Year 2 Project

Navigation Desktop Robot with Edge Avoidance

By: Nour Almosilli (201587776), Tom Roberts (201510137), Abdul-malik Al-marhoubi (201601921), Tianxing Ji (201676480), Yuhang Long (201676760).

Supervised by: Dave McIntosh

Department of Electrical Engineering and Electronics

10 May 2023

Abstract

This project aims to develop a navigation desktop robot with edge avoidance, mainly based on Arduino Uno Rev3, Adafruit Motor Shield V2, and Infrared sensors. It utilizes signals from infrared sensors to navigate, identify edges and prevent falls, after which it will travel inward the table and then go randomly without falling from the table. However, it can only travel clockwise and rectangular-shaped tables, which needs further development. Nevertheless, this project still recommends a possible way to expand the movement patterns of traditional robots on tables or areas with differential depths.

Moreover, the findings during this project also point out that simple optical-based sensors such as infrared sensors cannot output steady signal values because of the surrounding environment and the connection quality.

Declaration

I confirm that I have read and understood the University's definitions of plagiarism and collusion from the Code of Practice on Assessment. I confirm that I have neither committed plagiarism in the completion of this work nor have I colluded with any other party in the preparation and production of this work. The work presented here is my own and in my own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web). I understand the consequences of engaging in plagiarism and collusion as described in the Code of Practice on Assessment (Appendix L).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This part will show the background introduction of the robot area and motivation for conducting this project first, after which objectives will be given. Finally, there is a brief introduction to four significant electronic components of the Navigation Desktop Robot with edge avoidance, including their essential functions and theories but not their detailed principles.

Furthermore, the Navigation desktop robot with edge avoidance will be abbreviated to Navigation Robot or NDR after this part.

1.1 Background Introduction and Motivation

In recent years, different kinds of robots have experienced enormous change because of improved sensor technology [1]. Thus, versatile types of robots have been applied practically to satisfy various needs in several fields, such as Healthcare, Manufacturing and Travel [1], which shows the importance of developing more specific robots for potential users. However, the movement patterns of robots still need corresponding research.

1.2 Objectives

Therefore, Navigation Desktop Robot with edge avoidance aims to extend the function of traditional robots to platforms with differential depths. In other words, contrary to Merlo's [2] direction of optimizing the robot's traditional functions as much as possible, Navigation Robot's goal is to expand the movement patterns and the areas in which the robot can move.

1.3 Introduction for Four Main Electronic Components

Moreover, four essential components are made up for this product: Arduino Uno Rev3, Adafruit Motor Shield V2, Infrared Sensors, and Photo Interrupter Sensors. However, Photo Interrupter Sensors were abandoned at last for some reasons, which will be explained in this report.

1. Arduino Uno Rev3

The UNO is one of the series of Arduino families. Furthermore, Rev3 is the third revision of this type of Arduino Uno, which is also the latest one until nowadays [3]. Furthermore, it is a microprocessor board based on ATmega328P, which also includes other necessary things to support the microprocessor [3]. Additionally, the most suitable input voltage is 7V, which can ensure the computational ability of the microprocessor [3].

2. Adafruit Motor Shield V2

Adafruit is its brand name. Moreover, V2 is the second version of this type of motor shield, which is also the latest one until nowadays [4]. Unlike other motor drivers and shields, it has a fully dedicated PWM driver chip and a completely stackable design [4]. Additionally, it is a kind of I2C, which also occupies two GPIO pins plus 5v and GND on Arduino [4].

3. Infrared Sensors (IR sensors)

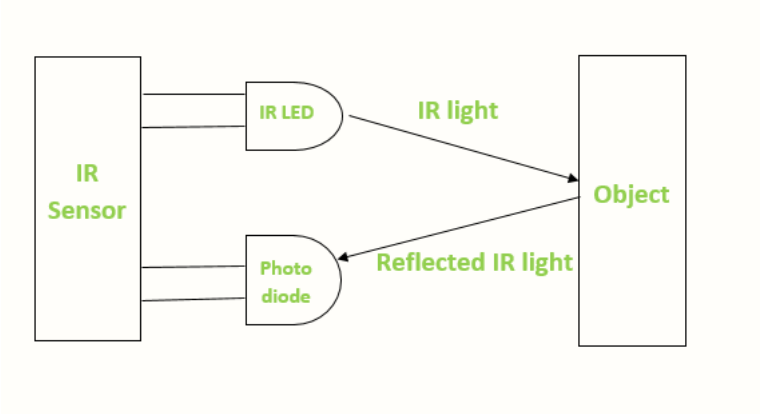


Figure 1.1. The function of IR sensors (taken from [5])

An IR sensor comprises an infrared emitter and receiver [5]. As shown in Figure 1.1, IR LED is the infrared emitter, and Photodiode is the infrared receiver.

The sensor is solely able to output Boolean values [5]. For example, if the infrared receiver gets the reflected IR light sent from the IR LED and is reflected by one of the surrounding objects, it will output a LOW signal value. Otherwise, it will output a HIGH signal value.

4. Photo Interrupter Sensors (PI sensors)

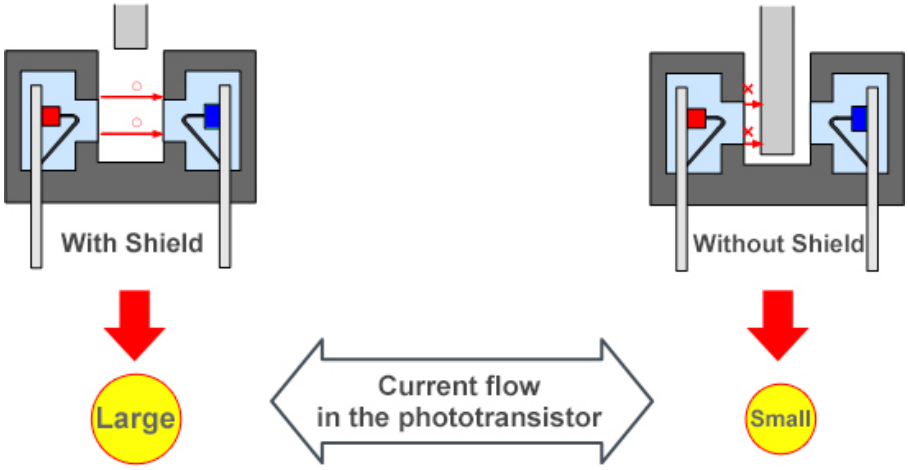


Figure 1.2. The function of PI sensors (taken from [6])

Like IR sensors, the red unit is an infrared LED, and the blue unit is a phototransistor [6], as shown in Figure 1.2.

