing approach 4: RGB channels
Purpose:
We try to use the RGB channels (red, green, and blue) to identify the trace of the path and give instructions for the car, e.g., turning left or right. The three channels have different parameters, which can be used to determine the trace and the instructions that our car should take.

Procedure:
We try to compare the parameters and adjust the thresholds of the three channels, i.e. red channel, green channel, and blue channel. The three thresholds of the channels

can filter several pixels, and then determine the clear trace of our car. Additionally, the subtraction or addition of the channels are attempted.

Result:

We use the adjustment bar of Openmv to adjust the thresholds to see if certain pixels can be filtered and the particular path can be remained. During the process, we sadly find that the parameters of the RGB channels on our tracing path and on other sideroads are similar, which means that RGB channels cannot provide much information for us to identify the trace. The result of our threshold modifications verifies our instinct finding. Though we modify the thresholds several times, the filtered image is not satisfactory and the trace in the image is not clear.



Conclusion:

The performance of using RGB channels is not ideal since the parameters of the three channels on the path and off the path have no significant differences. Therefore, the information offered in the RGB channels is not more compared to other grayscale images.

Title	Deep Research into Edge Detection (Tracing) by Canny and Filter		
Group	Image Group	**Week**	Week 5
Recorder	Jingluan Yang Qinlin Han	**Date**	2021.4.2

Objective:
The aim of this experiment is to find the edge of the path, which is the necessary part of the tracing, which is the main task in the Patio 1.

Plan:
We search and study the theory of needed operations. After analyzing each operation of processing images, we have tried each image processing operation and chosen the most suitable operation method at present.

The initial outline for tracing:
1. Receive the graph
2. Graying of image: reduce the unnecessary calculation and increase the speed of the process.
3. Find edge: Canny edge detector, the difference between the path and the environment is small, so we should use the algorithm with high accuracy. (this method didn't find clear edge in the lab in week 4 and 5, so we should process the image by the filter)
4. Find the middle line and deviation value: according to the edge of the path, accumulative the bias between the center of the car and the middle line of the path, which will determine the direction of the car.
5. Send the instruction and receive the new image.

Procedure:
1. The principle of two basic operation:

a) Image graying:

Gray scale image refers to an image with only brightness information and no color information, in order to represent the gray-scale image, it is divided into 256 levels from 0 to 255, of with 0 is the darkest and 255 is the brightest.

b) Image binarization:
It is a processing of converting a pixel image to a binary image. The gray value of the pixels on the image is set to 0 or 255. All the pixels whose gray value is greater than or equal to the threshold value are determined to belong to a specific object, and their gray value is represented by 255. Otherwise, these pixels are excluded from the object area, and the gray value is 0, indicating the background or the exception of the object area.

2. The principle of Canny Edge Detection Algorithm:

The steps of canny edge detection are preprocessing, calculating gradients, non-maximum suppression and threshold with hysteresis.

For preprocessing: the image is smoothed using Gaussian filter.

For calculating gradients: gradient magnitudes and directions are calculated at every single point in the image by using the finite difference of the first-order partial derivative:

The magnitude of gradient is: $m = \sqrt{{G_x}^2 + {G_y}^2}$

The direction of gradient is: $\theta = \tan^{-1}(\frac{G_y}{G_x})$

For non-maximum suppression: non-maximum suppression of gradient amplitudes simply means suppressed a pixel if the pixel is not a maximum. At each point, the center pixel m of the neighborhood is compared with the two pixels along the gradient line. If the gradient value of m is not greater than that of two adjacent pixels along the gradient line, the value of m is 0: $N[i,j] = NMS(M[i,j], \xi[i,j])$.

For threshold with hysteresis: Double threshold algorithm is used to detect and connect edges.

The histogram of the image is counted to determine the threshold. After finding the starting point of the boundary, all of edge pixels would be found out.

Specifically, pixel detection is carried out in turn to check whether the current pixel is an edge. Then check the two pixels in the direction of the edge, that is, two pixels perpendicular to the gradient direction. If the direction is in the same bin as the central pixel, the gradient magnitude is greater than the lower threshold and they are the maximum compared to their neighbors, then these pixels could be marked as edge pixels.

3. The principle of filters

a) Median filter:

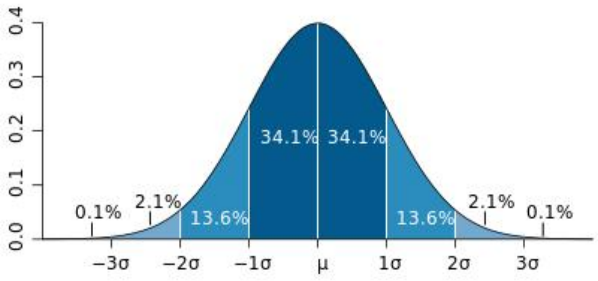
Each pixel in the image and its neighbors would be considered in turn to determine whether it is able to represents its surroundings. All the pixel values of the surrounding neighborhood are sorted in numerical order, and then the middle pixel value is used to replace the considered pixel.

b) Mean filter:

The main method of mean filtering is domain average, whose basic principle is to replace each pixel value in the original image with the average value. For the pixel (x, y) to be processed, a template composed of several neighboring pixels is selected to calculate the mean value of all pixels in the template, and then the mean value is given to the pixel (x, y). The gray level of the processed image at the point: $g(x,y) = \sum f(x,y)/m$, where m is the total number of pixels including the current pixel in the template.

c) Gaussian blur:

Gaussian blur is to use normal distribution to calculate the transformation of each pixel in the image. In two-dimensional space, the normal distribution equation is: $G(u,v) = \frac{1}{2\pi\sigma^2} e^{-(u^2+v^2)/2\sigma^2}$. The convolution matrix composed of non-zero pixels is transformed with the original image. The value of each pixel is the weighted average of the neighboring pixel values. The value of the original pixel has the largest Gaussian distribution value, so that it has the largest weight(the wight of normal distribution).



As the distance between the adjacent pixels and the original pixels becomes longer and longer, the weight becomes smaller and smaller. In this way, the edge effect can be preserved more effectively than other equalization fuzzy filters.

d) Erode and Dilate

The erode and dilate processor have similar function: noise elimination and separating image elements into segments and connecting the adjacent elements in the image. From a mathematical point of view, dilation or erosion is convolution of image and element. Dilation is the operation of seeking the local maximum value. Convolution with element is the maximum value of the pixels in the area covered by this element, and the maximum value is assigned to the pixels specified by the reference point, so as to increase the highlight area. Erosion is the opposite operation, it could acquire the local minimum value. When the results of the template and the corresponding elements in the input image are both not 0, the result is 0. This operation will delete some pixels of the object boundary and reduce the highlight area.

4. Results

```
THRESHOLD = (120, 167)
#(115, 167) # Grayscale threshold for dark things...
import sensor, image, time
from pyb import LED
import car

sensor.reset()
sensor.set_vflip(False)
sensor.set_hmirror(False)
sensor.set_pixformat(sensor.GRAYSCALE)   # RGB565
```

```
#sensor.set_pixformat(sensor.RGB565)

sensor.set_framesize(sensor.QQVGA) # 80x60 (4,800 pixels) - O(N^2)
max = 2,3040,000.
#sensor.set_windowing([0,20,80,40])
sensor.skip_frames(time = 2000)     # WARNING: If you use QQVGA it
may take seconds
sensor.set_gainceiling(8)
clock = time.clock()                # to process a frame sometimes.

while(True):
    clock.tick()
    img = sensor.snapshot()
    img.mean(1)       # mean filter
    img.gaussian(1)  # gaussian blur
    img=img.erode(1,threshold=8)     # erode for 8 pixels
   img.find_edges(image.EDGE_CANNY, threshold=(100, 200)) # Canny
   edge detection
```

Conclusion:

In this lab, we study the principle about a lot of filters and operator. After several tests and experiments, we could get a clear edge of the path by combination the different filters by a specific order and the result and code have been showed in the above part, while some problems and drawbacks still be pointed. The first one is that the graph cannot be shown stability so the future process will be influenced, another problem refers to the programme could just one side of the path, which will reduce the accuracy of the middle line. In the next week, we might try a different method to find edge based on the Canny edge detection also. In this lab, we try to use the filter to avoid the edge in the path extract the edge while we might try to amplify the edge in the path and then determine the path by the amount of the edge.

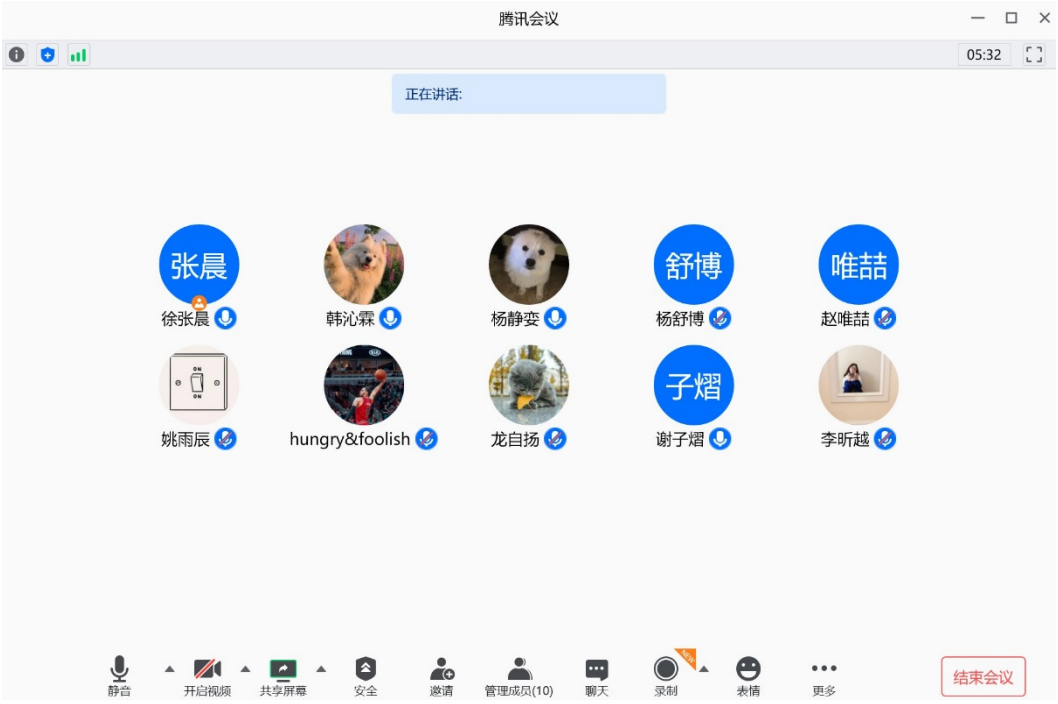
Week 4 – Group Seminar 4

Title	Group Seminar 4		
Group	All	**Week**	Week 5
Recorder	Zhangchen Xu	**Date**	2021.4.4

On April 4, the Mad Max group held a 30-minute group seminar at the Tencent conference platform. In this seminar, members of the image group shared their progress in tracing roadsides using different algorithms, which was well appreciated by everyone.

In addition, the meeting also discussed matters related to the mid-term report and lab book and decided to start writing the initial report in the following week.

After discussion, the tasks for the next week are as follows: the cart group uses the finished L298N to make the vehicle move and turn smoothly; the image group continues to optimize the tracing algorithm to make it have better performance.



Title	The Simulation for Colour-recognition		
Group	Image Group	**Week**	Week 6
Recorder	Jingluan Yang; Qinlin Han	**Date**	2021.4.6

Objective:

In order to complete the task 1of Patio 2, the camera should identify the colour correctly.

Plan:

Firstly, we code the program and test it by the simulation colour block. After that, we test the program in the actual environment to make sure the car could determine the correct direction. In the last step, OpenMV sends the information about angle and distance to the microprocessor.

