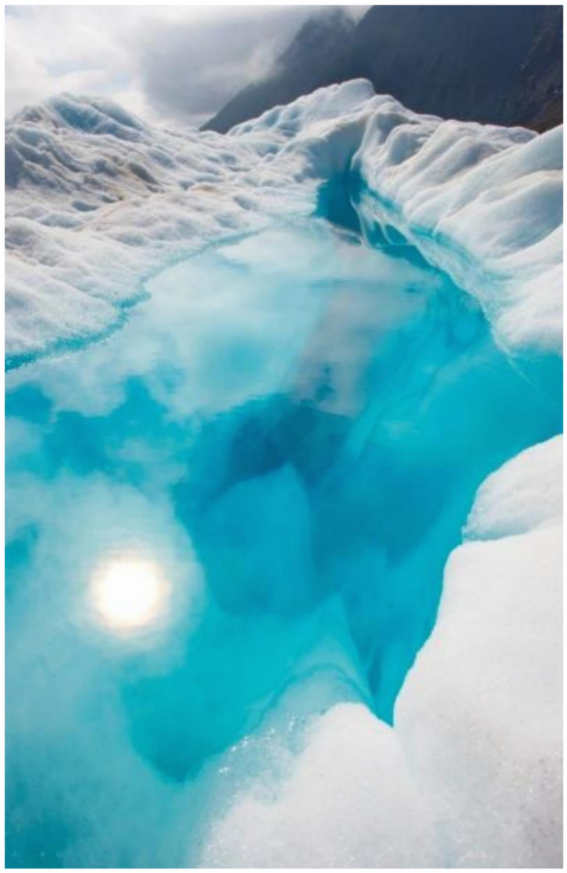


LAB REPORT

Group Name: Walking Eleven
Team Leader: Yiwen Kuang
Group Member: Hanjing Wang, Yifei Li, Erzhizhi Hu, Lanting Huang, Yuanru Yang
Programme: Computer Science and Technology (Joint Program)
University: Ocean University of China & Heriot Watt University
Year: Year 3
Submission Date: 12/01/2024

Yiwen Kuang January 12, 2024

Group Contribution

Name	Contribution	
	Experiment	**Lab Report**
邝镱雯 Yiwen Kuang Contribution Grade: 100%	1.IR Line Following Code 2.IR Line Following Test 3.QR locating and identifying 4.QR Path of Free Travel 5.Camera Line Following Test 6.Project Scheduling 7.Holding meetings	IR Line following QR Locating & Identifying Project Scheduling
胡尔知之 Erzhizhi Hu Contribution Grade: 100%	1.Menu Code 2.Menu Test 3.Remote Control-Camera 4.Remote Control Test 5.Circuit Hardware	Menu Code Circuit Hardware Remote Control
汪函静 Hanjing Wang Contribution Grade: 100%	1.Camera Line Following 2.Camera Line Following Test 3.IR Line Following Test 4.Maze Code 5.Body Design	IR Line Following Camera Line Following Summary
李奕菲 Yifei Li Contribution Grade: 100%	1.Robot Chassis & IO Expansion 2.Free Travel Code 3.Free Travel Test 4.IR Line Following Test 5.Recording meetings	Free Travel Body Construction Holding Meetings
黄兰婷 Lanting Huang Contribution Grade: 100%	1.Body Construction 2.Remote Control-Basic Movement 3.Log 4.IR Line Following Test 5.Free Travel Test	Remote Control Body Construction Log
杨元儒 Yuanru Yang Contribution Grade: 80% (often be absent on meeting and test, lose 10%) (do the less thing, lose 10%)	1.Remote Control Code 2.Fritzing Circuit Diagram	Fritzing Circuit Diagram

CONTENTS

Smart Functional Robotic Car Project

01 Menu - Robot Option (Eizhizhi Hu)

Initial Idea

Loops and recursions were used to implement this part.

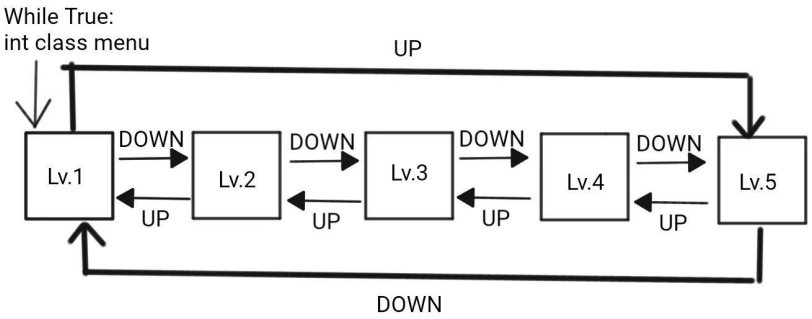


FIGURE 1.1: (initial thought)

When the menu state is at a level function (Lv.1, for example), we can invoke the adjacent level function by pressing button “DOWN”(Lv.2) or “UP”(Lv.5), and this enables the choice of functions.

Problem Encountered

- We found that in this way, there is no exit for the loop that is invoked, so the shitty Raspberry Pi might be fucked up if we looped too many times.

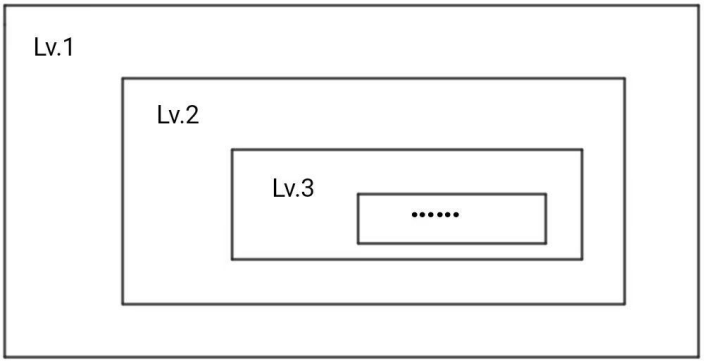


FIGURE 1.2.1: (loops with no return)

-Since the Raspberry Pi can only use Python, we cannot use the cursor system that LCD has, so how to display the menu system on LCD, how to let the users know which function mode they are selecting is another problem.

- The remote function entered via the menu function does not work.

- After invoking the function mode of the robot, we found that once we return to the menu, and invoke the function again, an error says the PWM pin has been used will occur.

- How to add parameters like speed, distance is also a problem, and how to push them into the function modes.

Problem Solved

- To avoid Raspberry Pi being fucked up by the loop, we will return the level function when it's going to invoke Lv.1 function, and then restatement the menu class.

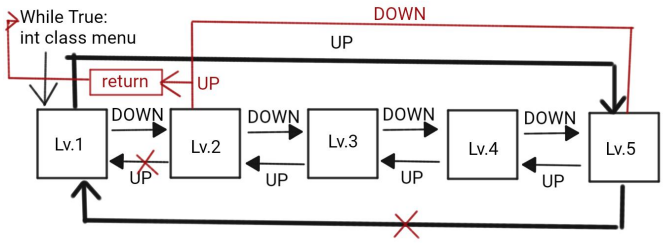


FIGURE 1.3.1: (improved thought)

- The menu system has been divided into few levels, so we can by printing messages like shown in FIGURE1.3.2. And the output on LCD will be in FIGURE 1.3.3.

```
lcd_text(">IR", LCD_LINE_1)
print(">IR")
lcd_text(" Camera", LCD_LINE_2)
print(" Camera")
```

FIGURE 1.3.2: (the printing code)

FIGURE 1.3.3: (output on LCD)

- Every time we push the "UP" or "DOWN" button, the output of new level that be invoked will overwrite the last output. In this way, we can implement the selecting function.