The sensor can only output Boolean values [6], as illustrated in Figure 1.2. More specifically, if a shield can cut down the light from the red unit to the blue unit, it will output a LOW signal value, which is also a tiny signal value. Otherwise, it will output a HIGH signal value, which is also a considerable signal value.

Chapter 2

Materials and Methods

This part will show the materials used in the project and the detailed steps of the whole construction.

2.1 Materials

1. Arduino Uno Rev3
2. Adafruit Motor/Stepper/Servo Shield for Arduino v2 Kit - v2.3
3. 20 Pieces IR Infrared Obstacle Avoidance Sensor Module 3-Wire Reflective Photoelectric Sensor Module Compatible with Arduino Smart Car Robot Raspberry Pi 3
4. Photo Interrupter Sensor
5. Mini 3-Layer Round Robot Chassis Kit - 2WD with DC Motors
6. TT Motor Encoder (Pack of 2)
7. 120-Piece Ultimate Jumper Bumper Pack (Dupont Wire)
8. 170-Tie-Points Mini Breadboard Kit
9. 3D printed clips
10. 9V Battery non-rechargeable
11. 4 x AA Battery Holder with On/Off Switch
12. Arduino stackable header

Explanation:

The first four main components have been introduced in the Introduction part.

The fifth one, which is a robot chassis kit, can save time in designing and implement more detailed things.

The sixth one, which is the TT motor encoder, is combined with photo interrupter sensors to calculate the distance that the robot has travelled.

The seventh and the eighth ones, which are jump wires and mini breadboard, are used to organize the wires to make Arduino Uno Rev3 can be connected with more IR sensors and easy to be connected.

The ninth one, which is a 3D-printed clip, is used to attach the infrared sensors to the robot.

The tenth and eleventh ones, which are 9V batteries and a battery holder, are used to power Arduino and motors.

The twelfth one, which is Arduino stackable header, is used to connect Arduino Uno Rev3 and Adafruit Motor Shield V2.

2.1 Methods

There are eight significant steps of this project, which are shown below.

2.1.1 Research and Understand Components

Step 1: Search related datasheets and tutorials online about the components

Step 3: Connect them to the circuits and Arduino

Step 3: Test the components to verify the information above

2.1.2 Install Robot Chassis

Step 1: Search related tutorials online about the Chassis

Step 2: Follow the instructions and assembly the Chassis

Step 3: Test stability for subsequent adjustment

2.1.3 Create 3D Printed Clips

Step 1: Measure the size of the relevant aperture of the chassis

Step 2: Create 3D print model in model software

Step 3: Print and test the 3D model

Additional notes:

Although 3D printed clips is the third important step in this report, it is actually presented at the end stage in the whole project. However, it should be presented at the beginning of the project ideally, which can make the project more organized and face less problems such as the unstable placements of sensors.

2.1.4 Test for Distance of Detection of IR Sensors on Different Materials

Step 1: Connect an IR sensor into the circuit

Step 2: Switching objects of different materials to test the bounds of its Boolean switch

Step 3: Repeat 5 times and calculate the average value of each material

Step 4: Record the values

2.1.5 Install all the Components on the Chassis

Step 1: Design the different positions of components

Step 2: Put them on the robot

2.1.6 Write codes that takes input from IR sensors which indicate presence of a surface and drive accordingly

Step 1: Design the processing of signal values

Step 2: Implement them in the code

Step 3: Compile and run the code

Step 4: Adapt code according to unexpected movement patterns and back to Step 3

2.1.7 Calculate distance of the edges by using the speed determined by the wheel encoder/time and therefore calculate the area of desk

Step 1: Design the measurement of distance and represent the related area of the desk

Step 2: Implement them in the code

Step 3: Compile and run the code

Step 4: Adapt code according to unexpected movement patterns and back to Step 3

Additional notes:

The measurement of distance is calculated by the combination of wheel encoders and photo interrupter sensors. However, there are more errors that will be illustrated in the related result part. Therefore, Arduino built-in clock was used to measure the distance and then calculate the area of the table in the end.

2.1.8 Write an algorithm that allows the robot to cover the whole area of the desk

Step 1: Design the movement patterns according to the area calculated by 2.1.7 step.

Step 2: Implement them in the code

Step 3: Compile and run the code

Step 4: Adapt code according to unexpected movement patterns and back to Step 3

Chapter 3

Results

This part will show the results of each method described in Chapter 2, accompanied by error and uncertainty analysis.

3.1 The Result of Components

This will be divided into two parts, which are function part and code part.

Sensors are mainly in the function part.

Arduino and Motor Shield in mainly in the code part.

Function part:

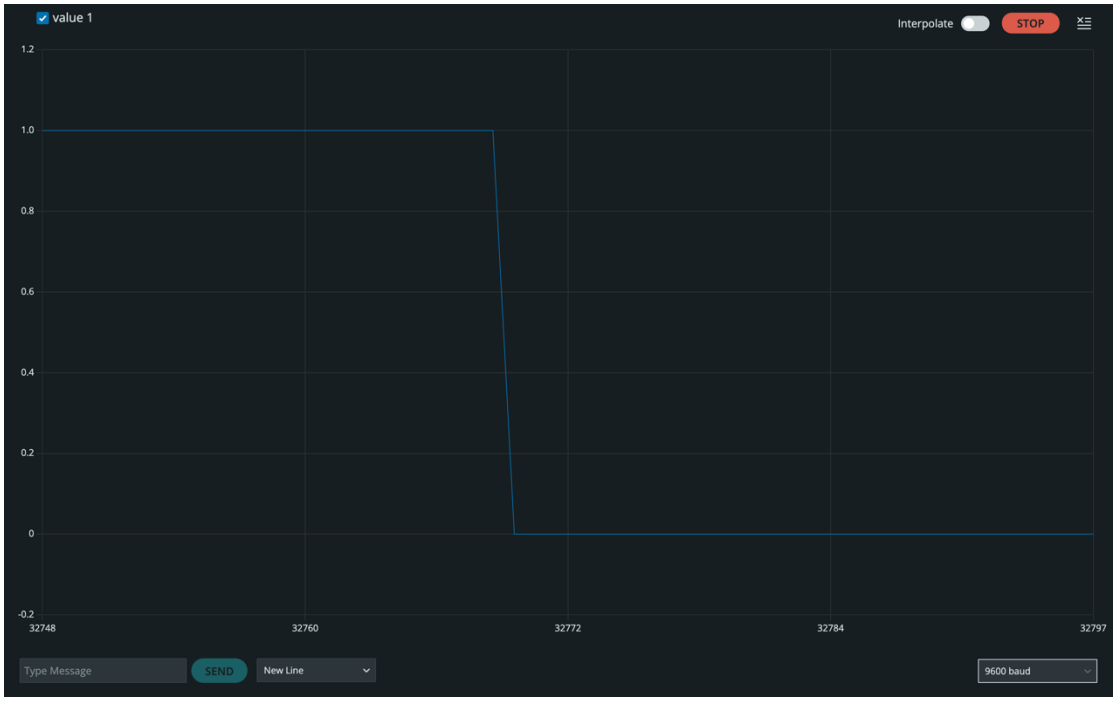


Figure 3.1. The response of IR sensor (created by authors)

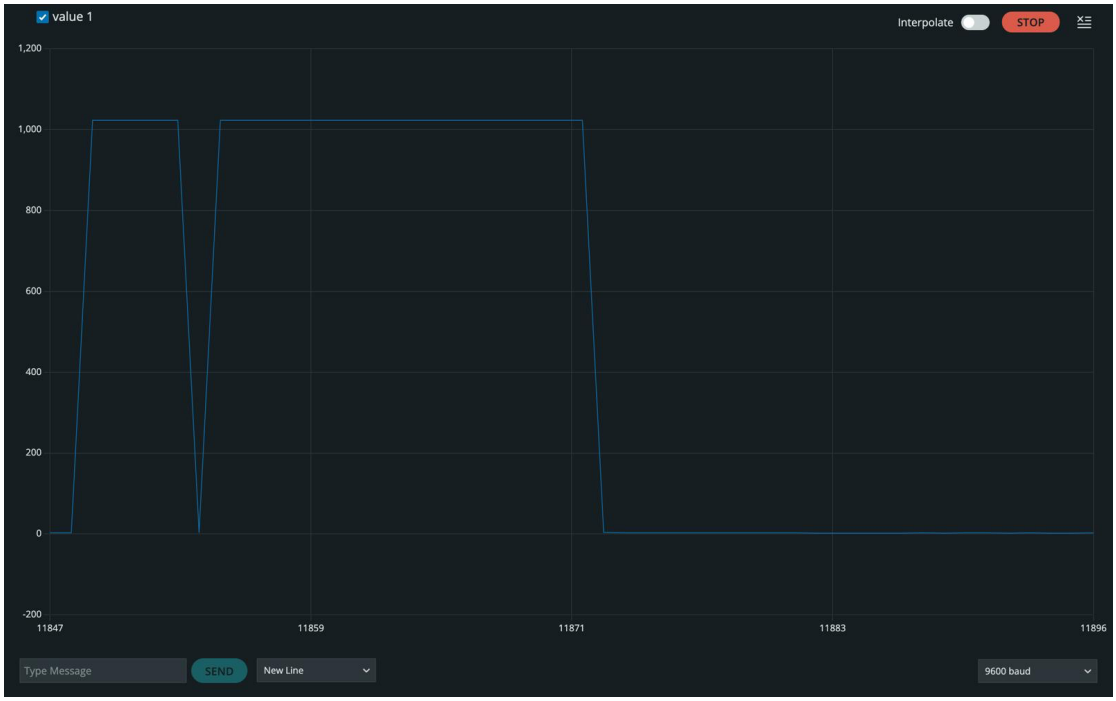


Figure 3.2. The response of PI sensor (created by authors)

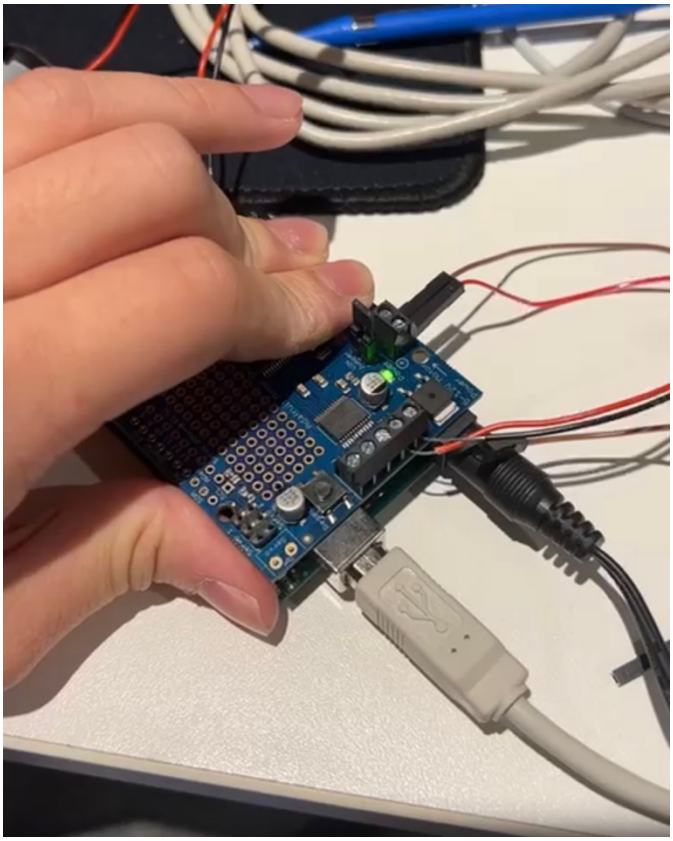


Figure 3.3. The connection between Arduino and Motor shield (created by authors)

Code part:

The tutorials of basic code knowledge can be found in articles from [3][4]. They show detailed parts of each coding method and related library that must be input before running the motor shield, which is in Appendix C.

Figure 3.1 and Figure 3.2 are generated by Arduino IDE, which is a platform for Arduino programming. Moreover, the IR sensor and the PI sensor should be connected to Arduino digital pins because they output Boolean values [5][6]. Then, the signal value can be read by Arduino IDE through reading commands. which is in Appendix C.