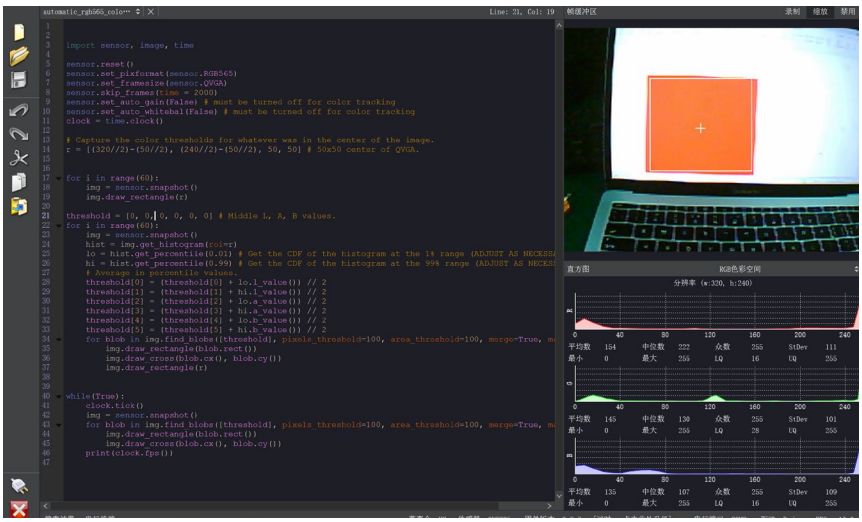
Procedure:

1. The outline of the program:
   OpenMV could capture the color thresholds for whatever was in the center of the image by the function img.get_histogram( ) which acquires the colour histogram of the image with Lab mode. Next, the function histogram.get_percentile-(percentile) could calculate and select the cumulative distribution function of the colour histogram in a certain range. And then, the range of colour threshold could be calculated by the simple equation. Finally, through the function img.find_blobs([threshold]) could find the colour block which has the similar colour threshold. And the code is listed:

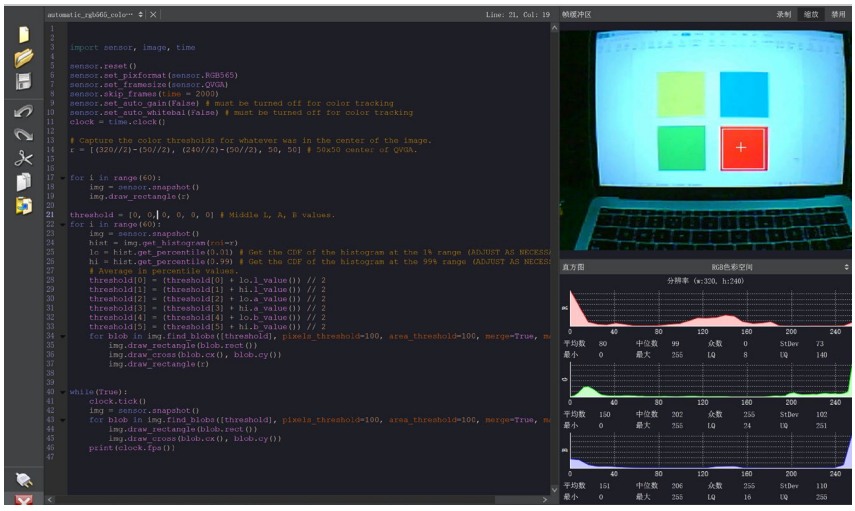
```
import sensor, image, time
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(time = 2000)
sensor.set_auto_gain(False) # must be turned off for color tracking
sensor.set_auto_whitebal(False) # must be turned off for color tracking
clock = time.clock()

# Capture the color thresholds for whatever was in the center of the image.
r = [(320//2)-(50//2), (240//2)-(50//2), 50, 50] # 50x50 center of QVGA.

for i in range(60):
    img = sensor.snapshot()
    img.draw_rectangle(r)

threshold = [0, 0, 0, 0, 0, 0] # Middle L, A, B values.
for i in range(60):
```

```
    img = sensor.snapshot()
    hist = img.get_histogram(roi=r)
    lo = hist.get_percentile(0.3) # Get the CDF of the histogram at the 1% range (ADJUST AS NECESSARY)!
    hi = hist.get_percentile(0.99) # Get the CDF of the histogram at the 99% range (ADJUST AS NECESSARY)!
    # Average in percentile values.
    threshold[0] = (threshold[0] + lo.l_value()) // 2
    threshold[1] = (threshold[1] + hi.l_value()) // 2
    threshold[2] = (threshold[2] + lo.a_value()) // 2
    threshold[3] = (threshold[3] + hi.a_value()) // 2
    threshold[4] = (threshold[4] + lo.b_value()) // 2
    threshold[5] = (threshold[5] + hi.b_value()) // 2
    for blob in img.find_blobs([threshold], pixels_threshold=100, area_threshold=100, merge=True, margin=10):
        img.draw_rectangle(blob.rect())
        img.draw_cross(blob.cx(), blob.cy())
        img.draw_rectangle(r)

while(True):
    clock.tick()
    img = sensor.snapshot()
    for blob in img.find_blobs([threshold], pixels_threshold=100, area_threshold=100, merge=True, margin=10):
        img.draw_rectangle(blob.rect())
        img.draw_cross(blob.cx(), blob.cy())
    print(clock.fps())
```

2. The test graph



(graph 1)



(graph 2)

These pictures are the results of the simulation. In the graph 1, the OpenMV capture the colour thresholds and then it could tag the similar colour block in the graph 2.

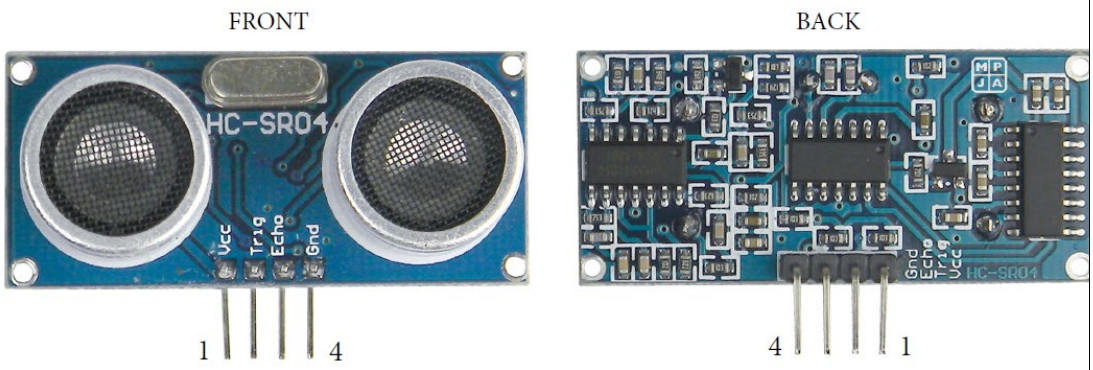
Conclusion:

In this lab, the code for colour-recognition parts has been completed, the OpenMV could collect the threshold in the specific position and mark the same colour block in the graph. The most significant question is how to improve the accuracy of the color-recognize and it has been solved by adjusting the range of the color thresholds. While some problems remain to be studied in the future, such as increasing the range of image acquisition and how to control the direction of the car by the OpenMV directly.

Title	Ultrasonic Distance Calculation		
Group	Image Group	**Week**	Week 7
Name	Shubo Yang	**Date**	4.17

Purpose:
We try to utilize an ultrasonic sensor to detect distance, which to detect distances to objects and give conditions for further instructions. The distances to the obstructions is essential, since they may be used to judge when to turn and when to stop.

Procedure:
The ultrasonic sensor adopted is HC-SR04 module. It has four pins connected with OpenMV: VCC, GND, ECHO, and TRIG. The module is shown below.



Initially, the Trig pin triggers a minimum 10us High level pulse. The sensor transmits a ultrasonic signal and receives the echo. The Echo pin is connected with an input pin on OpenMV, which should be set to wait the High level of the Echo pin. As soon as a high-level voltage is received, a timer in OpenMV should be initialized to count the time. The high-level duration is the ultrasonic propagation time in the air, and corresponds to the distance after several calculations. The following formula could be used to derive the specific distance:

$$Distance = (ECHO\ high\ level\ time \times ultrasonic\ velocity\ /2\ ,$$

Where ultrasonic velocity means the speed of sound in air, and is 340m/s. Thus, after capturing the period of high-level, the time is multiplied by 0.017 in our program.

Additionally, the program is wrapped into a function called "wave_distance_process" in our program, which aims at conveniently calling the function in the future. Consequently, whenever the function is called and the pins for the Echo and Trig are defined in the main program, the function "wave_distance_process" can be adopted to calculate the distance to the objects.

However, the calculation is based on the correct high-level time captured. Thus, how to capture correctly the high-level time remains to be tackled.

Conclusion:

After figuring out the configurations and internal logistics of the ultrasonic sensor, HC-SR04 module, the distance can be calculated by multiplying a coefficient with the high-level time of the Echo pin, after the Trig pin simulates the sensor.

Title	Ultrasonic distance interrupt on OpenMV		
Group	Image Group	**Week**	Week 7
Name	Shubo Yang	**Date**	4.19

Purpose:

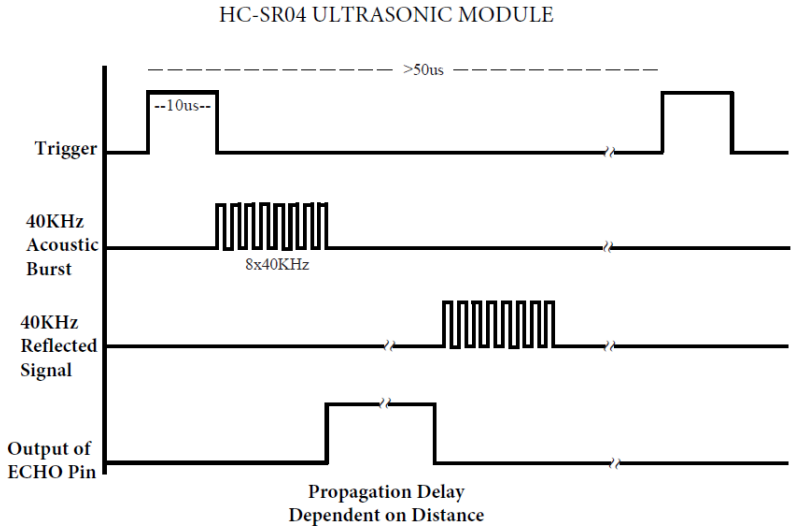
Since the calculation of ultrasonic sensor HC-SR04 module is finished, the high-level time inputted from the Echo pin should be exactly measured. This could be achieved by the interrupt function on OpenMV, but certain functions in program should be coded.

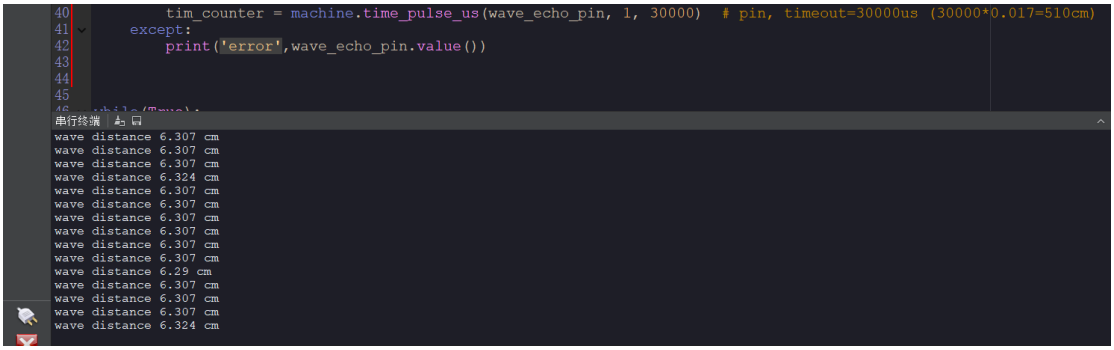
Procedure:

The simulation done by Trig pin outputting a certain pattern is achieved by the “utime.sleep_us()” function, which can be called.

The high-level time is measured using the function “machine.time_pulse_us” in micropython, which is the language used by OpenMV. This function automatically sets a timer and clicks for the high-level time, then it returns the time lasting in us unit. Therefore, the result of this function can be utilized to calculate the wave distance directly.

The result is converted into in unit centimeter, and is shown below. This is a screenshot of OpenMV ide, which proves the successful distance detection.





Conclusion:

The functions in microPython are coded for measuring the high-level time. Finally, the correct distance is calculated and printed out.

Title	Signal processing: screen segmentation		
Group	Image Group	**Week**	Week 8
Name	Shubo Yang	**Date**	4.23

Purpose:
We want to segment the whole picture captured by OpenMV to improve the performance of the signal processing. This is because, better performance of signal processing help to improve the performance of tracing. Signal processing methods may help better visualize the edges of the stones on the road.

Procedure:
The segmentation idea came up because the threshold and binariztion in different parts of the screen is different. Thus, if the screen segmentation is possible, then the thresholds can be set diversely according to the perspective characteristics of each segment. In this way, the processed image may has some performance advancement.

The whole idea is that: first, regions of interest (ROI) are defined, which consist of a grid that segments the screen. Then, for each ROI, specific threshold is adjusted to satisfy the demands. The adjusted performance should improve the edges and make them more visible and clear.

The first step should be defining the grid of ROI. Two matrices “ROIx” and “ROIy” are defined, and they contain the x-axis and y-axis of the corners of the ROI. This is achieved by defining two parameters “xnum” and “ynum”. They represent the number of grids we want to divide on x-axis and y-axis. Some calculations and storages in the matrices are performed. The code is presented as follows.

```
width=320
height=240

# how many ROI regions on x&y-axis
xnum=2
ynum=2
ROInum=xnum*ynum

# calculate the starting points of ROIx,ROIy
xaxis=[0]*(xnum+1)
yaxis=[0]*(ynum+1)
for a in range(xnum+1):
    xaxis[a]=a*width/xnum
    yaxis[a]=a*height/ynum

ROIx=[0]*ROInum
ROIy=[0]*ROInum
for a in range(xnum):
    for b in range(ynum):
        ROIx[b+ynum*a]=xaxis[a]
for a in range(ynum):
    for b in range(xnum):
        ROIy[b + xnum * a] = yaxis[b]
```

Consequently, we can extract the ROI by the corners by indexing “ROIx” and “ROIy”.