- We first thought it didn't work because the IR sensor model is broken, so we cannot use the controller to control the robot, but then we found that every time we invoke the remote function, we didn't clean the GPIO pins that have been configured on the Raspberry Pi, so some pins were multiplexed. So now every time we invoke a function, GPIO.cleanup() will be used to clean the pin setups. (FIGURE 1.3.4)

- This problem is quite similar with the former, since the GPIO.cleanup() cannot clean the PWM settings, we need to write a new code to clean the PWM settings. (FIGURE 1.3.5)

```
GPIO.cleanup()
ircontrol.start(speed)
main()
return
```

FIGURE 1.3.4: (invoking code)

```
if (GPIO.input(CANCEL) == 0):
        L_Motor.stop()
        R_Motor.stop()
        L_Motor = 0
        R_Motor = 0
        pwm.setPWM(1, 0, 0)
        pwm.setPWM(2, 0, 0)
#           del pwm
        GPIO.cleanup()
        pylirc.exit()
        return
```

FIGURE 1.3.5: (clean code)

- We use a menu level as the setting option, and when user select this option, a new class similar with the menu class will be initialized, and the setting level will be displayed. (FIGURE 1.3.6)

- And we can create global variables that will be sent to the function modes, we can use the setting class to change the value of them (FIGURE 1.3.7).

```
def lv5(self):
    lcd_text(">settings", LCD_LINE_1)
    print(">settings")
    lcd_text("", LCD_LINE_2)
    print("")
    time.sleep(0.2)
    while True:
        if (GPIO.input(UP) == 0):
            self.lv4()
            return
        elif (GPIO.input(DOWN) == 0):
            return
        elif (GPIO.input(CONFIRM) == 0):
            while True:
                set = setting()
                set.setting_lv1()
                if (GPIO.input(CANCEL) == 0):
                    return
```

FIGURE 1.3.6: (invoking code)

```
def change(self,para):
    global set_status
    global cur_speed
    global cur_light
    global cur_log
    global cur_distance
    global cur_colour

    if set_status == 1:
        cur_speed = para
        return
    elif set_status == 2:
        cur_light = para
        return
    elif set_status == 3:
        cur_log = para
        return
    elif set_status == 4:
        cur_distance = para
        return
    elif set_status == 5:
        cur_colour = para
        return
```

FIGURE 1.3.7: (changing code)

Final Version

So now the full system can be display like this:

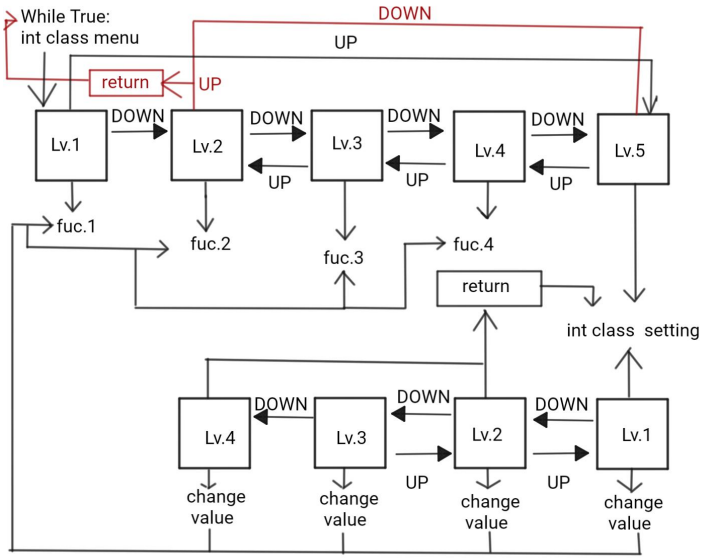


FIGURE 1.4(menu system)

02 Fritzing Circuit Diagram (Yuanru Yang)

Initial Idea

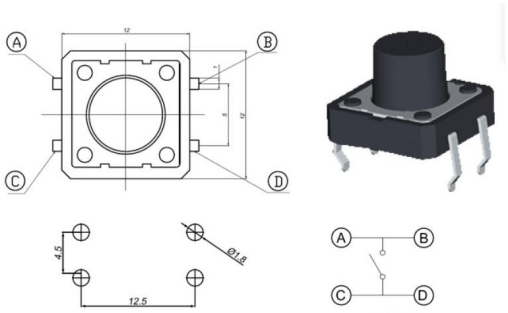
To design a wiring diagram, I first need to build up my knowledge base, such as learning to use Fritzing software. I need to understand the various interfaces of the T-expansion board and the LCD1062 display, and become familiar with the wiring methods of the breadboard, the use of four-pin buttons, etc. What I need to do is to connect the pins of the T-expansion board to the pins of the LCD1602 correspondingly through a breadboard, and also reserve the pins for other functions on the T-expansion board. Check the pins to make sure that they aren't being used for something else, such as the wheels or ultrasound. Additionally, I need to design the button connection indication and its function for the LCD1602.

Problem Encountered

When connecting the T-expansion board and the Raspberry Pi, I found that the pin names were different. I solved this problem by finding a pin correspondence table.
When testing the button, the LED did not light up. It was found that this issue was due to a lack of understanding of the internal circuit diagram of the button.

Problem Solved

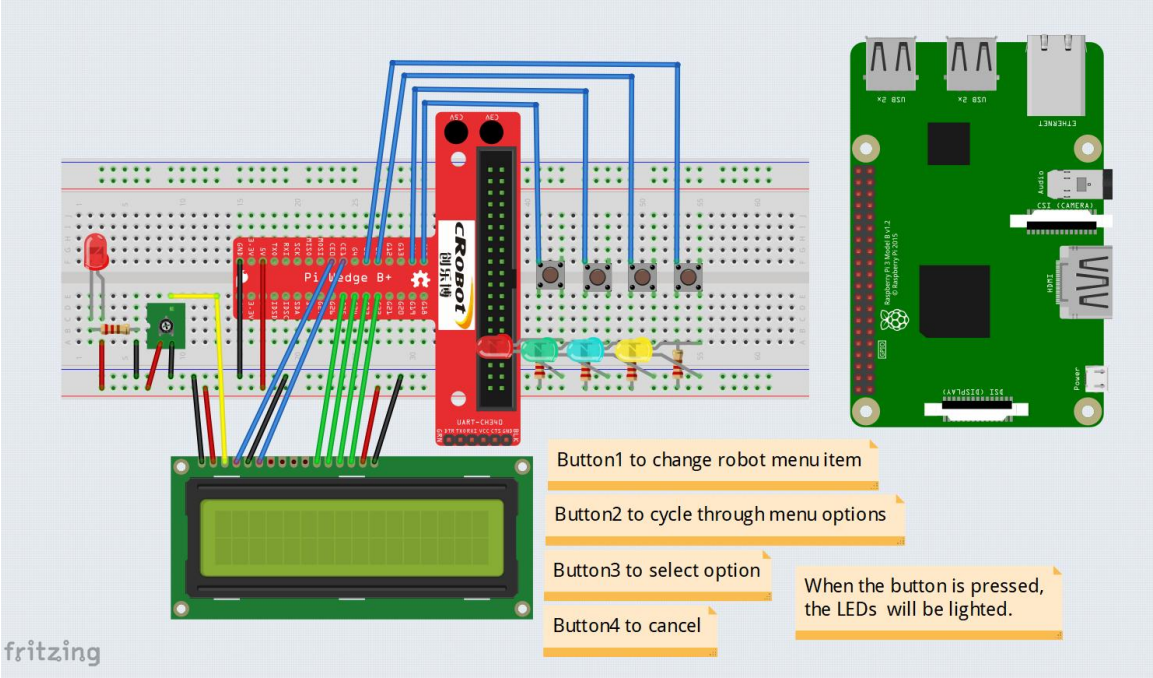
FIGURE 2.1: Button Schematic



By reading the schematic diagram of the button, I understand its wiring method. "A" and "B" represent the two external terminals of the switch that you would connect to your circuit. The line between "A" and "B" indicates that when the switch is pressed, these terminals are connected together, allowing current to flow through the switch. "C" and "D" represent the internal mechanism of the switch that closes the circuit. The line with the arrow pointing to "D" shows that the connection is normally open (NO), and it will only close when the switch is actuated (pressed).