Explanation:

Figure 3.1 and Figure 3.2 verify the functions declared in [5][6], which only output Boolean values according to different situations announced in the Introduction part. However, they cannot make reactions ideally. To be more specific, the signal value each has a slope when they are changed, which represents practical time delays. Moreover, the PI sensors always have fluctuations after being changed five times, which shows that this kind of simple PI sensors cannot output steady values. Therefore, these two features should be considered before starting this project.
Furthermore, Figure 3.3 shows the effective connection without soldering. However, this connection is not steady and can cause the Arduino to break, which must be paid attention to this issue. After a few weeks of agonizing, stackable headers were decided to connect Arduino and Motor Shield, which proves to be the suitable way.

Error Analysis and Advice:

According to [3], there is a time interval/delay caused by the fact that 1 ADC (Analogue to Digital Conversion) read needs 13 ADC clock cycles, which is the problem that gingerades the slope in Figure 3.1 and Figure 3.2.
Moreover, the photo interrupter sensor is an optical-based sensor [4], which means it is sensitive to the light from surroundings. Additionally, it is also a problem caused by the weak connection, which uses equal or more than two jump wires. Therefore, reading correct signal values from photo interrupter sensors needs one and only one jumper wire connection and a slightly dimmer environment, which makes the photo interrupter abandoned by this project because of these two complex requirements.

3.2 The Result of Robot Chassis

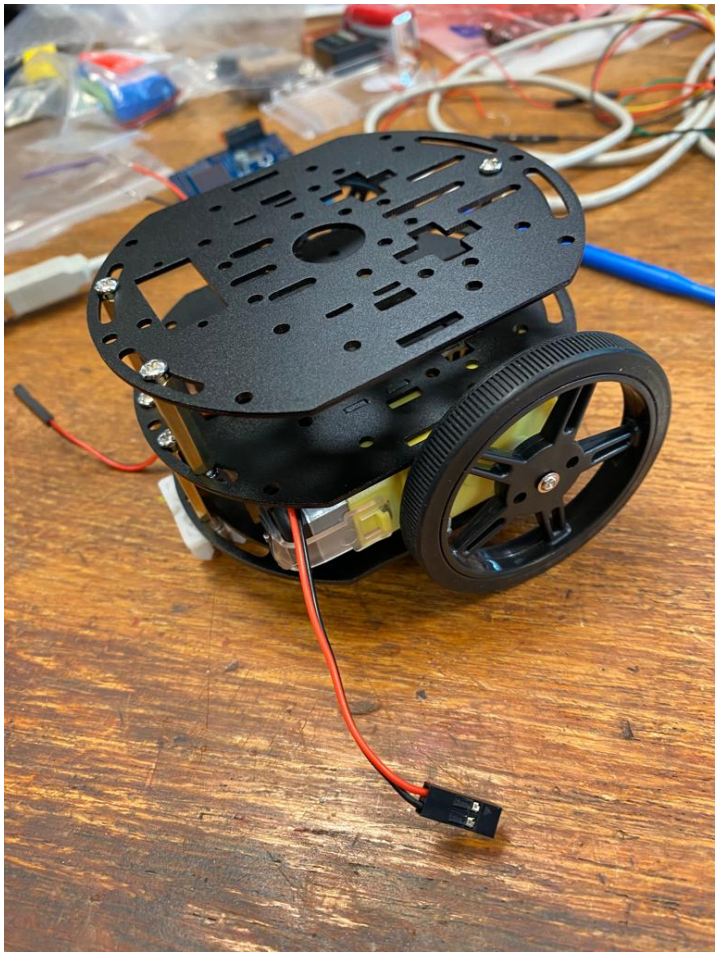


Figure 3.4. Completed assembly of the chassis (created by Abdul-malik Al-marhoubi)

Explanation:

According to the steps in [7], the parts of the robot are assembled in turn, and the final result is shown in Figure 3.4. After this, solidity should be tested to make sure that the chassis is put up correctly, which is a significant element in the next steps. In other words, the well-structured chassis can provide a safe shelter for the components used in this project.

Error Analysis and Advice:

During the assembly of the robot chassis, the wires on the DC motors fell off because the original solder is fragile, but can be soldered back on. The black and red wires can be connected casually to the metal of the two power connectors of the motor because the order of the wires does not matter and can be adjusted indirectly by the motor shield.

3.3 The Result of 3D Printed Clips

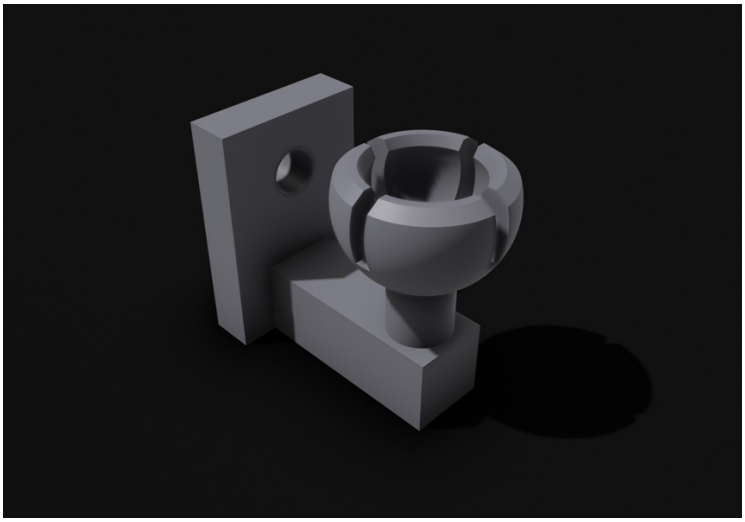


Figure 3.5. 3D model of one part of a 3D printed clip (created by Tom Roberts)

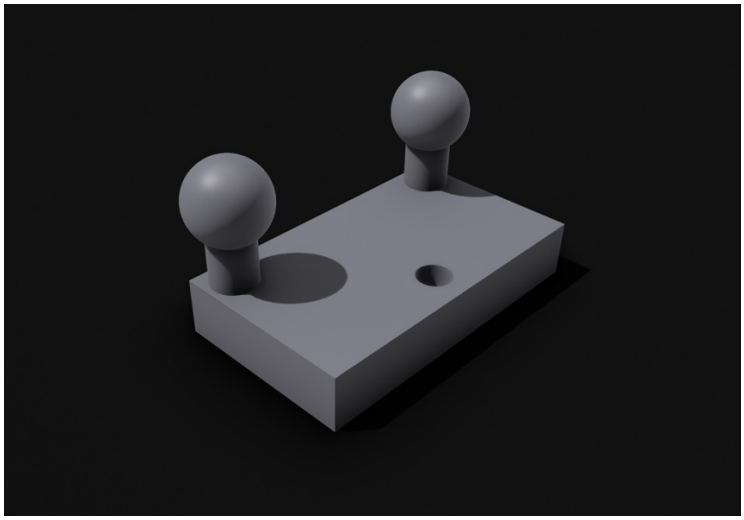


Figure 3.6. 3D model of the other part of a 3D printed clip (created by Tom Roberts)