The following processing steps should be: after binarization, if there are more white points in each ROI, set all pixels in that ROI to white, and the case is the same for black pixels. This is because there is so much interference in the image, and this step automatically eliminates the interference of the small amout of white points. The part of code presents how to detect and summarize wheter the pixels in that ROI should be set to white or black.

```
# process every ROI region
for i in range(ROInum):
    statistics=img.get_statistics(roi=(ROIx[i],ROIy[i],width/xnum,height/ynum))
    ROI=[ROIx[i],ROIy[i],ROIx[i]+width/xnum,ROIy[i]+height/ynum]

    # just to see the ROI rectangle
    img.draw_rectangle(ROI)
    time.sleep(0.5)

    if statistics.median()==255:  # statistics.mean()
        divide_process(ROI,255)
    else:
        divide_process(ROI,0)
```

Finally, after deciding the black or white, all the pixels should be set. 255 represents white while the number 0 represents black color. Two loops are used to set each pixel in that ROI.

```
# set pixels to 0/255
def divide_process(ROI,value):
    for x in range(ROI[0],ROI[1]):
        for y in range(ROI[2],ROI[3]):
            img=img.set_pixel(x,y,value)
```

Result:

Nevertheless, the segmentation process is too tedious and time-consuming. Additionally, the performance of setting each ROI all into black or white does not significantly improve the image performance and quality. This is not beneficial to further direction control judgement. Thus, this method only improves the processing time.

Conclusion:

The segmentation method cannot be adopted since it takes a large quantity of time to calculate and set the pixels into black or white. Moreover, the image segmentation method does not greatly improve the image processing performance. The further direction judgement cannot rely on a tedious and trivial method. The idea could be changed into setting different threshold according to the road orientation, road conditions, and the weather. This could be adjusted by the time of the tracing, instead of being adjusted by particular segment of the image, since parts of an image mostly resembles. After trailing, the significance of this method is recognized as trivial.

Title	Group Seminar 6 Week 8		
Group	All	**Week**	Week 8
Recorder	Zhangchen Xu	**Date**	2021.4.25

On April 25, the whole group held a 30-minute group seminar at the Tencent conference platform.

In this meeting, members in image group shared their experience in tracing algorithms. They also discussed several potential problems (e.g., the speed of the image processing is too slow) in the meeting. Moreover, the matured ultrasonic sensor reading program are demonstrated by Shubo Yang.

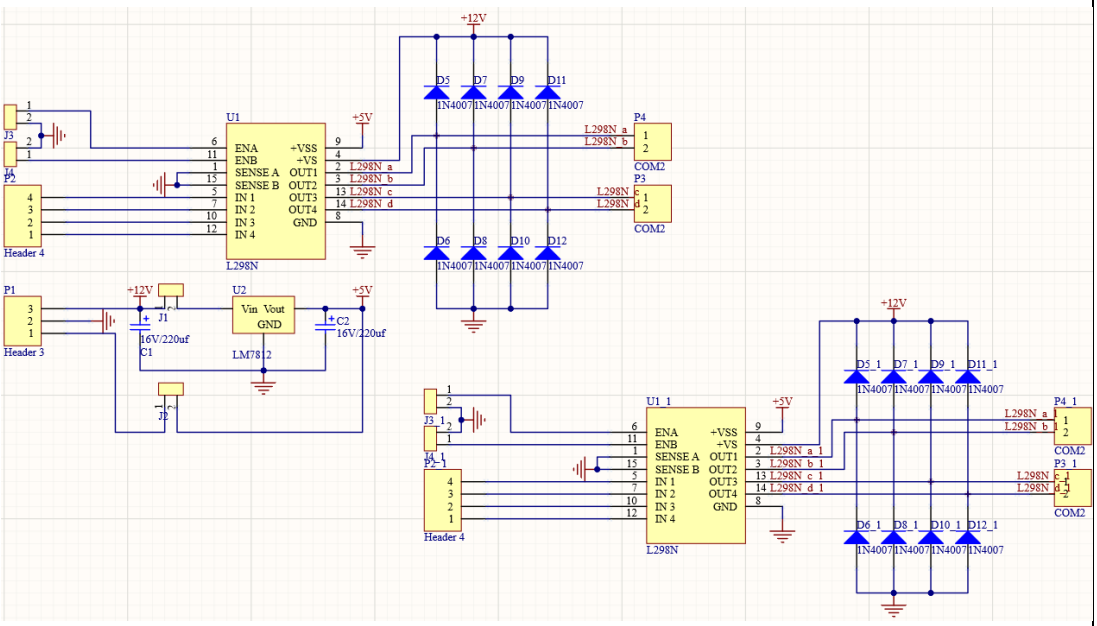
In car group, there is a error in the initial design of L298N driving module, so the new driving module is needed to design. The new design will be available for next week.

Enjoy the holiday!

Title	Two-L298N driver module PCB design		
Group	Car Group	**Week**	Week 9
Recorder	Weizhe Zhao	**Date**	2021.4.30

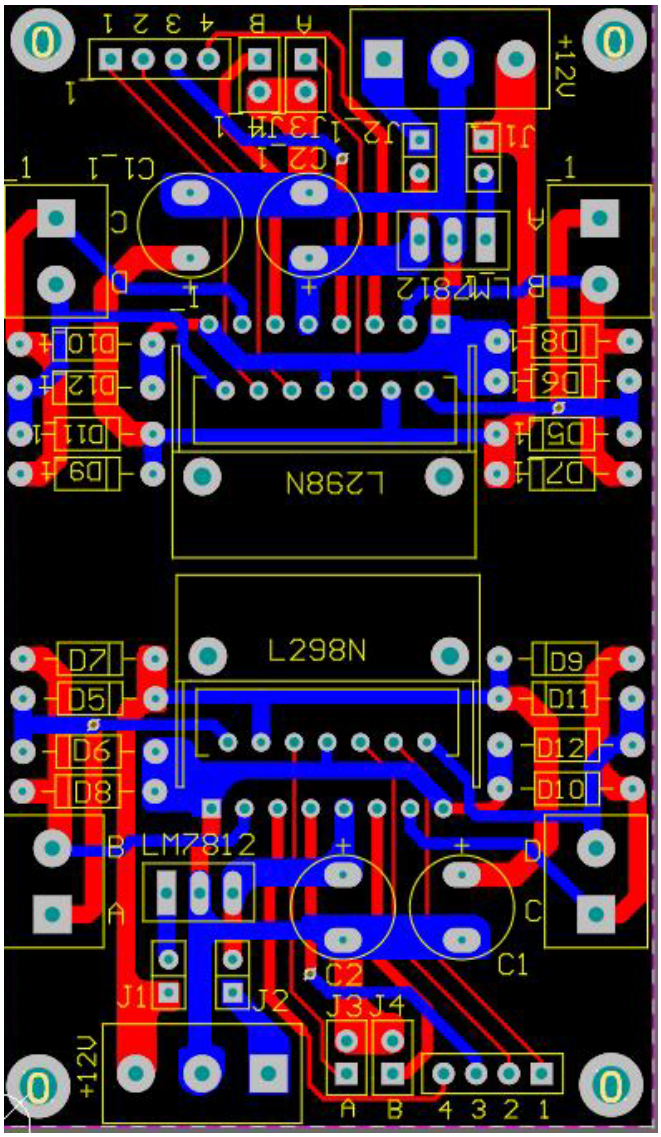
As our smart car is designed to be 4WD, a four-motor driver module is needed.

With reference to the two-motor L298N PCB layout, I designed the schematic plot of the four-motor L298N module as follows, where two L298 chips and one LM7805 voltage-regulator are used, and a bunch of diodes serve to protect the circuit.



Schematic Plot of 4-Motor Driver Module

Using Altium Designer, I designed and drew PCB based on the schematic plot. The package of the devices are downloaded from official website and then connected. After auto-routing, we fine-tune the wires. By making power and ground traces wider than average, we prevent burning wires as such traces have more current flowing through them. Also, 90-degree angles are removed, as the outside corner of that 90-degree angle has the likelihood of being etched narrower than standard trace width. This PCB layout is sent for printing.



PCB layout of 4-Motor Driver Module

Title	HC-12 Wireless Communication		
Group	Image Group	**Week**	Week 14
Recorder	Zhangchen Xu	**Date**	2021.5.4

In patio2 we will use the HC-12 for sending and receiving instructions. The HC-12 wireless serial communication module is a new generation of multi-channel embedded wireless data transmission module. The wireless frequency band is 433.4-473.0 MHz, multiple channels can be set and the communication range can be up to 1000 m in open areas. The communication protocol is UART.

Due to the powerful function of microPython, sending message from the microcontroller (OpenMV) to a computer is relatively easy to do. The only thing we need to do is to first initialize the ports for UART, and then using write command. The code script below illustrates the process.

```
# HC-12 UART Settings
UART_hc12=pyb.UART(1, 9600, timeout_char=1000)
UART_hc12.init(9600, bits=8, parity=None, stop=1, timeout_char=1000)
hc12_info="hello world?"
# Send Message to HC-12
UART_hc12.write(hc12_info)
```

However, receiving message is quite challenging. We use interrupt as our solution. First, we need to define a IRQ in the main function.

```
## HC-12 UART Interrupt
UART_hc12.irq(trigger = pyb.UART.IRQ_RXIDLE, handler = UART_HC12_ISR)
```

Then, we define a handler UART_HC12_ISR and put decoding procedures inside this handler. We also use a global variable to pass the message to the main function.

```
# HC-12 Communication Interrupt
def UART_HC12_ISR(t):
  global hc12_indicator
  msg_hc12=UART_hc12.readline()
  if msg_hc12 != None:
    msg_hc12=msg_hc12.decode('utf-8')
    print("Receive HC-12 Message:"+str(msg_hc12))
    hc12_indicator=hc12_indicator+1
  return
```

Title	The Main Programme for Patio 2		
Group	Image Group	**Week**	Week 8 and 9
Name	Jingluan Yang Qinlin Han	**Date**	2021.5.5

The Patio 2 has three main tasks, the color recognition, feeding and communication. The other important part is that the route of whole Patio. In order to further studying, we divided the whole programme into smaller stages.

Let the car arrives the target color and decide the threshold of color is the first part. The second step is to determine where to reach and the path of the car based on the threshold obtained in the front of analysis, and the next step is for the car to follow the specified path to the vicinity of the railing so that the car can enter the program of along the handrail.

In the fourth part, the car uses ultrasound to measure the distance in front and the right side to achieve the function of walking along the handrail. In the next step, the car according to the beacon tips, stop at the feeding location accurately, and control the robot arm to complete the feeding task. Finally, the car according to the established route to the wireless communication location, in order to complete the communication task.

And according to this logic we have determined the structure of the main program as follows:

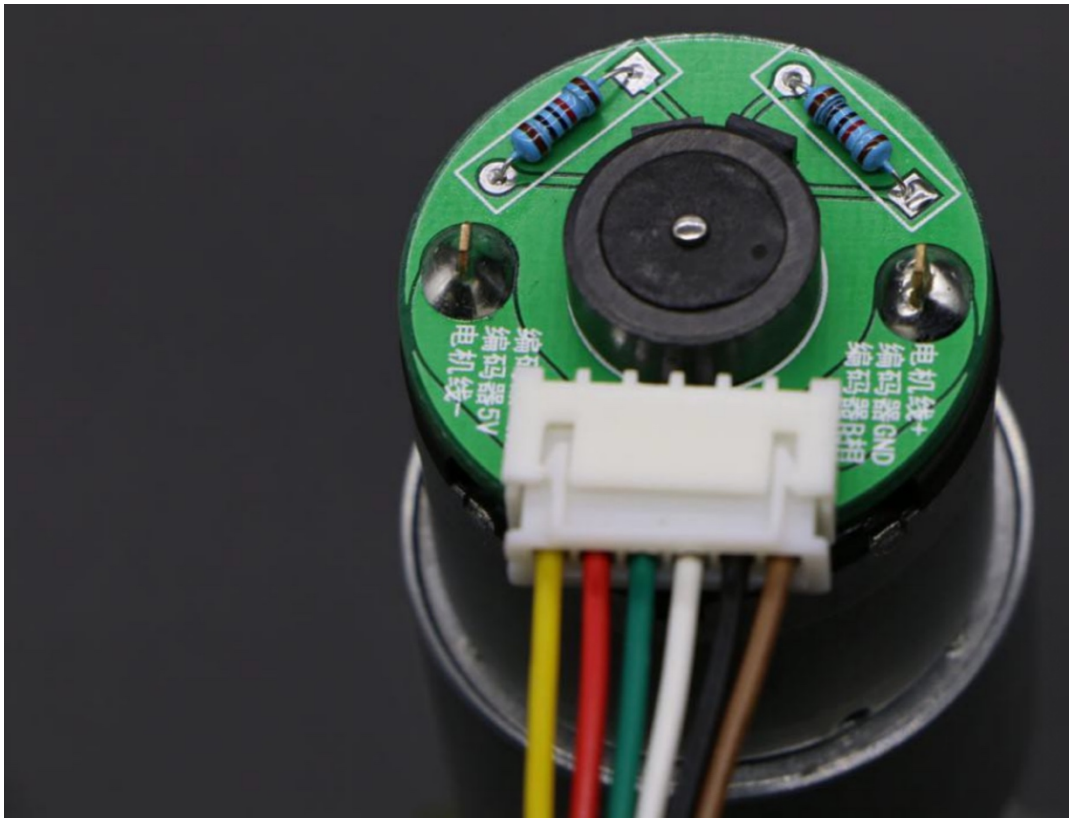
```
# Code Structure for Patio 2, TDPS
import sensor, image, lcd, time, utime, pyb, machine
from pyb import Pin

# ----- Functions -----
# Interboard Communication Interrupt
def UART_STM_ISR(t):
    msg_stm=UART_stm.readline() # To avoid reflection
    msg_stm=UART_stm.readline()
    if msg_stm != None:
        msg_stm=msg_stm.decode('utf-8')
        print("STM32 Message: "+str(msg_stm))
    return

# HC-12 Communication Interrupt
def UART_HC12_ISR(t):
    global hc12_indicator
    msg_hc12=UART_hc12.readline()
    if msg_hc12 != None:
```

```
        msg_hc12=msg_hc12.decode('utf-8')
        print("Receive HC-12 Message:"+str(msg_hc12))
        hc12_indicator=hc12_indicator+1
    return

# Ultrasonic Sensor Distance Calculation
def wave_distance_process(wave_echo_pin,wave_trig_pin):
    # Wave Start
    wave_trig_pin.value(0)
    utime.sleep_us(5)
    wave_trig_pin.value(1)
    utime.sleep_us(15)
    wave_trig_pin.value(0)

    # Receive echo signal
    try:
        tim_counter = machine.time_pulse_us(wave_echo_pin, 1, 30000)  # pin,
timeout=30000us (30000*0.017=510cm)
    except:
        print('error',wave_echo_pin.value())

    # calculate distance
    wave_distance = tim_counter*0.017
    return wave_distance

# Camera Initialization
sensor.reset()                      # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565)   # or sensor.GRAYSCALE
sensor.set_framesize(sensor.QVGA)
lcd.init(type=2)                    # Initialize the lcd screen.
clock = time.clock()                # Create a clock object to track the FPS.
THRESHOLD = (180, 255)
imgWidth=320                        # width=img.width() # 320 for QVGA
imgHeight=240                       # height=img.height() # 240 for QVGA

# Communication Initialization
# HC-12 UART Settings
UART_hc12=pyb.UART(1, 9600, timeout_char=1000)
UART_hc12.init(9600, bits=8, parity=None, stop=1, timeout_char=1000)

## HC-12 UART Interrupt
UART_hc12.irq(trigger = pyb.UART.IRQ_RXIDLE, handler = UART_HC12_ISR)

## Inter-Board Communication UART Settings
```

```
UART_stm=pyb.UART(3)
UART_stm.init(115200, bits=8, parity=1, stop=1, timeout_char=100)

## Inter-Board Comm UART Interrupt
UART_stm.irq(trigger = pyb.UART.IRQ_RXIDLE, handler = UART_STM_ISR)

# Ultrasonic Sensor Initialization
# Ultrasonic Sensor Pins
wave_echo_pin_1 = Pin('P7', Pin.IN, Pin.PULL_NONE)   # A0  P13, P7 D12,
wave_trig_pin_1 = Pin('P8', Pin.OUT_PP, Pin.PULL_DOWN)   # A1  P14, P8 D13,

wave_echo_pin_2 = Pin('P13', Pin.IN, Pin.PULL_NONE)   # A0  P13, P7 D12,
wave_trig_pin_2 = Pin('P14', Pin.OUT_PP, Pin.PULL_DOWN)   # A1  P14, P8 D13,

# HC-12 comm indicator
# Note variables in interrupt are global variables
global hc12_indicator
hc12_indicator = 0
# Direction and Speed
direction = 0 # The difference between car and the goal
# Define Stage

 # stage = 0  Color Recognition
 # stage = 1  Car arrives at the handrail
 # stage = 2 - Move to Feeding Window
 # stage = 3 - Feeding
 # stage = 4 - Move to Communication Point
 # stage = 5 - Communication and Move to Destination

 # Inter-board Communication with STM32
 stm_info=str(direction)
 UART_stm.write(stm_info)
```

Title	Speed Feedback		
Group	Car Group	**Week**	Week 10
Name	Xinyue Li Chenyang Fan	**Date**	2021.5.6

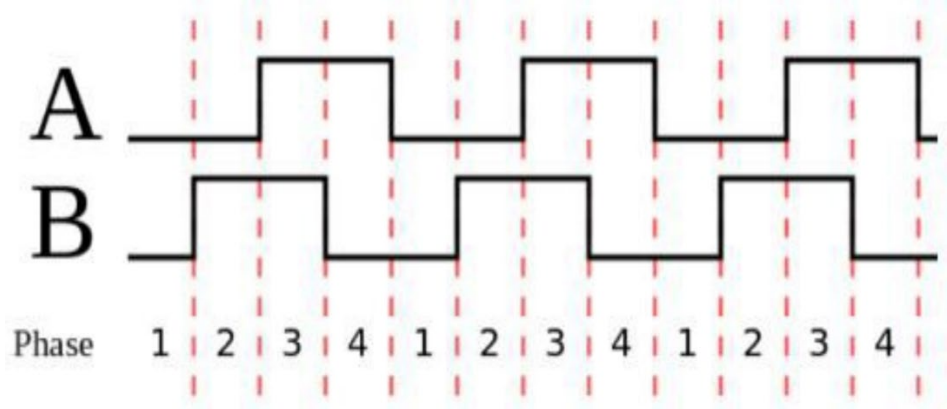
In order to enable the car to walk in a straight line with a small offset, whether on a cobblestone road or a flat road, we chose a Hall encoder to obtain the speed of the four wheels by reading the number of pulses received by the encoder.



(The red wire is the power cable； the green wire is the encoder phase A output;

the white wire is the encoder phase B output； The black wire is the encoder ground wire)

The encoder has AB phase output, so it can not only measure the speed, but also distinguish the direction of rotation. According to the wiring instructions in the above figure, we can see that we only need to supply 5V to the encoder power supply, and the square wave signal can be output through the AB phase when the motor is rotating. The encoder comes with a pull-up resistor, so there is no need for an external pull-up, and it can be directly connected to the microcontroller IO for reading.



The quadruple frequency technique is applied to the encoder by measuring the rising and falling edges of the A-phase and B-phase encoders. In this way, 12 times (3 cycles of 1234) can be counted in the same time. The number of lines per circle of our encoder is 390, so in the case of quadruple frequency and AB phase, there will be 1560 pulses per circle.

The single-chip microcomputer with encoder interface, STM32, can directly use the hardware to count and collect encoder data.

Encoder motor wiring

MG540 motor&encoder	
Encoder A phase	X1(CH1)
Encoder B phase	X2(CH2)
Encoder 5V	V+(Vin)
Encoder GND	GND
The motor 12V	12V power supply
The motor GND	Power supply GND

We use external interrupts to collect the number of pulses, and use formulas to calculate the speed of each wheel.

The code is shown below in the figures.

```
from pyb import Pin, Timer
#import pyb
import time
import cmath
import pyb


count=0

#class Encoder:
#     def __init__(self, encoder_freq):

#         self.pi = cmath.pi  # pi的值
#         self.encoder_precision = 4 * 390 * 30  # /48 # 编码器精度 = 倍频数*编码器精度（电机驱动线数）*电机减速比
#         self.encoder_freq = encoder_freq

#

#     def enc_to_speed(self):
#               while 1:
#                       time.sleep(0.1)
#                       speed = (count/encoder_precision) * (2 * self.pi * 0.0335) * 1000
#                       return speed
#                       count=0
#     def target_enc_process(self, speed):
#         '''
#         函数功能：将目标速度值转化为目标编码器值
#         入口参数：目标速度值
#         返回值  ：目标编码器值
#         # 目标编码器值=(目标速度*编码器精度)/(轮子周长*控制频率)
#         '''
#         enc = (speed *self.encoder_precision) / ((2 * self.pi * 0.0335) * 1000)
#         return enc
#def Encoder(count, extint1,extint2):




#       extint1.enable()
#       extint2.enable()
#       time.sleep(0.1)
#       extint1.disable()
#       extint2.disable()
#       return count

# def callback(line):
#       global count
#       count=count+1

def callback(line):


        global count
        count=count+1

# wheel 1 右后
A8 = Pin('A8')
B9 = Pin('B9')
extint1 = pyb.ExtInt(A8, pyb.ExtInt.IRQ_RISING_FALLING, pyb.Pin.PULL_DOWN, callback)
extint2 = pyb.ExtInt(B9, pyb.ExtInt.IRQ_RISING_FALLING, pyb.Pin.PULL_DOWN, callback)

count=0

# wheel 2 右前
B12 = Pin('B12')
B10 = Pin('B10')
extint3 = pyb.ExtInt(B12, pyb.ExtInt.IRQ_RISING_FALLING, pyb.Pin.PULL_DOWN, callback)
extint4 = pyb.ExtInt(B10, pyb.ExtInt.IRQ_RISING_FALLING, pyb.Pin.PULL_DOWN, callback)

count=0

# wheel 3 左后
A4 = Pin('A4')
A5 = Pin('A5')
extint5 = pyb.ExtInt(A4, pyb.ExtInt.IRQ_RISING_FALLING, pyb.Pin.PULL_DOWN, callback)
extint6 = pyb.ExtInt(A5, pyb.ExtInt.IRQ_RISING_FALLING, pyb.Pin.PULL_DOWN, callback)

count=0
```

```
# wheel 4 左前
A1 = Pin('A1')
B7 = Pin('B7')
extint7 = pyb.ExtInt(A1, pyb.ExtInt.IRQ_RISING_FALLING, pyb.Pin.PULL_DOWN, callback)
extint8 = pyb.ExtInt(B7, pyb.ExtInt.IRQ_RISING_FALLING, pyb.Pin.PULL_DOWN, callback)

count=0



def Encoder(n,f):
        global count
        if n==1:
                extint1.enable()
                extint2.enable()
                time.sleep(1/f)
                speed = (count / 1560) * (2 * 3.14159 * 0.0335) * f
                extint1.disable()
                extint2.disable()
        if n==2:


                extint3.enable()
                extint4.enable()
                time.sleep(1/f)
                speed = (count / 1560) * (2 * 3.14159 * 0.0335) * f
                extint3.disable()
                extint4.disable()
        if n==3:
                extint5.enable()
                extint6.enable()
                time.sleep(1/f)
                speed = (count / 1560) * (2 * 3.14159 * 0.0335) * f
                extint5.disable()
                extint6.disable()
                # pass

        else:
                extint7.enable()
                extint8.enable()
                time.sleep(1/f)
                speed = (count / 1560) * (2 * 3.14159 * 0.0335) * f
                extint7.disable()
                extint8.disable()
                # pass

        count = 0


        return speed
```

Title	Final Tracing Approach using OpenMV		
Group	Image Group	**Week**	Week 10
Recorder	Ziyi Xie	**Date**	2021.5.7

Final Tracing Approach 1: Edge Detection+ Surface Roughness

Purpose:
Today we went to the patio1 near the lake to see the real effect of our road tracing algorithm. We expect today we need to work hard and pay more attention to two things: one is for the image processing-how to get a stable and clear image without too much noise; the other thing is how to use the information we get to send instructions to control the car motion.

Problem:
This week, we tried our early version algorithm for patio1: road tracing. We found that only one threshold set for edge detection is not enough. For the other side of the road, this set threshold is hard to distinguish the edge very well.
Another problem is that the real road situation means there are always some regions in the camera that is mistaken as the road part. (Which would be binarized as the white points in the later process). This could mislead the car to go in the wrong direction.

Current solutions:
Problem 1 -Solution 1:
For the first problem, we came up with several methods. One way is to use one side of the road as the guide for the car motion instead of the two road sides.

Problem 1-Solution 2:
Another way is to segment the picture taken by OpenMV camera into two parts(one is for the left side of the road and the other is for right side of the road) and apply edge detection with different threshold to them.

```
# Binary threshold
thresholds = [(60, 255)]

width=320
height=240

# how many ROI regions on x&y-axis
xnum=2
ynum=2
ROInum=xnum*ynum

# calculate the starting points of ROIx,ROIy
```

```
xaxis=[0]*(xnum+1)
yaxis=[0]*(ynum+1)
for a in range(xnum+1):
    xaxis[a]=a*width/xnum
    yaxis[a]=a*height/ynum

ROIx=[0]*ROInum
ROIy=[0]*ROInum
for a in range(xnum):
    for b in range(ynum):
        ROIx[b+ynum*a]=xaxis[a]
for a in range(ynum):
    for b in range(xnum):
        ROIy[b + xnum * a] = yaxis[b]


while(True):
    img= sensor.snapshot()

    # find edges
    img.find_edges(image.EDGE_CANNY)

    # process every ROI region
    for i in range(ROInum):

statistics=img.get_statistics(roi=(ROIx[i],ROIy[i],width/xnum,height/ynum
))
        ROI=[ROIx[i],ROIy[i],ROIx[i]+width/xnum,ROIy[i]+height/ynum]

        # just to see the ROI rectangle?
        img.draw_rectangle(ROI)
        time.sleep(0.5)

        # set to 0/255
        if statistics.median()==255:  # statistics.mean()
            divide_process(ROI,255)
        else:
            divide_process(ROI,0)
```

Problem 2-Solution1:
Since the noise appeared in isolated white points far away from the picture center. We could choose ROI region(window selecting method) to only focus on the center part of the picture. Therefore, the interference effect will be reduced.