Final Version

FIGURE 2.2: Wiring document



The green board on the right side of the image is a Raspberry Pi single-board computer, which is connected to the T expansion board on the left via a ribbon cable.

Four push buttons are wired to the GPIO pins through the Pi Bridge B+ board. Each button has a designated function:
- Button1 is used to change robot menu items.
- Button2 cycles through the menu options.
- Button3 is for selecting an option within the menu.
- Button4 acts as a cancel button.

There are also several LEDs (light-emitting diodes) connected to the GPIO pins, which are designed to light up when their corresponding buttons are pressed, providing visual feedback. When the circuit is operational, the red LED on the left should be constantly on.

03 Circuit Hardware (Eizhizhi Hu)

Initial Idea

We connect the wires according to the code as shown in FIGURE 3.1.

Problem Encountered

- After we connect the wires and testing the code, an error said I2C address is wrong.
- The LCD is garbled.
- The LED doesn't shine.

Problem Solved

- We found out that the GPIO pin0, 1, 2, 3 are used by I2C, but we used it to connect the LCD pins, so we changed it as shown in FIGURE 3.2.1.

- First, I thought it was the T-board broken, then, it turned out it was my menu code having some mistakes.
- We connected our LED in a short circuit. (FIGURE 3.2.2)

```
LCD_RS = 2
LCD_E = 3
LCD_D4 = 4
LCD_D5 = 5
LCD_D6 = 6
LCD_D7 = 17
LED = 15
UP = 7
DOWN = 8
CONFIRM = 10
CANCEL = 14
F_LIGHT = 12
F_LED = 16
```

```
LCD_RS = 9
LCD_E = 11
LCD_D4 = 4
LCD_D5 = 5
LCD_D6 = 6
LCD_D7 = 17
LED = 15
UP = 7
DOWN = 8
CONFIRM = 10
CANCEL = 14
F_LIGHT = 12
F_LED = 16
```

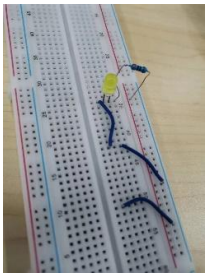


FIGURE 3.1:(pin code) FIGURE 3.2.1: (new pin code) FIGURE 3.2.2: (new)

Final Version

We finished our circuit hardware in this way:

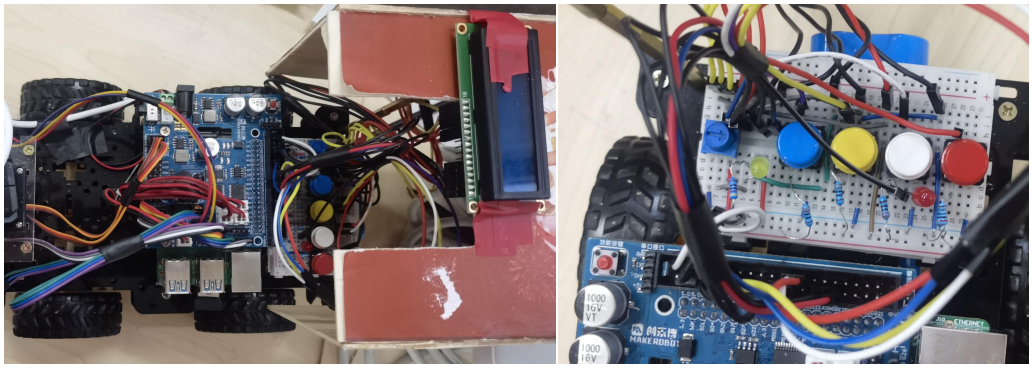


FIGURE 3.4:(full circuit hardware)

04 Body Construction (Yife Li + Lanting Huang)

Initial Idea

In this experiment, we used the Spring Festival topic for design. Glue through the wooden boards to form a shell. The design drawings of each surface of the shell are shown in the figure below.

FIGURE 4.1: Back **FIGURE 4.2: Up**

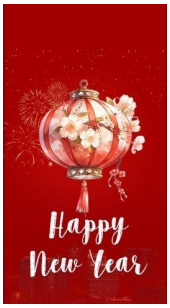


FIGURE 4.3: Left FIGURE 4.4: Right

Problem Encountered

- Due to the uneven edges of the cut wooden board, it was challenging to glue during assembly, requiring continuous manual fixation.
- As the edges of the plank cuts are not parallel, a gap is left where the two planks join.
- When setting the appearance, we used electronic software design, but the exported dimensions did not align accurately during printing.

Problem Solved

- To secure the wooden board frame, we pasted small triangular wooden blocks on the edges to make the frame more robust.
- For the parts with gaps, wooden blocks were glued into the gaps to pull and tighten the separated sections.

FIGURE 4.

- Printed multiple copies until the dimensions were suitable.

Final Version

We managed to cut the boards to the right size and successfully glued them together. And the wooden boards are decorated by printing and pasting, and the internal circuits can be adjusted through drilling and their design. Successfully kept the internal structure from being exposed.

FIGURE 1.4:

05 IR Line Following (Hanjing Wang + Yiwen Kuang)

Initial Idea

Using a set of infrared sensors evenly distributed along the robot's chassis, the placement distance depends on the width of the line. The number of sensors is determined by the complexity of the line-following task. For this task, we utilized three sensors (left, center, right) to ensure detection of three states: whether the robot is on the line, to the left of the line, or to the right of the line, without deviating too much.

Infrared sensors can detect reflected light on the ground. When a sensor is positioned over a black line, the reflection is weaker; whereas, when the sensor is over a white surface, the reflection is stronger. This contrast enables the sensor to identify the line. It is essential to adjust the sensitivity of the sensors based on the color of the line. Additionally, establishing a threshold allows determining whether the robot is on the line based on the intensity of reflected light. For example, if the sensor reading is below a certain threshold, the robot can be considered to be on the black line.

Based on the combination of sensor readings, a line-following strategy is devised for the robot. For instance, if the state is 011 or 001, indicating the robot is to the right of the line, the robot may need to adjust its direction to the left.

The IR sliding potentiometer is used to adjust sensitivity. Clockwise rotation reduces sensitivity, while counterclockwise rotation increases sensitivity. Adjust it until the background color (reflection intensity) and the black tape (only one light is on - the sensor has two lights: one is the switch indicator, and the other is the power indicator; black absorbs light and does not reflect light, but there is a thin layer of plastic film on the black tape that reflects light, so adjusting sensitivity is crucial) can be well distinguished.

When adjusting sensitivity, it is also possible to make a more intuitive judgment through the printed output of 'left' and 'right'. After running the code, the car will not move until the red "function button" on the board is pressed.

Problem Encountered

I. Pin misused: The confusion between the GPIO Broad mode and BCM (System on Chip) mode in the code.
II. “Naughty” behaviour: Car could not find the line well when the program code is logically right.
III. Variability in right angles: When the vehicle encounters right-angle turns, the IR sensors often output 110 or 011. Conventional turns, however, may not guarantee the vehicle's successful negotiation of right-angle turns. Simultaneously, there is a significant risk of the car veering out of right-angle turns, leading to an IR sensor output of 000.
IV. Ambiguous numeric fluctuations: Even with LED indicators on the IR sensor providing a visual cue for the current output state, observing the sensor, mounted at the bottom of the vehicle, can be challenging. Moreover, some IR sensors exhibit subtle changes in brightness, making it difficult to discern variations clearly; they might only exhibit slight increments or decrements. Adjusting the sensitivity of IR sensors to a more precise numeric value is crucial.

Problem Solved

I. By testing each pin, the mode was finally established: Broad mode. That is, the Physical pin 19 is equivalent to GPIO10, not GPIO 19.
II. The vehicle determines its operational state by evaluating the values outputted through IR sensors, hence the precise accuracy of IR sensor sensitivity is of paramount importance. To calibrate the sensitivity of IR sensors, I have developed code specifically for printing the numerical output of three IR sensors. It is imperative to verify the correctness of the sensitivity of these three IR sensors before each debugging session for IR line-following code.
III. When the IR sensor outputs 000, the vehicle initiates a reverse motion. By iteratively maneuvering backward, the car can adjust its orientation, ultimately navigating through right-angle turns. However, this process often requires multiple cycles of forward and backward movements for successful completion.