Explanation:

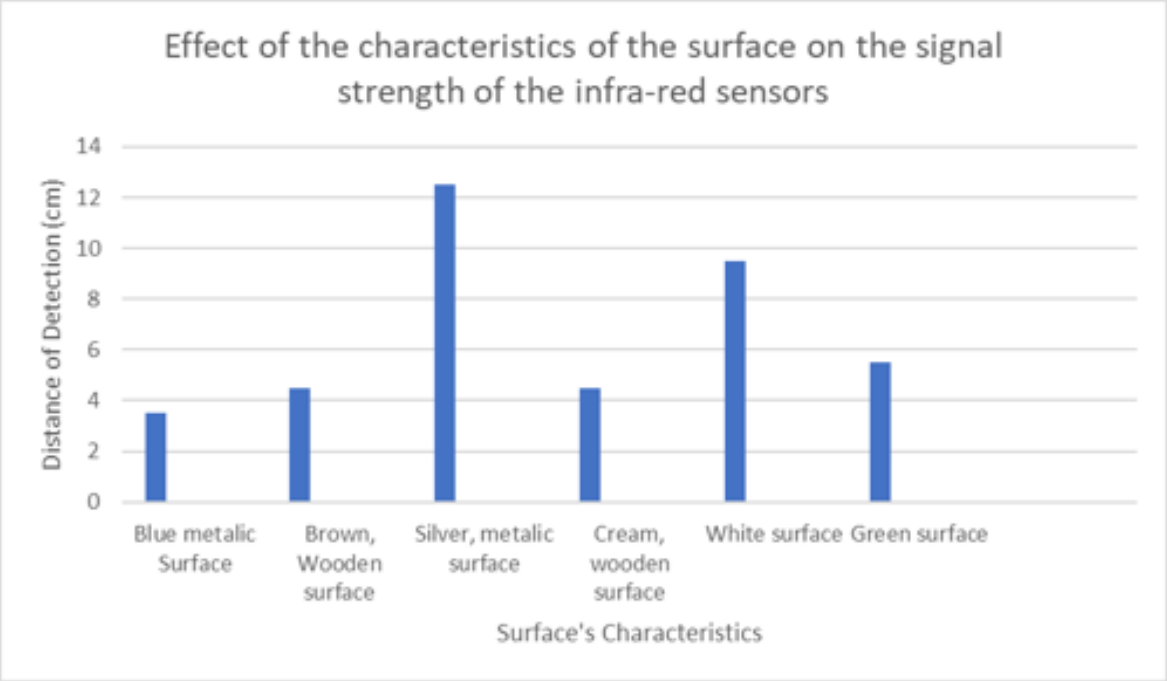
The combination of 3D models in Figure 3.5 and Figure 3.6 is the final 3D printed clip to attach the sensor to the chassis. Moreover, it suits the chassis because it is designed based on the small hole sizes for the chassis.

Error Analysis and Advice:

Because they are made for both adaptability and fixity, they can be changed easily sometimes, which can be not only an advantage but also a disadvantage. For example, the placements of sensors are suitable after adapting for a long time, but they can be changed unexpectedly when touched slightly, which will ruin the previous positions. Therefore, these two can be the models for the first draft and then designed for a more fixed clip, which can reduce the pressure of adjusting the placements of sensors.

3.4 The Result of Tests for IR Sensors

Table 1. The Effect of the Characteristics of the Surface on the Signal Strength of the Infra-red Sensors (with fixed sensitivity) (created by Nour Almosilli)



Explanation:

Table 1 shows the different distances of detection on various characteristics of surfaces with default sensitivity, which is based on the average value of 5 tests.
The horizontal axis means the different surface's characteristics while the vertical axis shows the distance from 0 to 14, whose unit is a centimetre.
The distance of detection on the silver surface is the largest one among all of them, after which is that on a white surface. Moreover, the rest are similar to each other, which is around 4 centimetres.

Error Analysis and Advice:

According to [5], the IR sensor is an optical-type sensor, which means the accuracy of the detection is determined by the infrared light. Meanwhile, there are numerous infrared emitters in the environment when the IR sensor starts working. Therefore, the infrared receiver of the IR sensor can be influenced by the surrounding

environment easily, which also throws up a major obstacle to the operation of the robot. Then, a change of sensitivity or position at any time is necessary for the infrared sensor and is the only way to do it in this project.

3.5 The Result of the Chassis with all Components

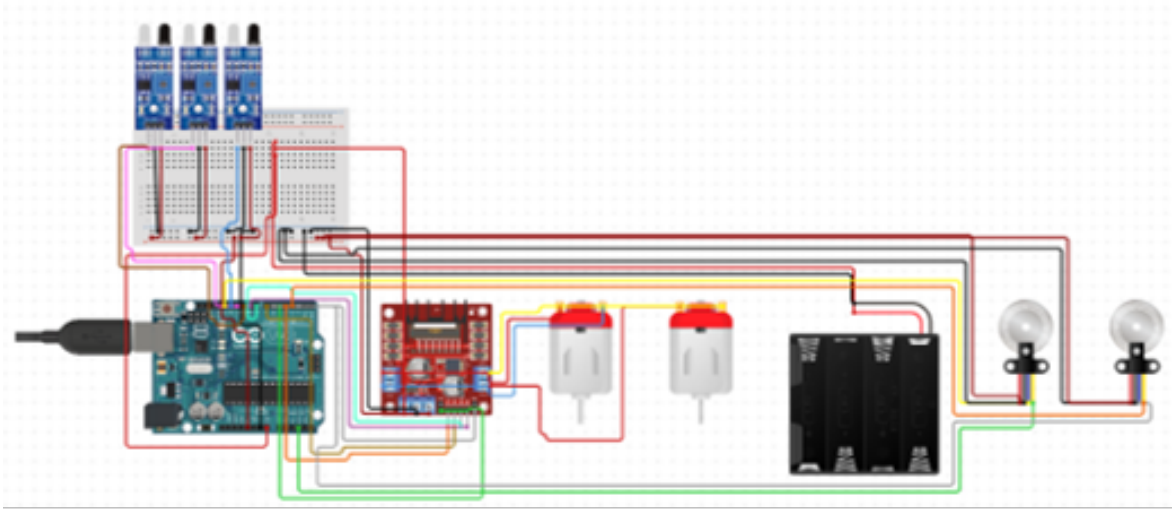


Figure 3.7. The whole circuit of this project (created by Nour Almosilli)