Problem 2-Solution2:
As the noise appeared in isolated white points, the road region has much higher density points than other regions. Therefore, we could divide the whole picture into small regions and count the density of each region to judge whether this region is road or not.

Result:
Using "`img.find_edges(image.EDGE_SIMPLE, threshold=(80, 200))`", the images are binarized according to different thresholds. We adjust the thresholds and choose the best one that has the clearest edges and traces. We also use

```
for i in range(1,320,2):
        for j in range(70,238,2):
            if img.get_pixel(i,j)==255:
                x+=i-160
                count+=1
    print("Dot Count=",count)
```

to calculate the density of the small region `to` judge whether the region is the road or not.
Additionally, we find that the performance is affected by the light and shades, so this solution is not robust against environmental changes. We need to test serval thresholds under different weather conditions.

Conclusion:
The performance of Edge Detection+ Surface Roughness has the best performance. Although, it still has some problems. For example, it will be affected by several factors, including weather, roughness of the road, lighting conditions, and shadows of environmental objects. Therefore, how to improve the stability and robustness is our next step.

Appendix: The final version code of today's test

```
import sensor, image, lcd, time, utime, pyb, machine
from pyb import Pin
from pid import PID # PID Algorithm

# ----- Functions -----
# ----- Communication -----
# Interboard Communication Interrupt
def UART_STM_ISR(t):
    UART_stm.readline() # To avoid reflection
    msg_stm=UART_stm.readline()
    if msg_stm != None:
        msg_stm=msg_stm.decode('utf-8')
        print("STM32 Message: "+str(msg_stm))
    return


# Initialization
sensor.reset()                          # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE) # or sensor.GRAYSCALE
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(15)
lcd.init(type=2)                        # Initialize the lcd screen.
clock = time.clock()                    # Create a clock object to track the FPS.

# HC-12 UART Settings
UART_hc12=pyb.UART(1, 9600, timeout_char=1000)
UART_hc12.init(9600, bits=8, parity=None, stop=1, timeout_char=1000)
#
# ## HC-12 UART Interrupt
UART_hc12.irq(trigger = pyb.UART.IRQ_RXIDLE, handler = UART_HC12_ISR)
#
## Inter-Board Communication UART Settings
UART_stm=pyb.UART(3)
UART_stm.init(115200, bits=8, parity=1, stop=1, timeout_char=100)

## Inter-Board Communication UART Interrupt
UART_stm.irq(trigger = pyb.UART.IRQ_RXIDLE, handler = UART_STM_ISR)


# PID Instantiation
x_pid=PID(p=1.5,i=0.02,d=-0.001,imax=100)
```

```
while(True):
    clock.tick()
    img = sensor.snapshot()
    img.find_edges(image.EDGE_SIMPLE, threshold=(80, 200))
    # 6.8 5:00  阴天  threshold=(120,200),count=180
    # 6.8 7:00  阴天  threshold=(80,200),count=130
    left_pwm=0
    right_pwm=0
    x=0
    count=0
    for i in range(1,320,2):
        for j in range(70,238,2):
            if img.get_pixel(i,j)==255:
                x+=i-160
                count+=1
    print("Dot Count=",count)


    # # Measure Distance
    #
distance1=wave_distance_process(wave_echo_pin=wave_echo_pin_1,wave_trig_pin=wave_t
rig_pin_1)
    #
distance2=wave_distance_process(wave_echo_pin=wave_echo_pin_2,wave_trig_pin=wave_t
rig_pin_2)

    # Communication with HC-12
    #hc12_info="["+str(utime.time())+"] Direction:"+str(direction)+" FPS:"+str(clock.fps())+"
Distance:"+str(distance1)+" cm "+str(distance2)+" cm \n"
    #UART_hc12.write(hc12_info)
    #print("hc12_indicator="+str(hc12_indicator))

    # Communication with STM32
    # Message Generation
    command_bit=1 # NO stop
    if left_pwm>0:    # 2xx - forward - positive PWM
        left_pwm=200+left_pwm
    else:
        left_pwm=100-left_pwm

    if right_pwm>0:    # 1xx - backward - negative PWM
        right_pwm=200+right_pwm
    else:
        right_pwm=100-right_pwm
```

```
stm_info=str(left_pwm)+str(right_pwm)+str(command_bit)
UART_stm.write(str(left_pwm)+str(right_pwm)+str(command_bit))
print(stm_info)

# LCD Display
lcd.display(img, x=0, y=0, x_scale=0.5, y_scale=0.32)
```

Title	Inter-board Communication and Coding Scheme		
Group	Image	**Week**	Week 10
Recorder	Zhangchen Xu	**Date**	2021.5.8

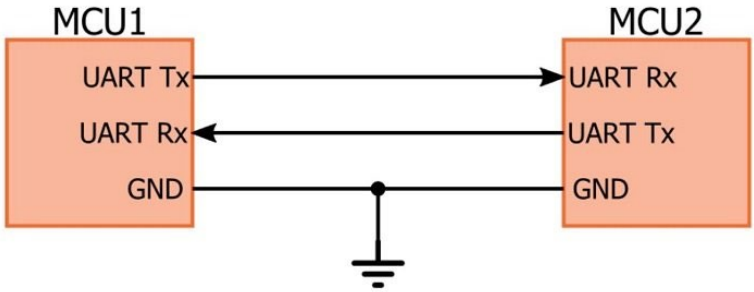
After discussion, we decided to use UART as our inter-board communication protocol. Compared to SPI and IIC protocol, UART has the following advantages:

- ✓ Requires fewer lines (TX, RX, GND)
- ✓ No clock signal required
- ✓ Powerful MicroPython functions

However, it also has limitations:

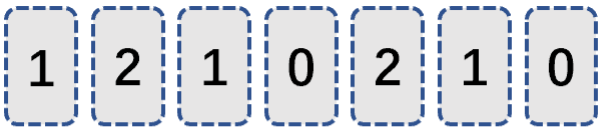
- Slow speed
- Same Baud rate
- Small Data frame

Consider the patio requirements, we found that OpenMV only need to send a very short instruction to the stm32F411 in every second, so speed and data frame limitation is not a problem. So we determine that we choose UART. The figure below shows the wire connection between two boards.



Coding Scheme

For different missions that may appear in the patios, I designed a coding scheme that are flexible, easy to decode and also easy to read by human. It has maximum 7-bits long, and the first bit is the mode bit. The different modes are shown in the figure below.



Bit 0: Mode Bit
= 1: PWM Mode (OpenMV-side PID)
= 2: Turn in Place
= 3: Turn while forwarding
= 4: Straight line algorithm 1
= 5: Straight line algorithm 2

And the whole protocol with examples are shown in figure below.

Bit 1: Mode Bit
- Mode = 0: STOP **0**
- Mode = 1: PWM Mode **1**210210 PWM10, Go forward
  - Continuous Control by PID in OpenMV
  - Bit 2-4: Left PWM
    - Bit 2 = 1: Negative PWM (backward)
    - Bit 2 = 2: Positive PWM (forward)
    - Bit 3-4: Magnitude(00-99)
  - Bit 5-7: Right PWM
    - Bit 5 = 1: Negative PWM (backward)
    - Bit 5 = 2: Positive PWM (forward)
    - Bit 6-7: Magnitude(00-99)
- Mode = 2: Revolving on the spot **2**1090 Turn left for 90 degree
  - Bit 2-5: Angle Value
    - Bit 2 = 1: Turn Left
    - Bit 2 = 2: Turn Right
    - Bit 3-5: 0-360
- Mode = 3: Turn while Moving **3**1020 turn left for 20 degree while moving
  - speed is defined by the car
  - Bit 2-5: Angle Value
    - Bit 2 = 1: Turn Left
    - Bit 2 = 2: Turn Right
    - Bit 3-5: 0-360
- Mode = 4: Straight Line Algorithm 1 **4**
- Mode = 5: Straight Line Algorithm 2 **5**

Title	Group Seminar 7 Week 10		
Group	All	**Week**	Week 10
Recorder	Zhangchen Xu	**Date**	2021.5.9

On May 9th, the whole group held a 1-hour group seminar at the Tencent conference platform. Due to the holiday, last seminar (in Week 9) is cancelled.

Because of 2-week period, both groups have many new results to share. The car group has finally got the correct speed information from Hall sensor with the help of some members in the image group, so they can continue working on the PID algorithm.

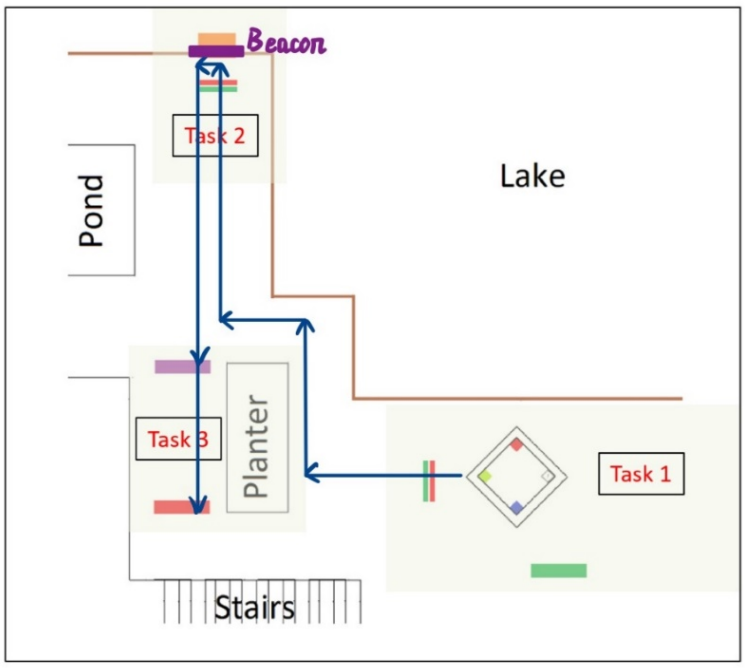
Another good new is that the image group has determined the final solution for the edge detection and decided to use PWM to control wheels of cars directly from instruction of OpenMV. As a result, a inter-board communication protocol using UART is written, and several modes are designed and coded. Moreover, the HC-12 communication module is now ready to use.

Title	The Route of Feeding Task (initial)		
Group	Image Group	**Week**	Week 10&11
Name	Han Qinlin Yang Jingluan	**Date**	2021.5.11

Beacon type: color beacon.
The principle is the same as that in color recognition of task 1 in patio 2.

The driving route of the trolley:
After discussion, we initially plan that after completing the task of color recognition, the car would go straight to the front of the planter and use ultrasonic ranging to control the distance between the planter and our car. After the forward ultrasonic senor detects that the distance between the car and the planter is less than 30 centimeters, OpenMV would send and instruction to make the car turn right. Next, The car heads straight toward the railing. During the process of going straight, the left side ultrasonic is used to control the distance from the planter (kept at about 20cm). The front ultrasonic detects the distance 20cm from the railing, and the car turns left. At the next stage, we plan to place the color beacon directly above the feeding place so that after the ca turning left, OpenMV could plan the route by looking for the color beacon to reach the feeding place acting like the task of color recognition.

The route plan and the placement of the beacon are shown in the following figure:



After finishing the task of feeding, the car will return. When the left-side ultrasonic sensor detect the distance less than 80 cm, the car stops to communicate wirelessly. Finally, go straight to pass through the finish line.

Title	Image Processing for Car Motion using OpenMV		
Group	Image Group	**Week**	Week 11
Recorder	Ziyi Xie	**Date**	2021.5.14

Final Tracing Approach 1: Car motion control combined with image

Purpose:
Today we went to the patio1to continue the last week's mission: how to use the information we get to send instructions to control the car motion.

Procedure:
Based on the image we got before, we can characterize the picture as the mathematics variables (vectors or matrix). We use weight 0 and 1 for the white and black points separately and use statics information to calculate the middle line of the road. Then, just let the car to move along the middle line for road tracing.

Problem1:
We found that the picture of the Openmv is not so stable and the processing speed is a little slow, which means the car will continue the instructions one or two second before before the new instructions is sent. This result bad things when the car is turning left or right, it will go out of the road and cannot find the road anymore.

Current solutions 1:
One way is very simple and easy, just slow down the car speed or let the car moves a smaller distance for each instruction. But it may make the movement not smooth.

Problem2:
One important thing for the turning time is that the road should consist of at least 30% of the car camera vision. Otherwise it is really hard for the car to determine which region is the road or which direction it should go further.

Current solutions 1:
Just change the position of the camera. Make it horizontally near the car instead of extend too far. Otherwise, when turning round, the car is still in the road region but the camera is already outside the road region.

Current solutions 2:
Set up an error-correction procedure. For both going straight or turning around. Car has the risk of making a wrong decision. Therefore, we can let the car self-check and self-correction. The principle is straightforward- always make sure there are enough road features(characterized by the white points) in the machine version.

```
    if count<130:
        left_pwm=-15
        right_pwm=-15
    else:
        x_av=x/count
        x_error=x_av
        x_output=x_pid.get_pid(x_error,0.8*0.5)
        img.draw_rectangle((160+int(x_error),0,10,243))
        print('x_output:',x_output)
        print('x_error:',x_error)
        left_pwm=int(25-x_output)
        right_pwm=int(25+x_output)
```

Conclusion:

We finally combined problem1-solution1 and problem2-solution2. The result is quite good. The car will succeed in most cases and even it is at the edge of the road, it will begin self-check and self-correction and return to the middle of the road.