Final Version

The value transmitted by the IR sensor determines the state of motion of the car in the following cases:
(1) 010: Car proceeds normally straight ahead;
(2) 110: Car made a slight left turn;
(3) 100: Car makes a more substantial left turn;
(4) 001: The car makes a relatively large right turn;
(5) 000: The car rotates slowly in place;
(6) 111: Maintain the original motion.

06 Free Travel (Yifei Li)

Initial Idea

The primary function of Free Travel is to navigate obstacles while capturing and storing randomly distributed QR codes on the walls. The obstacle avoidance module is initially planned to combine infrared sensors and ultrasonic sensors. In the previous semester, we used infrared sensors for line following. This semester, infrared sensors are not only placed under the car but also on the top with two sensors. The IR sensors on the sides of the car effectively prevent side collisions. The key is to integrate QR code recognition and obstacle avoidance modules, considering the use of multi-threading.

Problem Encountered

- Device Issues:

- Low battery or long working hours of the car may cause device lag during remote WiFi connection.
- Unstable camera connection after collisions leads to errors like VIDIOC_DQBUF: No such device Unable to stop the stream: No such device.
- Due to limited interfaces, LED needs to be connected, and initially planned not to use side IR sensors. However, side collision issues were severe during testing. At first, I attempted to use the obstacle avoidance logic from the previous semester, letting the ultrasonic sensor continuously detect to reduce side collisions. Due to the slow response of the Raspberry Pi, a significant delay prevented the ultrasonic sensor from nodding at a small angle continuously.
- Side IR sensing range is not large, still posing a risk of side collisions. Also, easily disturbed after installing the casing.
- Multiple collisions led to the breakage of the connection between the ultrasonic sensor and servo motor.

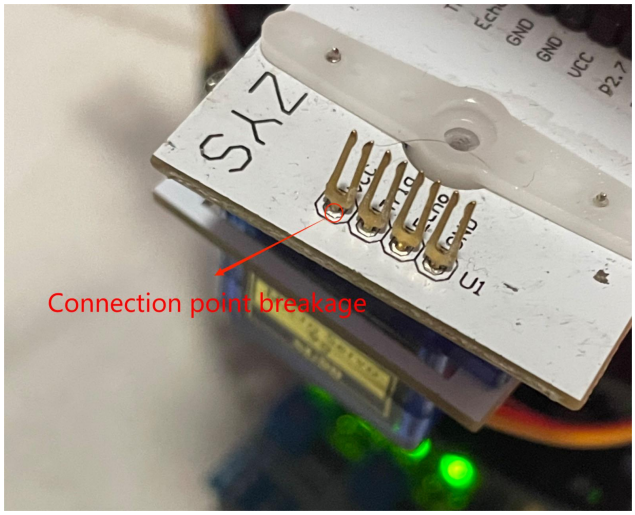


Figure 4.1

- Human factors: Ultrasonic sensor and servo motor went missing, possibly taken by someone.

Code Issues:

- The 'q' key needs to be pressed to exit the QR code recognition, and placing it in a ‘loop’ prevents self-exit, causing movement issues.
- If the program is terminated by pressing the stop key, there is a risk of high-speed rotation of the wheels.And high speed rotation of wheels may cause detachment.
- Steering issues during initial obstacle avoidance, tending to turn towards the direction of obstacles due to incorrect settings of AIN and BIN high-low levels for different directions.

```
def t_right(speed,t_time):
        L_Motor.ChangeDutyCycle(speed)
        GPIO.output(AIN2,False)#AIN2
        GPIO.output(AIN1,True) #AIN1

        R_Motor.ChangeDutyCycle(speed)
        GPIO.output(BIN2,False)#BIN2
        GPIO.output(BIN1,False) #BIN1
        time.sleep(t_time)
```

Figure 4.2

- Initially used the imwrite function for image storage, but an error occurred - AttributeError: module 'cv2' has no attribute 'imread,' indicating that Cv2 module does not have imwrite.

Problem Solved

- Device problems solved:

- Lagging problem solved:
  - Raspberry Pi connected to a charging or power line significantly reduces VNC lag issues.
  - Imported QR_Detect in the obstacle avoidance code to reduce code redundancy.
  - Removed PyCharm and directly used Python IDE Thonny for script execution.
- After multiple experiments, I found collisions caused unstable connections with the camera.
- Side collision problem solved:
  - Connected two IR sensors to the sides using pins 12 and 16.
  - Adjusted IR sensitivity multiple times.
  - Used nuts at the corners of the top plate to lift the casing, reducing interference with IR sensors.

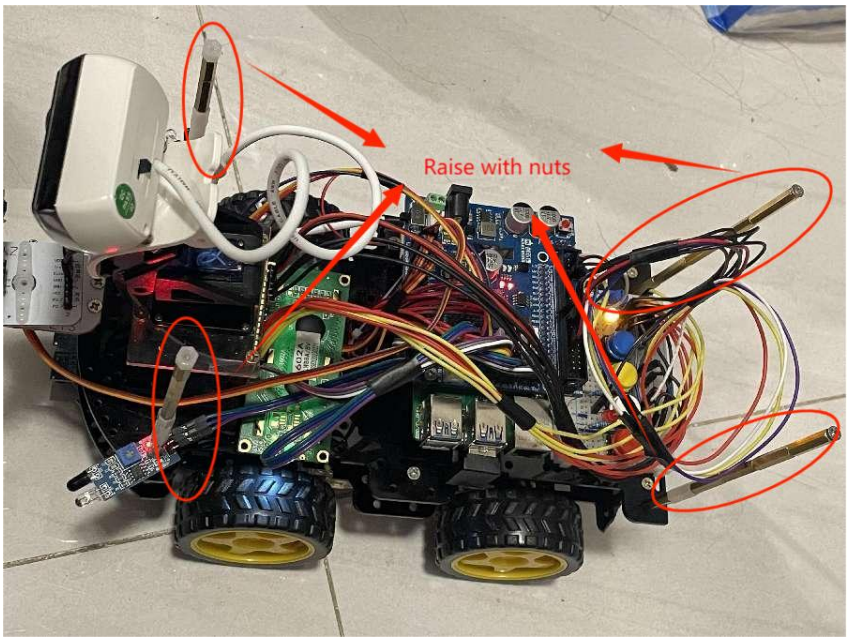


Figure 4.3

- Utrasonic sensor problem solved:

・Plan 1： Attempted to re-weld the fork-shaped metal component, but it was challenging.

・Plan 2： Used pin headers instead of forked pins, inserted into a female header. However, the pins and headers inside the female header would block each other.

Figure 4.4

・Final solution (before sensor was stolen): Directly inserted the ultrasonic sensor into the servo motor board's socket and welded it in place.

- Ultrasonic sensor and servo motor went missing, and we used last semester’s obstacle avoidance servo motor as a replacement, achieving perfect results.

-Risk: The replacement servo motor is easily knocked off in case of a frontal collision.

Coding problem solved:

- Used multi-threading, as PyCharm was removed, allowing the Raspberry Pi to run multi-threaded programs.

```
thread1 = threading.Thread(target=detect)
thread2 = threading.Thread(target=ultrasound)

thread1.start()
thread2.start()

try:
        thread1.join()
        thread2.join()
```

Figure 4.5

- Added try-except KeyboardInterrupt to exit the program using Ctrl+C.
- Modified AIN and BIN True and False values for forward, backward, left, and right movements.

```
def t_up(speed,t_time):
        L_Motor.ChangeDutyCycle(speed)
        GPIO.output(AIN2,False)#AIN2
        GPIO.output(AIN1,True) #AIN1

        R_Motor.ChangeDutyCycle(speed)
        GPIO.output(BIN2,True)#BIN2
        GPIO.output(BIN1,False) #BIN1
        time.sleep(t_time)
```

False

True

Figure 4.6

- Checked and corrected the spelling mistake in ‘ imwrite’ , which was accidentally written as ‘imwirte’.