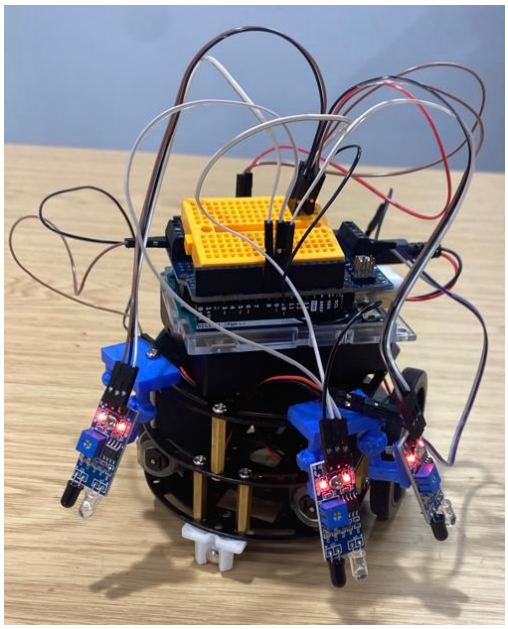


Figure 3.8. Completed assembly of the chassis with all components (created by Abdul-malik Al-marhoubi)

Explanation:

Figure 3.7 is the circuit of this project. There are three IR sensors, two PI sensors, two motor encoders, a battery holder, a motor driver and an Arduino Uno Rev3. However, it is slightly different from the real one. To be more specific, the combination of the PI sensor and the motor encoder is abandoned at last, which will be demonstrated in part 3.7. Moreover, Adafruit Motor Shield is used in this project instead of a motor driver, which is added because there is no motor shield component on the digital circuit. Additionally, the USB port will be replaced by a 9V battery in this project for the driving of the Navigation Robot.
Figure 3.8 shows the completed assembly of the chassis with all components, which lays a necessary foundation for the next operation.

Error Analysis and Advice:

Though this part is used to show the assembly of the robot chassis with all components and the whole circuit, there is still one impartment that can be applied, which is that the countless jump wires can be organized in the different layers of the chassis, which can make the robot looks neater and easier to diagnose connection errors.

3.6 The Result of Codes for Driving Robot according to Signal Values from Sensors

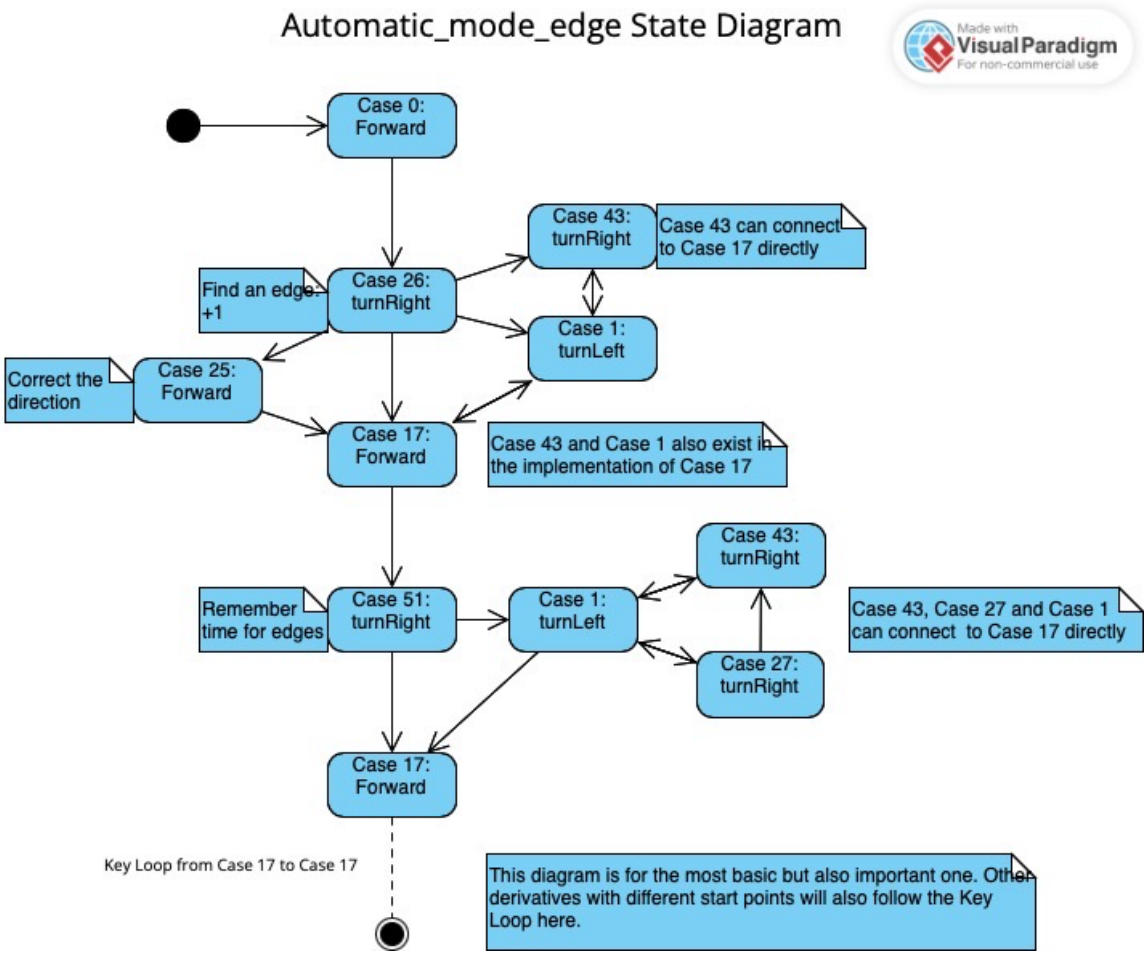


Figure 3.9. The state diagram of basic automatic_mode_edge (created by Tianxing Ji)