Title	Why We Forsake MPU-6050		
Group	Car Group	**Week**	Week 11
Name	Ziyang Long	**Date**	2021.5.15

The first gyroscope we tried was MPU6050, which uses the I2C protocol.

Firstly, it is significant to master the relative information of I2C protocol. To better understand I2C protocol, we refer to the relative materials and find that there are two different methods to write the code, which are 'pyb' and 'machine' respectively. To be specific, we can write code either '*from pyb import I2C*' or '*from machine import I2C*'

The figures below illustrate the detailed codes for the two methods respectively.

For 'pyb':

```
i2c = I2C(1,I2C.MASTER,baudrate=400000)
mem_write
mem_read
```

For 'machine':

```
i2c=I2C(scl='PB6', sda='PB7', freq=400000)
readfrom_mem
writeto
```

After understanding the basic coding of I2C protocol, we took step to learn the working principles of MPU6050. MPU can output three angular velocities 'Gyro' around three axis 'x, y, z', where the unit is °/s. After obtaining the angular velocity, a formula is necessary to change the angular velocity into angle, and the easiest way is:

$$angle = angular\ velocity \times time$$

The process of obtaining angle can be regarded as integral calculation when adding the angular velocity per unit time together. The code below presents the main logic of calculation angle.

```
sum=0
from time import sleep
while 1:
    sum=sum+MPU.read_Gyro_z()*0.002#采样率是1kHz,后面延时了0.1ms，所以用0.002
    if(sum>=90 or sum<=-90):
        print('Sum is ',sum)
        sum=0
        sleep(0.5)
    sleep(0.001)
```

Although MPU6050 performs well (the excellent ability to capture the change of position) when rapidly rotating 90 degrees, it might lead to large error if the slight deflection occurs in unit time. On the condition that the small deflection occurs, the angular velocity will be unexpected small, and due to the limitation of MPU6050's sampling rate, (maximum value is 8KHz), some unwanted omission may happen during the accumulation of the angular velocity per unit time

```
def reset(self):
    self._write_byte(MPU_PWR_MGMT1_REG, 0x00) # Configure the power management register openMPU6050
    self._write_byte(MPU_GYRO_CFG_REG,  config_gyro_range<<3) # gyro sensor,±2000dps
    self._write_byte(MPU_ACCEL_CFG_REG, config_accel_range<<3)#  acceleration sensor ,±2g
    self._write_byte(MPU_SAMPLE_RATE_REG,0x01)#The sampling frequency >512
    self._write_byte(MPU_CFG_REG,0x00)#Set the digital low pass filter to the first mode and the output frequency is 8KH
    self._write_byte(MPU_INT_EN_REG,0X00) #Close all interrupts
    self._write_byte(MPU_USER_CTRL_REG,0X00) #I2C main mode off
    self._write_byte(MPU_FIFO_EN_REG,0X00) #close FIFO
    self._write_byte(MPU_INTBP_CFG_REG,0X80) #INT Pin low level valid

    buf = self._read_byte(MPU_DEVICE_ID_REG)
    if buf != self._address:
        print("NPU6050 not found!")
    else:
        pass
```

Considering the defects of MPU6050 mentioned before, we determined to give up using MPU6050.

Title	Group Seminar 8 Week 11		
Group	All	**Week**	Week 11
Recorder	Zhangchen Xu	**Date**	2021.5.16

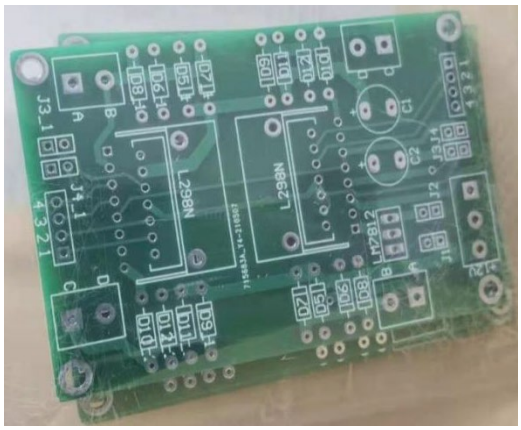
On May 16th, the whole group held a 1-hour offline group seminar at main building A1-305.

The good new from the image group is that they use a OpenMV-end PID algorithm successfully finished the work of tracing. The result is quite good. The car will succeed in most cases and even it is at the edge of the road, it will begin self-check and self-correction and return to the middle of the road. We are so happy about this.

However, there are bad news from image group. The MPU6050 gyroscope is problematic and is not suitable for our mission. when rapidly rotating 90 degrees, it might lead to large error if the slight deflection occurs in unit time. After discussion, the whole team agree to change this module and buy a more powerful module.

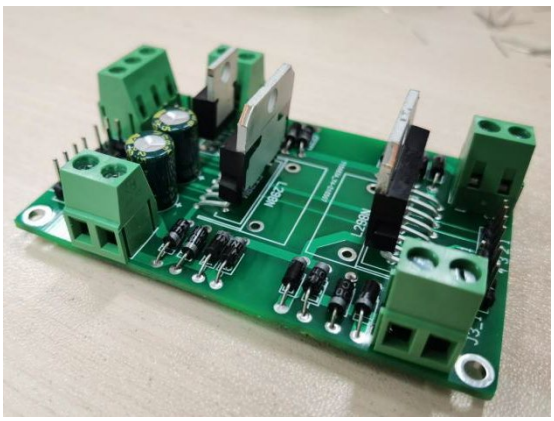
Title	L298N driver module assembly and welding		
Group	Car Group	**Week**	Week 12
Recorder	Weizhe Zhao	**Date**	2021.5.20

Providing factory with the designed PCB layout, the PCB board for L298N driver module is printed out.



Printed Driver Module Board

Using chips we bought and other electronic components provided in the laboratory, we welded the board. Then we tested its output with given input and found no problem, so the board is ready for use.



Welded Driver Module Board

Title	Group Seminar 9 Week 12		
Group	All	**Week**	Week 12
Recorder	Zhangchen Xu	**Date**	2021.5.23

On May 23rd, the whole group of MAD MAX held a 1-hour group seminar online in Tencent platform.

In the last week, the car group had made a new L298N module which can support 4 motors simutaneously, and the test result was great. Also, they finally determined the robotic arm solution for patio 2. They designed a four degrees of freedom robotic arm, which can move more flexibly for releasing foods.

As for car group, they started Road Tracing Test & Improvement for patio 1.

Title	The Design of Robotic Arm for Feeding		
Group	Car Group	**Week**	Week 13
Name	Xinyue Li Chenyang Fan	**Date**	2021.5.24

The object of this week is to design the robotic arm for feeding. the process of design is mainly divided into 3 stages

- **Identify the Requirement and Problem Formulation**

  Initially, it is significant to identify the requirement of the robotic arm. The robot is required to release the fish food at the release point. The figure below shows the size of the railing at releasing point.

  The programming requirement is that the robotic arm receives instructions from openmv, when openmv transmits the releasing instruction, the robotic arm release the fish food to the target position.

  It is noteworthy that, due to the sophisticated design of our car, the size of our robotic arm should be carefully controlled. Also, we are required to avoid influencing the view of camera which is displayed at the front of the car.

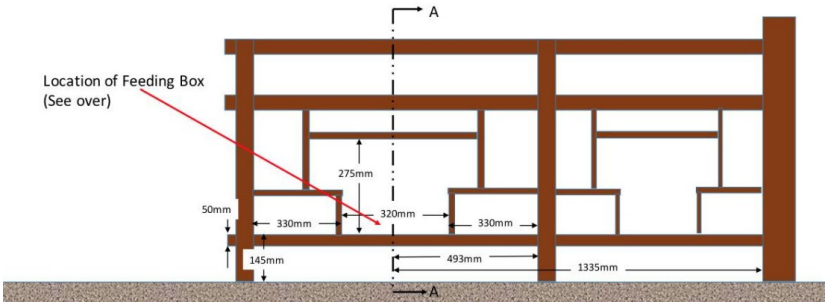


Figure 1. The size of releasing point

- **Ideas of robotic arm:**

  At the inception, we adopted two methods.

  Method 1:

  We combined a Conveyor belt and a robotic claw together, after receiving instruction of releasing, the conveyor belt transmits the claw to the release point, then the claw is programmed to release the food.

  This method effectively solves the problem of influencing the camera, however, it transmits stiffly and cannot adjust the angle flexibly.

Figure 2. The figure of conveyor belt

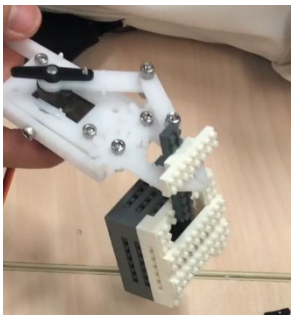


Figure 3. The figure of robotic claw

Method 2:

We designed a four degrees of freedom robotic arm, which can move more flexibly.

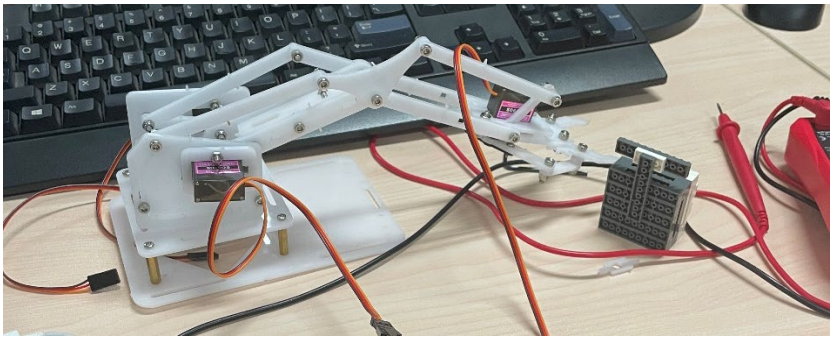


Figure 4. The figure of Robotic arm

- **The final design of robotic arm and basic working principle**

  At last, we adopted the four degrees of freedom robotic arm. The four main joints are controlled by 4 steering gear engines.

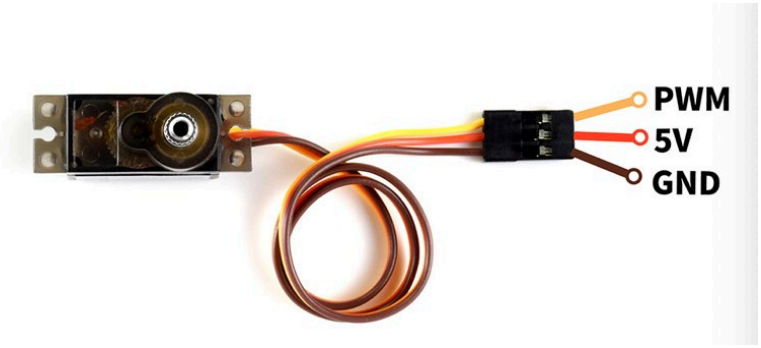


Figure 5. The figure of steering engine

The steering engine is powered by 5V, and can rotate to different angles by adjusting the PWM of the signal that transmits to the steering gear engine. The code for controlling the robotic arm is shown as below:

main.py - C:/Users/97357/Desktop/main.py (3.9.4)

File Edit Format Run Options Window Help

```
# Steering engine
#爪子8&11
#右侧舵机初始9，结束12
#左侧舵机结束12

A3 = Pin('PA3') #control the left steering gear engine
tim = Timer(2, freq=50)
ch = tim.channel(4, Timer.PWM, pin=A3)

B3 = Pin('PB3')# control the right steering gear engine
tim1 = Timer(2, freq=50)
ch1 = tim.channel(2, Timer.PWM, pin=B3)

B0 = Pin('PB0')#for claw
tim2 = Timer(1, freq=50)
ch2 = tim.channel(3, Timer.PWM, pin=B0)

A7 = Pin('PA7')#for the fundation
tim3 = Timer(3, freq=50)
ch3 = tim.channel(2, Timer.PWM, pin=A7)

i=9
m=7
ch1.pulse_width_percent(9)
while True:
    ch2.pulse_width_percent(7.5)
    ch3.pulse_width_percent(8)
    i+=0.1
    m+=0.1
    time.sleep(0.3)
    print(i)
    if i>12.3:
       i=12.3
       ch2.pulse_width_percent(11)
    if m>10:
        m=10
    ch.pulse_width_percent(i)
    ch1.pulse_width_percent(m)
    pass
```

Ln: 48 Col: 30

Figure 6. The figure for code

Title	The idea for turning on the spot		
Group	Car Group	**Week**	Week 13
Name	Xinyue Li Yuchen Yao	**Date**	5.27

It is essential for car to turn on the spot at a target angle precisely, especially when finishing the task of patio two. To be specific, turn at an angle precisely is the foundation for performing the other tasks, as the car may fail to approach the desired position without a correct initial direction.