Final Version

Although the process was somewhat convoluted, the final solution remained essentially consistent with the initial idea. Side IR sensors and ultrasonic sensors are used together for obstacle avoidance, and multi-threading is employed to simultaneously capture QR codes. The ultrasonic sensor and servo motor utilize leftover components from the previous semester, resulting in a satisfactory outcome.

07 Remote Control (Eizhizhi Hu + Lanting Huang)

Initial Idea

Our initial goal was to control motors and a servo motor using a Raspberry Pi, enabling the movement of a small car and adjustment of the camera's direction. Simultaneously, we aimed to implement remote control of the car through an infrared remote control.

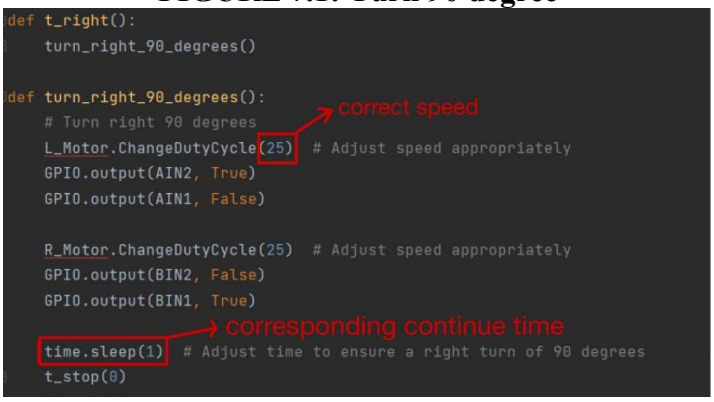
Problem Encountered

- During our initial run, we faced challenges in hardware connections. Incorrect wiring of motors, servo, and the infrared receiver, along with insufficient power supply, impeded the normal functioning of the program.
- Due to problems importing third-party libraries such as cv2, and pylirc, we encountered installation errors and version incompatibility.
- The code encompassed motor control, servo control, and infrared remote control event handling, leading to logic errors and inadequate functionality in certain modules.
- At the beginning, the coding of the buttons on the remote control was not matched with the coding in the system, resulting in incorrect button response and no response.
- Can't stop turning left and right, keeps spinning in place.

Problem Solved

- We ensured correct wiring of motors, servo, and the infrared receiver, along with adequate power supply, and verified pin configurations to match the code.
- All required libraries were reinstalled, ensuring correct versions that aligned with the code's compatibility requirements.
- The logic pertaining to motor control, servo control, and infrared remote control event handling underwent careful review and refinement to eliminate errors and enhance functionality.
- Check and confirm the codes corresponding to different buttons and the distance that can be confirmed by infrared remote sensing.
- By controlling the speed and time of the circle, we can ensure that the left and right turns can turn exactly 90 degrees and then stop, instead of just spinning in place. Take the code for turning right as an example, as shown in the figure above.

FIGURE 7.1: Turn 90 degree



Final Version

Following the resolution of hardware connection issues, proper installation of dependencies, and refinement of code logic, the final version successfully achieved the following functionalities:

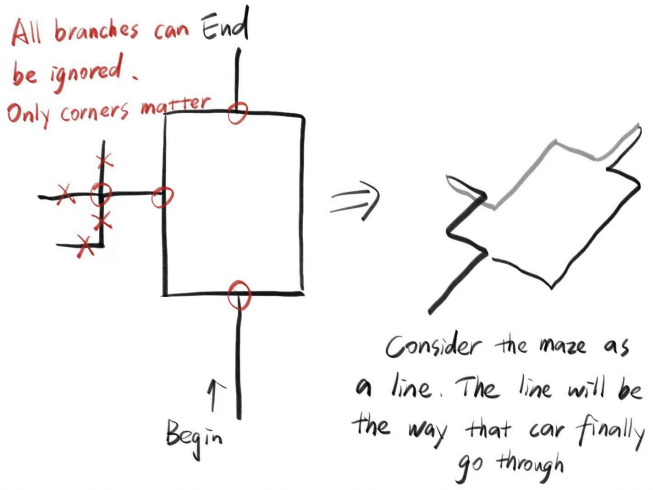
- Remote control of the car using an infrared remote control for actions like forward, backward, left turn, right turn, and stop.
- Speed adjustment of the car by controlling the motor speed.
- We can also use the remote control to control the servo of the camera to be able to rotate both horizontally and vertically, this enables the camera to capture objects in different positions and angles, and we also use the remote control to control the camera to take pictures and store the captured photos to a designated path.

08 Camera Line Following (Hanjing Wang)

Initial Idea - Version 1 - Maze:

Initially, there was a contemplation of measuring the length of the path traversed by the vehicle through time recording and devising an algorithm to draw the corresponding maze map. However, due to concerns regarding potential glitches in the car's movement and the instability of its speed, this approach was discarded. To ensure the thorough exploration of the maze, an alternative perspective was adopted, treating the maze as a three-dimensional construct represented by the illustration below.

FIGURE 8.1: Maze ideas



Initial Idea - Version 2 - Maze:

On the Basis of Version 1, several additional issues necessitate resolution: How to detect a loop? If the paths at the current intersection have been traversed, how does the vehicle return to the preceding intersection? Once the vehicle turns back to the previous intersection, how should the stored left and right turn states be modified? In response to these three challenges, I have devised the following solutions:

- Utilizing a list to store each corner (simulating a stack structure), all corners where all feasible paths have been traversed are deleted using the pop method. When the list is not empty, determine if the bottom corner of the stack forms a loop with the

current intersection (whether the shapes correspond), as illustrated in the algorithm depicted in the figure below.

FIGURE 8.2: Implementation Code

```
def is_circle(corner):
    # check if meets the end of a circle
    global corner_stack
    if corner_stack:
        if corner_stack[0].corner_type == RIGHT_T and corner.corner_type == LEFT_T:
            return True
        elif corner_stack[0].corner_type == LEFT_T and corner.corner_type == RIGHT_T:
            return True
        elif corner_stack[0].corner_type == T and corner.corner_type == LEFT_T:
            return True
    return False
```

- Each corner stores the method to return to the previous corner. By examining the draft as illustrated in the figure below, it is evident that the direction of the car's rotation is related to the method of returning to the previous corner. Leveraging this, an algorithm can be formulated: represent LEFT, RIGHT, and STRAIGHT using integer values, where a left turn is denoted by subtracting 1, straight or right turn by adding 1, and finally performing modulo 3 division. (FIGURE 8.3)

As above, based on the simulation of the various possible scenarios, the corresponding pattern can also be found, as shown in the FIGURE 8.4.

FIGURE 8.3: Algorithm ideas

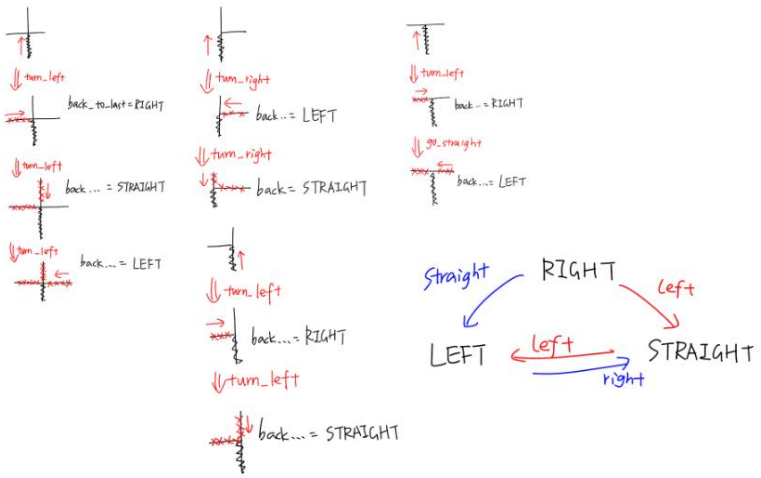
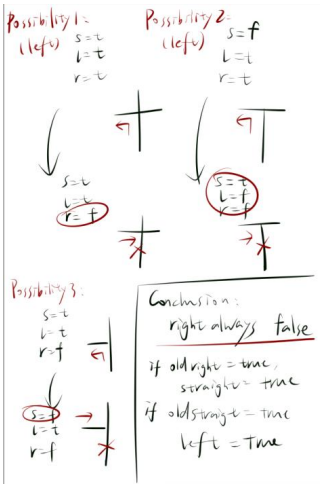


FIGURE 8.4: Algorithm ideas



Based on the above algorithm, the following code can be written.