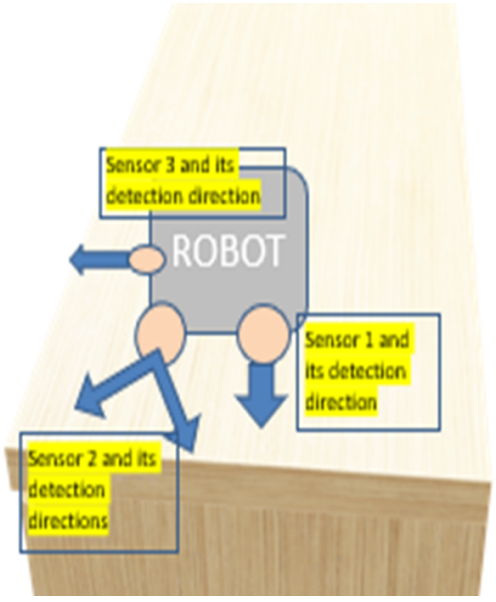


Figure 3.10. The positions of IR sensors and their direction directions (created by Nour Almosilli)

Brief Introduction of the logic of driving code:

Figure 3.9 shows the most basic but also the most important one in Automatic_mode_edge() in code, which is included in Appendix C.

There are several points that need to be clarified before the detailed explanation of the state diagram:

1. The method name represents the automatic mode of the robot to find an edge. Thus, this method is named Automatic_mode_edge().
2. The signal value from the IR sensors is a Boolean value, which is 0/1. 0/1 can also be defined as HIGH/LOW and ON/OFF.
3. XX of Case XX in Figure 3.9 is calculated by the summation of the multiplications of the assigned value and signal value from different IR sensors plus a Boolean value to determine the state of whether the robot finds an edge or not. In this project, Sensor 3 is assigned a value of 8, Sensor 2 is assigned a value of 26 and Sensor 1 is assigned a value of 16 according to Figure 3.10.

   Then the formula will be:

   26 * signal value of Sensor 2 + 16 * signal value of Sensor 1 + 8 * signal value of Sensor 3 (+ edge_find)
4. edge_find is a Boolean value to determine the state of the robot that it finds an edge or not, which is default 0. After finding an edge, it will become 1.
5. Forward, turnRight and turnLeft under Case XX are the movement patterns the robot needs to take when in Case XX.
6. Most importantly, Case 51 will not only include the movement pattern to turn right 90 degrees but also the function to remember the time, which lays a

foundation for the measurement of the length of the edge and area of the table in part 3.7.

Detailed Explanation:

The expected movement patterns can be determined and conducted by the case number (XX) calculated by the formula above, which is a transformation of digital signals and physical behaviors.

The ideal case follow is from Case 0 vertically down to the second Case 17. this series of changes represents the robot being placed in the middle of the table at first and then moving forward until the first infrared sensor on the left finds the edge, then it allows the second sensor on the left to detect the edge and move forward along the edge to the corner, then it then has the second sensor on the left detect the edge and travels forward along the edge to the corner, before making a 90-degree right turn and continuing on its way. To be more detailed, there are some related formulas to calculate the case numbers according to different scenarios:

1. In the middle of the table: 26 * 0 + 16 * 0 + 8 * 0 (+ 0) = 0.
2. The first sensor on the left detects an edge: 26 * 1 + 16 * 0 + 8 * 0 (+ 0) = 26. (After this step, the edge_found will become one because it finds an edge)
3. The second sensor on the left keeps detecting an edge: 26 * 0 + 16 * 1 + 8 * 0 (+ 1) = 17.
4. In the corner: 26 * 1 + 16 * 1 + 8 * 1 (+ 1) = 51.
5. Back to step 3: 26 * 0 + 16 * 1 + 8 * 0 (+ 1) = 17.

However, the process is so idealised that it does not always proceed directly from Case 0 to the second Case 17. more specifically, there are many other derivative cases during the robot's progress, which are shown on the left and right sides of the idealised path in Figure 3.9. Therefore, when the robot generates these unexpected

case numbers, it responds accordingly to get back to the desired path, which is from Case 0 to the second Case 17. Furthermore, the case numbers of these derivative cases are similar to the formula shown above.

Moreover, the path from the first Case 17 to the second Case 17, which will loop 5 times to make the robot get back to the first corner, is the most important loop in this mode. For example, this loop will always be executed five times to wherever the robot starts such as in the corner or near the edge as noted in Figure 3.9.

Error Analysis and Advice:

However, the conversion between cases sometimes does not follow the state diagram, which appears to be wrong. For example, a common scenario in this project is that when the robot triggers case 26, edge_find becomes one because the robot has found the edge, and then it proceeds to Case 1 in a continuous loop due to some unexpected movement patterns or positions of sensors, which is excluded in Figure 3.9.

Therefore, there are two possible solutions to this problem.

The first one is that design more tolerant movement patterns according to different cases, but it is complex and takes a lot of time because there is more than one case after the change of the current case.

The second one is that find a more suitable placement for each sensor. This one is more practical and intuitive. Meanwhile, a steady way to place the sensors is required to conduct this method such as 3D printed clips.

3.7 The Result of Measurements of the Lengths of Edges and the Areas of Tables

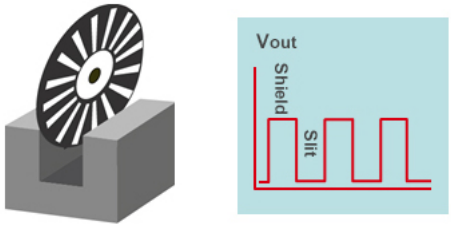


Figure 3.11. The combination of motor encoder and PI sensor (taken from [6])

Explanation:

Figure 3.11 shows the previous idea to measure the length of edges in this project. The rotation of the motor encoder will create a continuous change of signal values output by the PI sensor, which can be used to calculate the distance travelled by the robot. After this, the reprocessed distances can then be used to represent the edge lengths and areas. Moreover, this part of the code is in Appendix C, whose name is counter(). However, there are some problems that cannot be figured out by using this combination, which will be talked about in the Error Analysis and Advice below. Therefore, time-based measurement of edge lengths is used to replace this method. To be more specific, time-based measurement is based on the Arduino built-in clock, which can be called by millis() method in Arduino IDE. And this method only has errors that can be accepted and ignored. Therefore, this time-based measurement was used in this project in the end.