Firstly, the stm32 will received a turn angle from Openmv, then the turn angle will be transmitted into the target angle that we desired to turn to. The gyroscope offers z angle to the stm32, which denotes the current angle (direction)of the car. The actual angel we want to turn is:

$$delta = z\ angle - turn\ angle(the\ old\ version, will\ revise\ later)$$

With delta, we can define the direction of turning:

$$delta > 0, anticlockwise\ \ delta < 0, clockwise$$

If delta <2°,we choose to ignore the difference angle and stay unchanged, or the car will repeatedly revise it angle for a long time.

However, we faced with a challenge that the gyroscope provides the angle ranged from -180° to 180 °, for instance, the problem occurs when z angle=179°, while target angle=-179°, the actual result we want is to turning anticlockwise for 2°, the result will be turning clockwise for 358°. To solve this problem, we revised our codes to calculate the correct turn angle and added a list of ‘if, else’.

The code below illustrates how to realize turning precisely with the instructions.

```
UART_Gyroscope = UART(2)
UART_Gyroscope.init(38400, bits=8, parity=None, stop=1, timeout_char=100)
count=0
turn_angle=90     # >0: anticlockwise, <0: clockwise
turn_complete=0
global z_angle
UART_Gyroscope.irq(trigger = UART.IRQ_RXIDLE, handler = UART_Gyroscope_ISR)
global original_angle
time.sleep(1)
# get target angle in first time interrupt
print(type(original_angle))
target_angle=original_angle+turn_angle
if target_angle<=-180:
      target_angle=target_angle+360
if target_angle>180:
      target_angle=target_angle-360

# # turning
while(turn_complete==0):
  if z_angle<=-90 and z_angle>=-180 and target_angle<=180 and target_angle>=90:
      delta=z_angle-target_angle+360
  elif z_angle>=90 and z_angle<=180 and target_angle>=-180 and target_angle<=-90:
      delta=z_angle-target_angle-360
  else:
      delta=z_angle-target_angle
      if z_angle-target_angle>=180:
          delta=-(360-delta)
```

```
      if z_angle-target_angle<=-180:
          delta=-(360+delta)
  # print(delta,z_angle)
  if abs(delta)>=2:
      print(delta,z_angle,target_angle)
      turn(delta)
  if abs(delta)<2:
      # print(delta)
      turn_complete=1
      s_original_angle=z_angle  #indicate turning complete, get out of turning loop
# Car.run_(0,0,0,0,0,0,0,0)               #stop turning
```

Title	Read z-angle from JY901 Gyroscope		
Group	Car Group	**Week**	Week 13
Recorder	Weizhe Zhao	**Date**	2021.5.28

For communication between gyroscope and stm32, we apply UART protocol as it is simple to use and there are vacant UART pins of stm32.

```
UART_Gyroscope = UART(2)
UART_Gyroscope.init(9600, bits=8, parity=None, stop=1, timeout_char=100)
UART_Gyroscope.irq(trigger = UART.IRQ_RXIDLE, handler = UART_Gyroscope_ISR)
```

Initialize and set parameters for UART from stm32

In the interrupt service routine of UART, stm32 read data sent from the gyroscope and call the data processing method to extract z-axis angle information. We also define the variable “count” to count the interrupt time in order to distinguish the first interrupt. In the first interrupt, the read angle is set to be original angle for the smart car.

```
def UART_Gyroscope_ISR(t):
    global turn_angle
    global count
    global signal
    global command
    global original_angle
    global z_angle
    msg_Gyroscope=UART_Gyroscope.read(UART_Gyroscope.any())
    #x = int.from_bytes(msg_Gyroscope,'big')
    z_angle = jy901.DueData(msg_Gyroscope)
    if count==0:
        original_angle=z_angle
    count=count+1
    return
```

Define UART ISR

```
def DueData(inputdata):   #新增的核心程序，对读取的数据进行划分，各自读到对应的数组里
    global  FrameState    #在局部修改全局变量，要进行global的定义
    global  Bytenum
    global  CheckSum
    global  a
    global  w
    global  Angle
    for data in inputdata:  #在输入的数据进行遍历
        if FrameState==0:   #当未确定状态的时候，进入以下判断
            if data==0x55 and Bytenum==0: #0x55位于第一位时候，开始读取数据，增大bytenum
                CheckSum=data
                Bytenum=1
                continue
            elif data==0x51 and Bytenum==1:#在byte不为0 且 识别到 0x51 的时候，改变frame
                CheckSum+=data
                FrameState=1
                Bytenum=2
            elif data==0x52 and Bytenum==1: #同理
                CheckSum+=data
                FrameState=2
                Bytenum=2
            elif data==0x53 and Bytenum==1:
                CheckSum+=data
                FrameState=3
                Bytenum=2
        elif FrameState==1: # acc    #已确定数据代表加速度

            if Bytenum<10:            # 读取8个数据
                ACCData[Bytenum-2]=data # 从0开始
                CheckSum+=data
                Bytenum+=1
```

Part of data processing method

Title	Apriltag Test		
Group	Image Group	**Week**	Week 13
Recorder	Ziyi Xie	**Date**	2021.5.29

Final Tracing Approach 1: Apriltag

Purpose:
In order to go straight through the bridge for patio1. We need to know when to turn directions and adjust the current positions of the car.
Apriltag could satisfy our requirements as it can tell the 3d dimensional coordinates to judge the car positon

Procedure:

1、Measure the constant parameter for the real distance and the image pixel:

```
#kz=real/tz
#20 cm kz=25cm/6.734
#tz=
#real distance =kz*tz=
```

2、Print the certain id Apriltag and try to position it.

3、Get the distance in the z direction which is the horizontal distance between the beacon and the car.

4、Get the rotation of the beacon to adjust the car position to has a 90 degrees angle between the car and the bridge. Then send turning 90 degrees instructions to go through the bridge.

Problem:

Apriltag cannot use QVGA/VGA as our openmv does not have enough RAM for processing pictures. It could be a problem for road tracking as it will reduce the solution of the pictures.
We will try to test QQVGA for our road tracking algorithm or we may give up the Apriltag method and use other beacons.

Appendix:

```
# AprilTags Example
# This example shows the power of the OpenMV Cam to detect April
Tags
```

```
# on the OpenMV Cam M7. The M4 versions cannot detect April Tags.

import sensor, image, time, math

sensor.reset()
sensor.set_pixformat(sensor.GRAYSCALE)
sensor.set_framesize(sensor.QQVGA) # we run out of memory if the
resolution is much bigger...
sensor.skip_frames(15)
sensor.set_auto_gain(False)  # must turn this off to prevent image
washout...
sensor.set_auto_whitebal(False)  # must turn this off to prevent
image washout...
clock = time.clock()

f_x = (2.8 / 3.984) * 320 #
f_y = (2.8 / 2.952) * 240 #
c_x = 320 * 0.5 # (image.w * 0.5)
c_y = 240 * 0.5 # (image.h * 0.5)

def degrees(radians):
    return (180 * radians) / math.pi

while(True):
    clock.tick()
    img = sensor.snapshot()
    for tag in img.find_apriltags(fx=f_x, fy=f_y, cx=c_x, cy=c_y):
        img.draw_rectangle(tag.rect(), color = (255, 0, 0))
        img.draw_cross(tag.cx(), tag.cy(), color = (0, 255, 0))
        rotation_simple=degrees(tag.rotation())
        print_args = (tag.x_translation(), tag.y_translation(),
tag.z_translation(), \
            degrees(tag.x_rotation()), degrees(tag.y_rotation()),
degrees(tag.z_rotation())


        print_args)
        print(tag.id(),rotation_simple,tag.z_translation())

    print(clock.fps())
```

Title	Group Seminar 10 Week 13		
Group	All	**Week**	Week 13
Recorder	Zhangchen Xu	**Date**	2021.5.30

On 2021.5.30, Sunday, the whole group of the MAD MAX held a 1-hour group seminar at main building A1-305.

In the last week, the image group tested a new beacon, that is Apriltag. AprilTag is a visual reference system suitable for a variety of tasks including augmented reality, robotics and camera calibration. Using a common printer to create the target, AprilTag can calculates the precise 3D position, orientation and identification of the tag in relation to the camera, which is suitable for our mission. The test shows it works well. Also, they are conducting color modification for patio 2.

The main contribution of the car group in last week is the idea for turning on the spot. With the new gyroscope, they are able to turn on the spot at a target angle precisely, which is important when finishing the task of patio two.

Title	Color Recognition Modification		
Group	Image Group	**Week**	Week 13 and 14
Name	Jingluan Yang Qinlin Han	**Date**	2021.6.3

During this lab, we start testing the color recognition in the practical environment and during this process, we find four main question and provide the new idea to sole these problems.

We have implemented the threshold of extracted the target color block during previous experiments by OpenMV, and to find the same threshold color block based on this threshold. **However**, in the observing of the later experiment, because the actual environment is complex and the threshold range obtained by the mg.get_histogram( ) is wide and the error is large, so we choose to pre-set the color threshold, and then match the target color with color threshold library, in order to improve the accuracy of color recognition.

```
thresholds=[(23, 32, -3, 15, -58, -22),     #pink
            (42, 60, -27, 1, 17, 127),      #yellow
            (49, 0, -128, 127, -128, 127)]  #blue

    for blob in img.find_blobs(thresholds, pixels_threshold=200,
area_threshold=200):
        img.draw_rectangle(blob.rect())
        img.draw_cross(blob.cx(), blob.cy())
        #print(blob.code(),type(blob.code()))
        if blob.code()==1:
            m=1
        elif blob.code()==2:
            m=2
        else:
            m=3
```

The second problem in the actual test is that because the camera's field of view angle is limited, we provide the car with an initial angle of rotation based on the recognized target color, ensuring that the second color block to be reached determines the car's forward path within the vehicle's field of view.

When we locate the color block, we use blob.cx() and blob.cy() to get the coordinates of the center point of the color block and calculate the deflection angle between the color block and the center of the car by the code and control the direction of the car from this angle.

The code for calculating the angle is following:

```
    for blob in img.find_blobs([thresholds1], pixels_threshold=20,
area_threshold=20, merge=True,margin=10):
        img.draw_rectangle(blob.rect())
        img.draw_cross(blob.cx(), blob.cy())
        img.draw_line(160,240,blob.cx(),blob.cy(),color=180, thickness=2)
        Intermediate_Variable = (160-blob[5])*(160-blob[5]) + (240-
blob[6])*(240-blob[6])
        Distance= math.sqrt(Intermediate_Variable)
        theta_err = math.asin((160-blob[5])/Distance)*(180/math.pi)
```

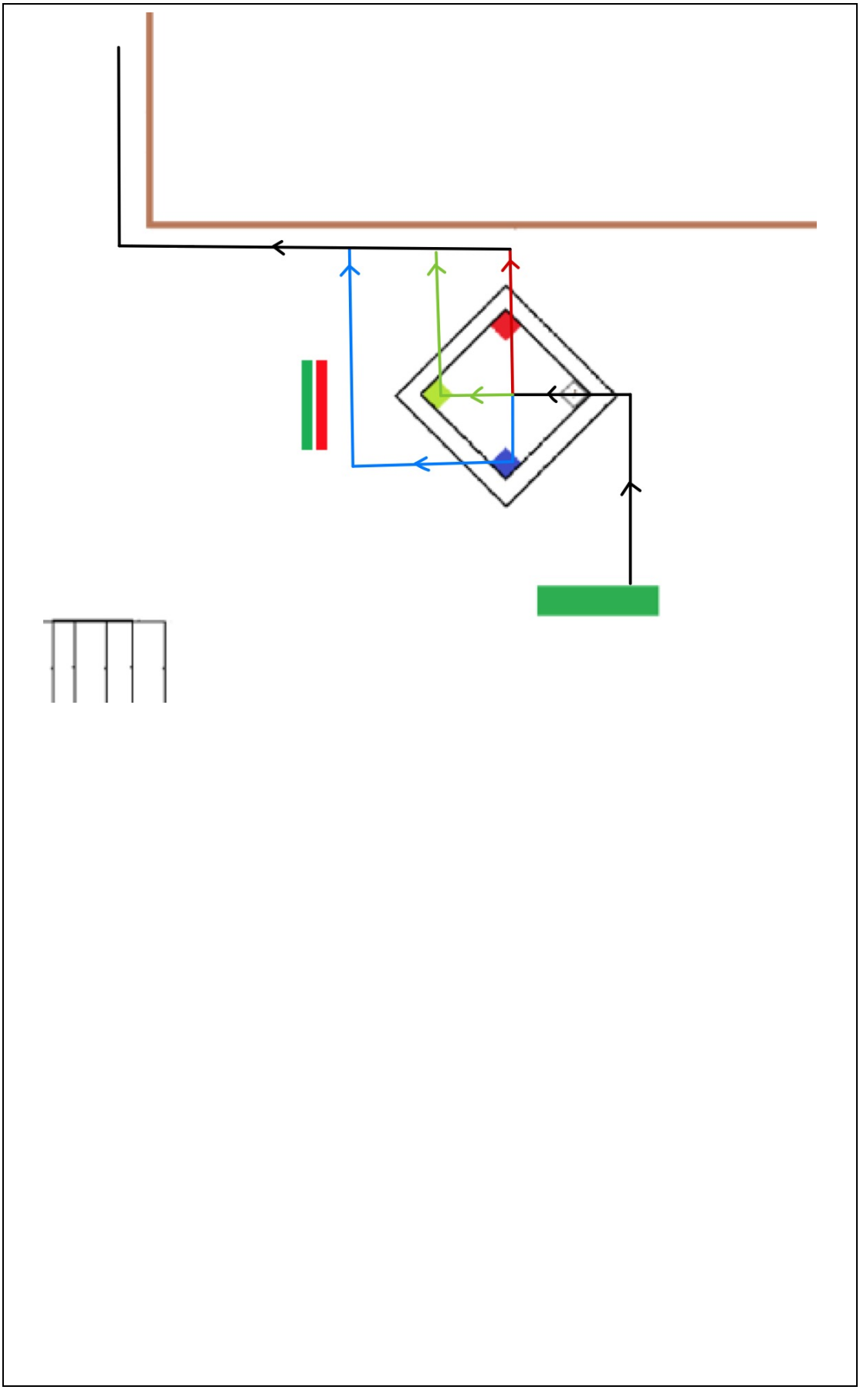
Another problem that needs to be solved is that when the car is very close to the color block, the error of the central coordinate judgment of the color block is large, the route of the car will also have a large deviation. In order to solve this problem, we use the blob[4] function to acquire the pixel value of the color block, and according to the color block pixel value The size of the car and the distance between the color block, when the car adjusts the direction and moves forward to a certain distance, the end of the image processing for the angle of the judgment and continue to go straight, when the pixel value of the color block is larger, the car has reached the color block.

```
for blob in img.find_blobs([thresholds1], pixels_threshold=20,
area_threshold=20, merge=True,margin=10):
        img.draw_rectangle(blob.rect())
        img.draw_cross(blob.cx(), blob.cy())
        Distance= math.sqrt(Intermediate_Variable)
        if blob[4]>100:
             stop_bit=0
        else:
             stop_bit=1
```

And this code could also used to find the beacon.