FIGURE 8.5: Implementation Code

```
# turn left first
if corner.left:
    # go left is possible
    # change the corner's possible directions
    temp_straight = corner.straight
    temp_right = corner.right
    corner.left = temp_straight
    corner.right = False
    corner.straight = temp_right
    if corner.to_last_corner == STOP:
        corner.to_last_corner = RIGHT
    else:
        corner.to_last_corner = (corner.to_last_corner - 1) % 3
    turn_left()
elif corner.right:
    # go right is possible
    # go left is already impossible
    # change the corner's possible directions
    corner.left = False
    corner.right = corner.straight
    corner.straight = False
    if corner.to_last_corner == STOP:
        corner.to_last_corner = LEFT
    else:
        corner.to_last_corner = (corner.to_last_corner + 1) % 3
    turn_right()

elif corner.straight:
    # go straight is the only way
    corner.straight = False
    corner.finished = True  # finish travelling the corner
    corner.to_last_corner = (corner.to_last_corner + 1) % 3
    turn_straight()
```

Initial Idea - Camera Line Following

In coursework02, the identification of intersection directions has been successfully accomplished, providing a foundation for controlling the car's operational state. As the modified paths do not involve particularly complex intersections, there is no need to prioritize different directions. The primary focus is on correctly identifying and navigating through right-angle turns.

Problem Encountered

I. In the case of a maze containing loops, version1 of the maze would result in the car becoming trapped within the loop, unable to navigate out. In scenarios where loops are absent but numerous corners exist, the car would traverse a longer path, consuming more time.

II. When the car encounters a right-angle turn, the camera has already surpassed the turn, and the display indicates the absence of a path.

III. After making a turn, the car tilts, and continuing straight may cause it to progressively deviate from the intended line.

IV. During code execution, the following errors may be encountered:

FIGURE 8.6: Error message

```
Unable to stop the stream: Invalid argument
```

V. While running the car, it was noticed that the program was stuck, causing the car to turn early or go over corners.

VI. When the code was finally run, the car could not be stopped at the end of the line.

Problem Solved

I. When the car identifies a turn, a predefined action is implemented to allow the car to precisely navigate through the right-angle turn.
II. When the line deviates from the center of the screen, corresponding adjustments are made with left or right turns to fine-tune the car's orientation, ensuring it returns to the straight line. This method can also achieve line following through non-right-angle turns by continuously altering the car's orientation.
III. In response to issue 4, it was ultimately identified as an interface number error with the camera. Appropriate modifications were made (changing 0 to 1 or 1 to 0) to resolve the problem.
IV. Regarding issue 5, the smoothness of the car's operation is contingent on both the battery level and the complexity of the code. Ensuring an ample supply of power is crucial. Additionally, modifications were made to the code. The original "identify_path" code invoked a relatively complex library for linear fitting, with the original code shown in the figure below.

FIGURE 8.7: Original code

```
def merge_similar_lines(lines, angle_threshold=10):
    """
    Merge similar lines
    :param lines: (array)
    :param angle_threshold:
    :return: (array)
    """
    # calculate the angle of each line
    angles = np.degrees(np.arctan2(lines[:, 3] - lines[:, 1], lines[:, 2] - lines[:, 0]))

    # if the samples are not enough, skip this function
    if len(lines) <= 2:
        return lines

    kmeans = MiniBatchKMeans(n_clusters=2, batch_size=600).fit(angles.reshape(-1, 1))
    labels = kmeans.labels_

    # merging straight lines in the same cluster
    merged_lines = []
    for label in np.unique(labels):
        cluster_lines = lines[labels == label]
        merged_line = np.mean(cluster_lines, axis=0)

        merged_lines.append(merged_line)

    return np.array(merged_lines)
```

This code fits all detected lines into two, leading to a certain degree of lag in the car's operation. Deleting this section of code has no impact on the actual functionality; it suffices to select one line from the array when determining the straight line. Consequently, this section and its corresponding calls were ultimately removed.
V. In response to issue 6, conditional statements were added in the code to handle scenarios where no straight line is detected. When no straight line is identified, the status 'stop' is returned, causing the car to halt its movement. The modified code is illustrated in the figure below.

FIGURE 8.8: Changed code

```
if lines is not None:...
else:
    print('stop')
    return 'stop'
```

Final Version

Detecting straight lines through the camera involves several scenarios:
- If a relatively short straight line is detected, the status 'stop' is returned.
- If a relatively long straight line is detected, the status 'straight' is returned.
- If a left turn intersection is detected, the status 'left' is returned.
- If a right turn intersection is detected, the status 'right' is returned.

Based on the returned status, the movement of the car is adjusted accordingly. Initially, it is necessary to calculate the centroid of the line coordinates to determine if the car is on the line (and whether the body state needs adjustment). If the received status is 'straight' and the centroid on the x-axis is left-shifted or right-shifted, the body state is adjusted accordingly (slightly left or right turn). The code is shown in the figure below.

FIGURE 8.9: Final version code 1

```
if result == 'straight':
    if 280 > moments_x > 120:
        go_forward(final_speed, 0)
    elif 280 < moments_x < 400:
        go_right(20, 0)
    elif 120 >= moments_x > 0:
        go_left(20, 0)
```

To smoothly navigate through right-angle turns, additional variables, left_count and right_count, are introduced. These variables are used to record the occurrences of the status 'left' and 'right,' respectively. If left_count or right_count exceeds a predefined threshold, the turning logic is triggered. Taking a left turn as an example, the car initially moves straight for a few seconds (the speed and duration need adjustment based on actual conditions) to ensure the car is positioned correctly for the turn. After the straight movement, the car performs a left turn in place until the camera captures a straight line (status received as 'straight'). At this point, the turn is completed, and the car resumes normal line following. The implementation code is shown in the figure below.

FIGURE 8.10: Final version code 2

```
elif result == 'left':
    left_count += 1
    if left_count >= threshold:
        go_forward(30, 1.2)
        while True:
            go_left(turning_speed, 0)
            if result == 'straight':
                break
```

09 QR Locating & Identifying (Yiwen Kuang)

Initial Idea

My code uses the OpenCV library and the PyZBar library to achieve QR Code identification. The recognition process involves image preprocessing, contour detection, and decoding the QR code information.

I. Image Preprocessing

Grayscale Conversion: The original image is converted to grayscale using `cv2.cvtColor()`.
Median Blur: A median blur is applied to the grayscale image using `cv2.medianBlur()` to reduce noise and smoothen the image.
Otsu's Thresholding: Otsu's thresholding is employed on the blurred image to create a binary image with clear boundaries between the foreground and background.
4. Erosion: The binary image is further processed with erosion `cv2.erode()` to enhance the integrity of the contours.