The last problem is that when a small car on the track completes a turn on a tile, the slip phenomenon is more serious and the instruction cannot be completed accurately. Thus the car will adjust the direction in the center of the square.

To avoid the above problems, the last color-recognized path of the car is shown in the following image:



Title	Group Seminar 11 Week 14		
Group	All	**Week**	Week 14
Recorder	Zhangchen Xu	**Date**	2021.6.6

On 2021.6.4, Sunday, the whole group of the MAD MAX held a 1-hour group seminar online at Tencent Conference.

The members in the image group shows us the modified plan for patio 2, and after discussion, we determined the final route and the software structure for patio 2. We designed several stages and switch them while moving.

There is no news from car group. They are conducting the car system optimization, namely optimize codes (e.g., straight algorithm) for better performance.

In the end, we discussed the presentation, deadline of TDPS and the devision of work of the final report.

Title	Send down-bridge signal to OpenMV		
Group	Car Group	**Week**	Week 15
Recorder	Weizhe Zhao	**Date**	2021.6.7

In order for Openmv to recognize that the car is driving down bridge, y-axis angle information is required. To meet this requirement, we adjust the data processing method of gyroscope message, and send y-axis angle to Openmv.

```
def get_angle(datahex):
    rxl = datahex[0]
    rxh = datahex[1]
    ryl = datahex[2]
    ryh = datahex[3]
    rzl = datahex[4]
    rzh = datahex[5]
    k_angle = 180.0

    angle_x = (rxh << 8 | rxl) / 32768.0 * k_angle
    angle_y = (ryh << 8 | ryl) / 32768.0 * k_angle
    angle_z = (rzh << 8 | rzl) / 32768.0 * k_angle
    if angle_x >= k_angle:
        angle_x -= 2 * k_angle
    if angle_y >= k_angle:
        angle_y -= 2 * k_angle
    if angle_z >=k_angle:
        angle_z-= 2 * k_angle
    return angle_z,angle_y
```

Adjusted data processing: return both z and y angle

Accordingly, the ISR of UART is also modified by allocating values to z_angle and y_angle from the obtained angle list from gyroscope.

```
def UART_Gyroscope_ISR(t):
    global turn_angle
    global count
    global signal
    global command
    global original_angle
    global s_original_angle
    global angle
    global z_angle
    global y_angle
    msg_Gyroscope=UART_Gyroscope.read(UART_Gyroscope.any())
    angle = jy901.DueData(msg_Gyroscope)
    z_angle=angle[0]
    y_angle=angle[1]

    if type(z_angle)==float:
      if count==0:
        #print('count==0')
          original_angle=z_angle
          s_original_angle=z_angle
      count=count+1
    # print(original_angle)
    return
```

Adjusted ISR: get both z and y angle from gyroscope

Then, to reduce work load of Openmv, y angle information is translated to 1-bit down-bridge signal (or "enableCamera" bit) in stm32. After the translation, the bit is sent to Openmv through the established UART connection.

```
global y_angle
print("y_angle:",y_angle)
if(y_angle>10):
    enableCamera=1
    print("sent enable camera.")
    UART_OpenMV.write(str(enableCamera))
```

Send signal bit to Openmv

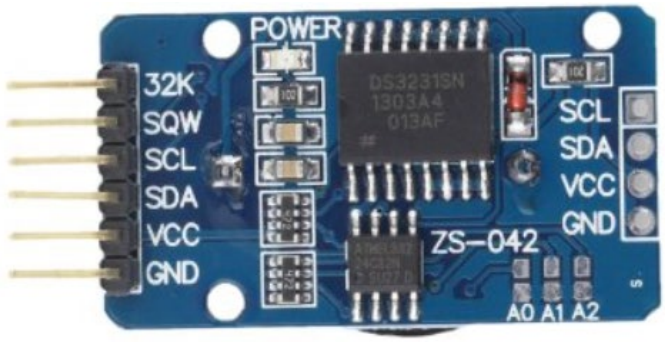
Title	Clock module DS3231		
Group	Image Group	**Week**	Week 15
Name	Shubo Yang	**Date**	6.8

Purpose:

We try to utilize a clock module, DS3231 to return the value of current time for further HC-12 transmission. DS3231 real-time clock (RTC) is a low power, full binary-coded decimal (BCD) clock/calendar, with alarm output.

Procedure:

The DS3231 module has six pins, but only four pins are considered for IIC communication with OpenMV, that are, SCL, SDA, VCC, and GND. The module is shown below:



First, four pins should be connected with OpenMV I2C pins, and I2C port 4 is used. The current time should be set at first. A class named “DS3231” is created for further reference and function call.

```
class DS3231():
    def __init__(self, i2c):
        self.i2c = i2c
        self.setReg(DS3231_REG_CTRL, 0x4C)

    def DecToHex(self, dat):
        return (dat//10) * 16 + (dat%10)

    def HexToDec(self, dat):
        return (dat//16) * 10 + (dat%16)

    def setReg(self, reg, dat):
        self.i2c.writeto(DS3231_I2C_ADDR, bytearray([reg, dat]))

    def getReg(self, reg):
        self.i2c.writeto(DS3231_I2C_ADDR, bytearray([reg]))
        return self.i2c.readfrom(DS3231_I2C_ADDR, 1)[0]
```

The function returns a matrix of time, containing the date, correct time and second. In this way, we can derive the specific time and write it on the UART of HC-12. Consequently, HC-12 can transmit the time.

```
DS3231_time=ds.DateTime()
year=DS3231_time[0]
month=DS3231_time[1]
day=DS3231_time[2]
weekday=DS3231_time[3]
hour=DS3231_time[4]
minute=DS3231_time[5]
print("year",year,"month",month,"day",day,"hour",hour,"minute",minute)
```

Conclusion:
The time is first set for DS3231, and then the time can be read from DS3231 and transmitted by HC-12.

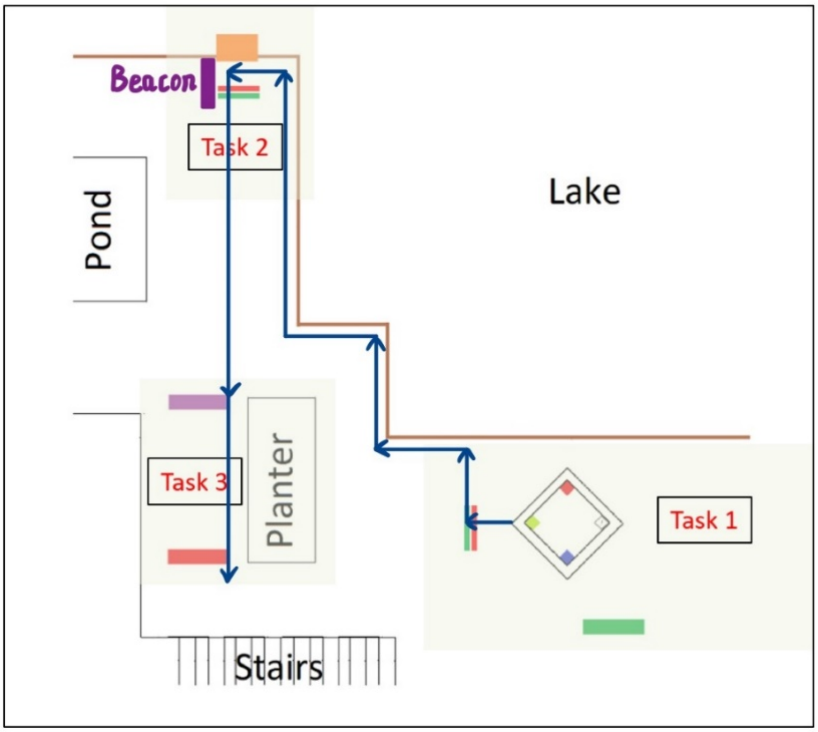
Title	The Route of Feeding Task (final)		
Group	Image Group	**Week**	Week 14&15
Name	Han Qinlin Yang Jingluan	**Date**	2021.6.10

The driving route of the trolley:

The original plan was to let the car go straight to the planter and then turn right after completing the task of color recognition. In order to improve the accuracy of the car driving, we changed the plan: after the car completes the color recognition, it will turn right immediately and go forward to the place 15cm away from the railing measuring by forward ultrasonic sensor and then turn left. After that, the trolley would drive along the railing with a constant distance about 10 centimeters until it reaches the feeding point.

Therefore, the position of color beacon is changed. After the car completes the last turn, the front ultrasonic sensor starts to detect the distance from the beacon, and OpenMV performs color recognition. When the OpenMV recognizes the color of the beacon and the distance measured by the front ultrasonic sensor is less than 20cm, the car stops.

The route plan and the placement of the beacon are shown in the following figure:



The process of return to perform wireless communication would not change.

Processing of the accuracy of ranging information:

During the on-site debugging of the trolley, we found that sometimes the ultrasonic sensor could detect wrong distance, which would make the car execute wrong instructions while it is moving.

In order to eliminate the influence of erroneous data, we take the average of the data measured by the ultrasonic sensor, and take the average for every four data, so that we can eliminate the wrong instructions to the car due to the erroneous data.

Distance adjustment:

In order to ensure that the car can accurately stop at the feeding point, the distance between the trolley and the railing needs to be continuously adjusted during the traveling of the trolley. The specific method is:

Set an appropriate range. When the right ultrasonic detects that the distance between the car and the railing is greater than the set acceptable range, the car will be fine-tuned toward the railing. If the distance is less than this range, let the trolley make fine adjustments away from the railing.

The corresponding code:

```
def stop():
    direction='0'
    print(direction)
    stm_info=str(direction)
    UART_stm.write(stm_info)
    time.sleep(0.5)

def go_forward():
    direction='5'
    print(direction)
    stm_info=str(direction)
    UART_stm.write(stm_info)
    #time.sleep_ms(run_time)
    #stop()

def turn_left():
    stop()
    direction='21090'
    print('turn left',direction)
    stm_info=str(direction)
    UART_stm.write(stm_info)
    time.sleep(2)
    go_forward()
```

```
def turn_right():
    direction='22090'
    print('turn_right',direction)
    stm_info=str(direction)
    UART_stm.write(stm_info)
    time.sleep(5)
    go_forward()

def adjust_left():
    direction='31010'
    print('adjust_left',direction)
    stm_info=str(direction)
    UART_stm.write(stm_info)
    time.sleep(1)
    go_forward()
    time.sleep(0.2)
    go_forward()
    time.sleep(0.2)
    go_forward()
    time.sleep(0.2)

def adjust_right():
    direction='32010'
    print('adjust_right',direction)
    stm_info=str(direction)
    UART_stm.write(stm_info)
    time.sleep(1)
    go_forward()

turns=0
while(True):
    right_total=0
    forward_total=0
    for i in range(4)
        right_distance_i=wave_distance_process(wave_echo_pin_3,wave_tr
        ig_pin_3)
        right_total=right_total+right_distance_i
    right_distance=right_total/4 #average right distance
    print("right:",right_distance)
    for i in range(4)
```

```
    forward_distance_i=wave_distance_process(wave_echo_pin_2,wave_
    trig_pin_2)
    forward_total=forward_total+forward_distance_i
forward_distance=forward_total/4 #average forward distance
print("forward:",forward_distance)
if 0<right_distance<5:
    adjust_left()
elif right_distance>20:
    adjust_right()
elif right_distance>100 or right_distance<0:
    turn_right()
    turns=turns+1
elif 0<forward_distance<10:
    turn_left()
    turns=turns+1
else:
    go_forward()
time.sleep(0.3)
```