II. Contour Detection and QR Code Localization

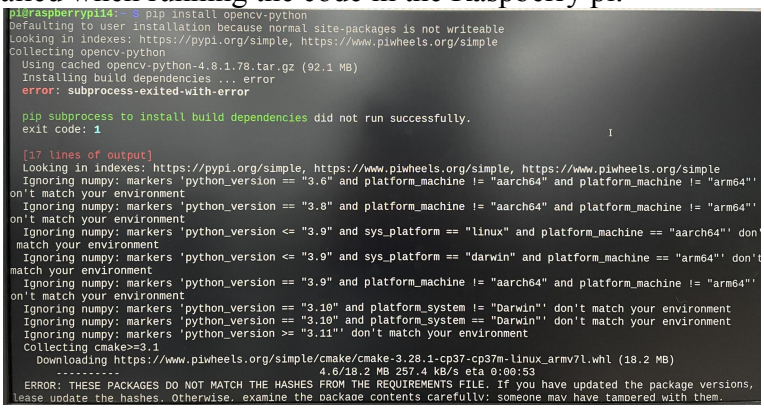
Find Contours: The `cv2.findContours()` function identifies contours in the eroded image.
Hierarchy Analysis: The hierarchy of contours is analyzed to identify potential QR code patterns. Specifically, contours with two levels of hierarchy are considered.
Filtering and Validation: Contours are filtered based on their lengths, and a list of validated contours representing potential QR code positions is obtained.
Centroid Calculation: The centroid of each validated contour is calculated using the moments of the contour.
Drawing Location Triangle: A triangle is drawn around the detected QR code position based on the centroid and contour area.

III. QR Code Decoding

Capture from Camera: The code initializes a video capture using `cv2.VideoCapture(0)` to capture frames from the default camera.
Decode and Display: The `decode_display(image)` function is used to decode QR codes from the captured frames. It utilizes the PyZBar library to extract QR code information and draw bounding boxes around the codes.
Feedback and Display: The decoded QR code information, along with bounding boxes and polygonal outlines, is displayed on the live camera feed. The information is also printed to the console.

Problem Encountered

- Opencv was failed when running the code in the Raspberry pi.



- Error: "AttributeError: module 'numpy' has no attribute 'int'. `np.int` is a deprecated alias for the built-in `int`. To avoid this error in existing code, use 'int' directly. This will

not alter any behavior and is safe. When replacing 'np.int', you may want to specify precision, for example, 'np.int64' or 'np.int32'. If you wish to see the current usage, refer to the release notes link for more information."
- Error: error: (-215:Assertion failed) !image.empty() in function 'cv::imencode'

Problem Solved

- We changed a bunch of mirror sources. It should succeed theoretically but the results disappointed us. Then we downloaded the Pycharm to pip opencv in it. Gradually, we found the internal storage was unable to support the Pycharm IDE. The program was so easy to be stuck that we finally download the orignal opencv fial through Internet in somehow instead of using terminal.
- In the current configuration environment, uninstall NumPy first, and then change the NumPy version to 1.22.

```
pip uninstall numpy
```

```
pip install numpy==1.22
```

```
Proceed (Y/n)? Y
  Successfully uninstalled numpy-1.26.1
(venv) PS C:\Users\Lenovo\PycharmProjects\pythonProject> pip install numpy==1.22

[notice] A new release of pip available: 22.3.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

- When cv2 reads the image, the path of the image cannot contain Chinese or spaces, and can only be English or numbers, otherwise an error will be reported!

Modification

The QR code recognition function can be invoked by the "free travel" project. Just include the storage path for the QR code images and the function for generating QR code names. The specific code is shown in the following figure:

```
temp_ID = 1
image_path = '/home/pi/PycharmProjects/final_project/MotorHAT'
str1 = "QR"
image_name = str1.join(str(temp_ID))
cv2.imwrite(image_path + image_name + '.jpg', frame)
```

10 Logging(Lanting Huang)

Initial Idea

This experiment aims to implement logging functionality, including two levels of logging: minimal and extensive. Through logging, we can better understand the status of the robot during operation and troubleshoot when problems occur.

Problem Encountered

- Initially, we used Python's built-in logging module for logging, but found that there was only one logger and it was impossible to switch between two different modes: minimal and extensive.

- It is found that the old log file needs to be deleted every time before a new log file can be recorded.
- The log file is not named with the required date
- After using the menu, if the log file is placed outside the function, it will be read first and then the menu is modified, causing the log to record wrong default data first.
- Put the log into the function and find that the logger is not defined, as shown in the figure
- The log file reported an inexplicable error after debugging, and the log file could not be opened. The error message showed was not UTF-8.

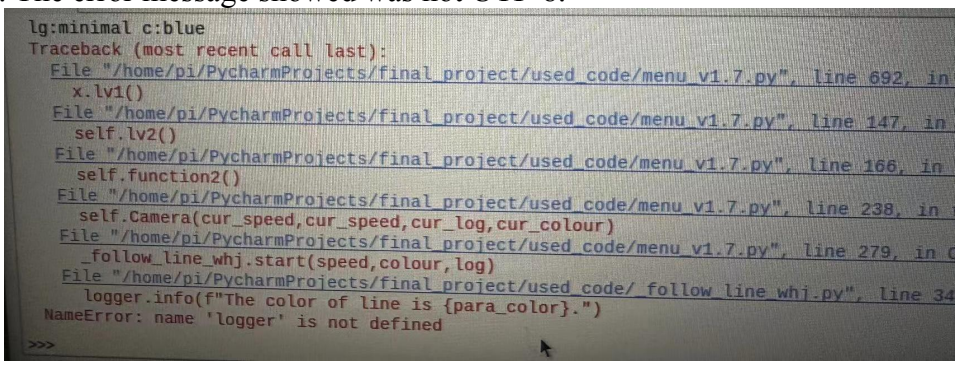


FIGURE 10.1

Problem Solved

- Use the menu module and add flag and if functions to select minimal and extensive modes.
- Add a logger and confirm the address of the original log file to ensure that the new log record can be stored in the existing file; if there is no file in the path, create a new one to store the log record. The code was changed from figure 10.2 to figure 10.3.
- Get today's date by calling the datetime library and calling datetime.today(). And use the join function of the os library and the correct format to add the date to the file naming. This ensures that files change every day and the amount of data stored is not too large.
- Put the log code into the function and call it to ensure the correct running sequence.
- Define a logger before calling the log function to ensure the existence of the logger.
- The logger file can be opened in python instead of arduino.

```
log_file_path = '/pi/PyCharmProject/logs/log_file.log'

log_format = '%(asctime)s - %(levelname)s - %(message)s'
logging.basicConfig(filename=log_file_path, level=logging.INFO, format=log_format)
```

FIGURE 10.2

```
new_logger = logging.getLogger('logger')
new_logger.setLevel(logging.DEBUG)

file_handler = logging.FileHandler('logger.log')
file_handler.setLevel(logging.DEBUG)

formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
file_handler.setFormatter(formatter)

new_logger.addHandler(file_handler)
```

FIGURE 10.3

Final Version

After solving the above problems and merging with the menu, the final version achieves the following functions:
Switch between disabled, minimal and extensive modes, which are no log, simple log and detailed log respectively. Differentiate log record files according to date and different functions, and record the time and operations performed by the system for easy inspection.

11 Project Scheduling (Yiwen Kuang)

Initial Idea

Microsoft Project is a project management software developed by Microsoft that is widely used for planning, tracking, and managing projects. One of the key features of Microsoft Project is its ability to create Gantt charts, which are visual representations of a project schedule. A Gantt chart is a bar chart that provides a visual representation of a project schedule over time.

I expected to manage the project according to the timeline, the tasks are divided into 6 deliverables:Project Schedule, Team Meetings agenda and minutes, Body design, Circuit design, Product demonstration, Lab report.

Problem Encountered

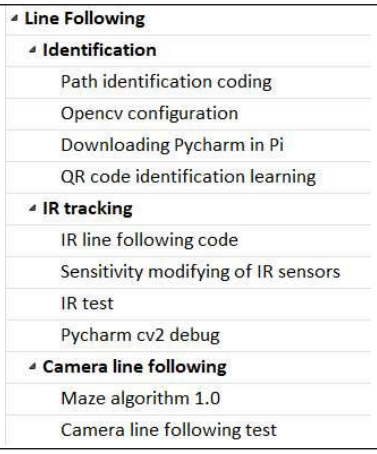
In the final month, the project requirements underwent numerous updates. There was an attempt to provide detailed descriptions of each phase's specifics and task arrangements. However, the project schedule could not be organized in chronological order, and the retrieval of dependencies became very chaotic.

For instance, the section on Camera Line Following combines maze navigation, QR code recognition, and path determination. Initially, tasks progressed smoothly in sequential order, but after revising the project requirements, the maze algorithm was removed. If additional subtasks were continued to be added within the same section, it appeared confusing due to the disrupted hierarchical relationships, rendering some tasks obsolete.

Problem Solved

Based on the aforementioned issues, I have reorganized the project in chronological order. We had a total of five meetings. After each meeting, individuals summarized their tasks from the previous week, and I updated and assigned tasks for the upcoming week. During this process, there were often iterative updates to the code, with the possibility of overturning previous work. To address this, I established different versions to reallocate tasks within the same phase.

Remarkable Changes

Initial Version	Final Version	Desciption
Menu	Menu (Part 1) + Menu (Part 2)	For Part 1 (28/12/23 - 29/12/23), group members constructed a structure of menu, like the wash machine menu we have done in previous lecture. For Part 2 (02/01/24 - 04/01/24), group menbers added real functional options in the menu.
Remote Control	Remote Control (Part 1) + Remote Control (Part 2)	For Part 1 (27/12/23 - 29/12/23), group members had successfully finished the basic movements, especially the left and right turn (not rotation anymore). For Part 2 (09/01/24 - 10/01/24), group menbers added the camera movement in remote control.
Path Identification + IR Line Following + Camera Line Following	Line Following	
Free Travel	Free Travel (Part 1) + Free Travel (Part 2)	For Part 1 (26/12/23 - 28/12/23), group members tested and modified the case code. For Part 2 (09/01/24 - 10/01/24), group menbers tried to combine QR code identification and obstacle avoidance.

12 Holding Meetings (Yifei Li)

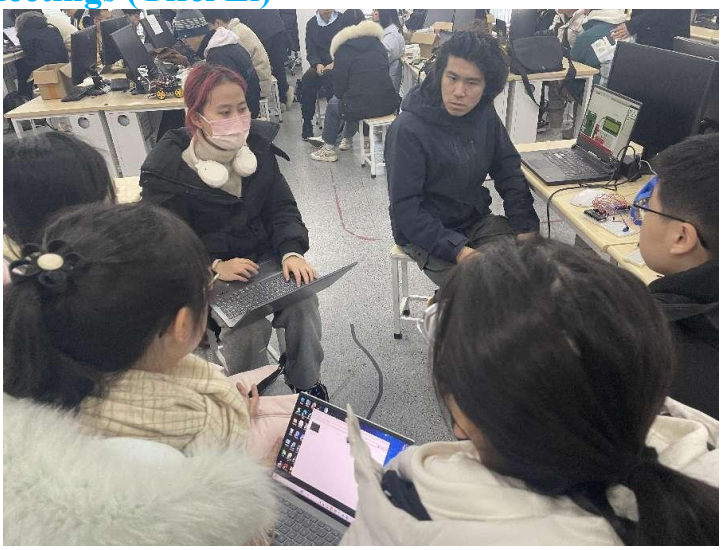


Figure 12.1

Meeting Date：05/12/2023

1.Team leader Kuang Yiwen elaborated on the project documents and gave an overall explanation of the project requirements, including Path identification, QR code, Robot operation. Then take project division according to individual abilities.
2. Group member discuss the appearance design plan together, including themes, Led lights, NFC parts installation and other ideas.

Meeting Date：22/12/2023

1. Overall report on project phase progress by each group member
The team discussed Path Identification, focusing on the utilization of IR sensors for blue line tracking.
2.Arrange the project work plan for the next week.

Meeting Date：28/12/2023

1.Yiwen Kuang summarise IR Line Following function from four parts
-Accomplishment in coding, sensitivity modification, delay parameters modifications
-Test (already finished): the straight line, curve, corner
-Problem unsolved in speed, stability and corner
2. Group member reported on Camera funtion from three parts:QR code identification, QR-camera angle, the maze algorithm
3.Explanation of menu function and LCD by Erzhizhi Hu
-Text display of the lcd
-The logical function implementation of the button
-The logical hierarchy of the menu
4.Overview of obstacle avoidance function
-Yifei Li talked about initial idea is to combine IR sensors with ultrasonic sensors for use.
5.Overview of Remote Sensing Function

Lanting Huang detailed the coding logic for the remote sensing function, explaining that infrared remote sensing is employed to remotely control five fundamental functions: forward, backward, left turn, right turn, and parking.

Meeting Date：02/01/2024

1.Review of updated final plan
-Introduce the changes to the final plan
-Introduce the new requirements made for each module due to changes
2.Report on Free Travel module
Yifei Li introduced an enhanced Free Travel mode based on final requests, incorporating existing functions for seamless operation.
3.Update remote control button: Erzhizhi Hu introduced two new requirement
4.Update robot option-line color: New requirement need us to recognize lines of different colors on the ground by using IR sensor or camera
5.Update wires requirement need us to standardized use male to male and female to male
6.Light sensor address: New requirement need us to use RGB sensors instead of initial planned IR sensors for obstacle avoidance, autonomously adjusting the color of the light according to the ambient light.

Meeting Date：09/01/2024

1.Overview of updated final criteria by Yiwen Kuang
-Summarize the progress of last week's work
-Introduce the second changes to the final plan
-Arrange the final test task to each member
2.Introduce update LCD placement by Erzhizhi Hu
We need to update the placement strategy for the LCD, ensuring it can be easily removed and repurposed for future projects. Team has proposed a change where the "Button" is attached to the breadboard on the car, and the LCD is affixed to a separate breadboard.
3.Introduce update construction by Yifei Li
4.Introduce update camera line following by Hanjing Wang
5.Introduce new requirement for Light QR address by Yiwen Kuang

13 Summary (Hanjing Wang)

In this project, we acquired proficiency in remote connectivity with the Raspberry Pi, facilitating the implementation of various functionalities for the robot. In the IR line-following module, we leveraged IR sensors to discern the vehicle's optimal direction. The free travel module saw us adeptly implementing obstacle avoidance, QR code recognition, and photography. In the menu and remote control module, we honed our understanding of circuit connections and the integration of code. The logging module provided insights into automated log file storage. Lastly, in the camera line-following module, we developed skills in intersection recognition and parameter adjustments to ensure the vehicle's adherence to expected performance.

This comprehensive exploration has not only enhanced our technical prowess in programming but has also deepened our understanding of practical applications of robotics. Moving forward, we aspire to further delve into this field, contributing to the ongoing advancements in robotic technology.

Appendix

References of Menu Session:
https://www.jianshu.com/p/4cadf321f5e8

References of QR Code Identification Session:
https://blog.csdn.net/whatday/article/details/91990475/
https://blog.csdn.net/wu_zhiyuan/article/details/126545476
https://blog.csdn.net/search_129_hr/article/details/121236834
https://blog.csdn.net/c10WTiybQ1Ye3/article/details/123564013

References of QR Code Path Session:
https://blog.csdn.net/Chockong/article/details/127898927
https://blog.csdn.net/qq_45505081/article/details/131719811
https://blog.51cto.com/u_16175446/6817770

References of Numpy Session:
https://blog.csdn.net/m0_62988777/article/details/131859178
https://blog.csdn.net/m0_52848925/article/details/130877566
https://blog.csdn.net/yuan2019035055/article/details/128709583
https://blog.csdn.net/BGONE/article/details/122718727

References of Camera Session:
https://blog.csdn.net/the_future_way/article/details/116666536
https://apachecn.github.io/opencv-doc-zh/#/

References of Logging Session:
https://zhuanlan.zhihu.com/p/166671955

References of Free Travel Session:
https://blog.csdn.net/qq_39852676/article/details/96430831
https://pythonjishu.com/no-such-device/