Error Analysis and Advice:

It is common that the PI sensor cannot work and record the change of the signal value sometimes because it is an optical-based sensor [6]. In other words, surrounding objects or sunlight can influence the function of this sensor and make it not work somehow. Meanwhile, the signal values are not steady if connected more

than two jump wires, which has been shown in Figure 3.2, which makes the robot not stable. Meanwhile, there is a slight time delay when millis() method has been called every time, which is 1 microsecond. Because this error is smaller than the missed count of the PI sensor, which can also be ignored, the time-based measurement is recommended. Moreover, if the speed is obtained from the PI sensor and then the distances can be calculated, the time is still needed to set the amount of time to drive the robot. Therefore, the time-based is more intuitive and suitable for this project.

3.8 The Result of Codes for Covering the whole Table

This part is called Certain_mode() in the code, which is in Appendix C, which is accompanied by Automatic_mode_random().

Explanation:

After part 3.5 and part 3.6, the time used to navigate each edge can be calculated, which is represented by edge_length[0] and edge_lenth[1] separately in the code.
The first method name, which is Certain_mode, comes from the fact that the time for travel inward the table has been preset in the code based on the calculation of Automatic_mode_edge(). Therefore, it is a kind of hardcode and gives up the signal values coming from the sensors, which will make the movement of the robot become dangerous and have possibilities of falling down from the table.
The second method name, which Automatic_mode_random, comes from the fact that the robot will run randomly based on the signal values of two front sensors, which has the same logic as Automatic_mode_edge.

Error Analysis and Advice:

Certain_mode():

The logic of this mode is correct. However, it is giantly dependent on the time calculated by Automatic_mode_random, which makes it unsteady in many cases. To be more specific, only Case 51 in Figure 3.9 remember the time, based on which the length of each edge will be measured. Therefore, the time for correcting the directions of the robot in derivative cases will also be counted in the length of the edge. Thus, there are two possible ways to solve this problem.

The first one is that calibrate the time for the length of the edge by a specific formula or value, which is also the method in this project. However, this calibration formula and value will become more and more complex as the rising times of correcting the robot's directions, which is hard to use and not recommended.

The second one is that stops the time calculation to ensure time accuracy. However, this approach is difficult to implement given the author's current knowledge base, although it sounds feasible and less complex than the first method.

Automatic_mode_random():

This model is not designed by the authors of this project, which comes from the online tutorial [8]. It has been tested. Thus, there is no advice for this model.

Chapter 4

Discussion and Conclusions

4.1 Explanation for Finding

1. Sensors

The IR sensors and PI sensors are optical-based sensors [5][6], which means that they need strict environments to work ideally. Otherwise, they will output unsteady signal values shown in Figure 3.2, which can have sa ignificant influence on the movement patterns of robots and derivative algorithms. Moreover, the jump wires used to connect them should be high-quality to establish stable connections between Arduino and these components.

2. 3D printing

3D printed products can easily be suited to the project if needed. Therefore, it is suitable for those who want the necessary components but cannot buy them through rational channels. Moreover, this kind of 3D printed product takes an enormous amount of time to model and calibrate, which should be designed at the beginning of a project.

3. Optimization

A circuit that works and one that works well are two different things. In other words, the complete chassis with all components as shown in Figure 3.8 can work, but it throws a huge strain on the implementers and subsequent proofreading changes because of the haphazard lines, which should be organized before or optimized later.

4. Algorithm

The algorithm in the code is divided into two parts.

The first part is in the Automatic_mode_edge() method and Automatic_mode_random() method. This part of the algorithm is intricate, and as Figure 3.9 shows, a Case does not stand alone but is interconnected with many other cases, influencing and transforming each other. Designing and sorting out their relationships requires a state diagram.
The second part is in a Certain_ mode. This part of the algorithm is simple. It is finished by a simple loop, which tells the robot how long to travel at each side.

4.2 Description for Objectives

The findings above support this project to achieve part of the original objectives. To be more specific, this project expands the movement patterns of traditional robots on the table, which is part of differential depths.
Meanwhile, there are still more parts that need to be realized such as height difference, colour difference and similar stuff, which can really expand the movement areas of the Navigation Robot.

4.3 Self-evaluation

The project is progressing slightly on schedule, although the following issues have arisen:

1. Two Arduinos were compromised due to a misunderstanding of the motor shield
2. Untimely project management, such as the introduction of 3D printing, made the whole project seem tight.
3. IR sensors and PI sensors output unstable signals due to the surrounding environment, which needs to be fixed but is beyond the authors' current knowledge base. So, some functions were abandoned

And there are some solutions to these problems:

1. Listen to the tutorials online and try rational alternative methods. Meanwhile, anticipate the possible outcomes
2. Design and organize one project beforehand and then manage the time schedule.
3. Pick more robust sensors or find other sensors that can act as alternatives.

Because of the limited time, there are several limitations of Navigation Robot from small to large:

1. The Navigation Robot of this project can only travel clockwise.
2. It can only travel on the rectangular shaped table
3. Less robust sensors cause unexpected values during the driving process
4. The second mode in the code, which is called Certain_mode, is a kind of hard code. Moreover, it also abandons the use of sensors and then puts the robot in danger.

However, this is still an important project and precious report for those who:

1. Interested in movement patterns of robots
2. Interested in how to make the robot travel on the table without falling down
3. Want to expand the movement patterns of robots with edge avoidance

Because this report includes basic and detailed pieces of knowledge for the realization of all these kinds of movement patterns with edge avoidance on the table, which can be and deserves to be expanded to a higher level.

4.4 Further Development

For the limitations that have existed in this project, some recommendations are put here:

1. Increasing adaptivity by using more sensors to allow the robot to turn anticlockwise.
2. Improving the code so the robot can function on any shaped table not only squared and rectangular desks.
3. Use more sensitive sensors, e.g., ultrasound sensors because the strength of the infrared sensor's signal can be influenced by characteristics of the desk surface and external light.
4. Additional hardware such as an SD card could be used to give the robot the ability to map the desk.

For the practical applications, the theory and the time-measured method of this project are similar to what F. Merlo et al. [2] pointed out in their articles. However, the dynamic time scaling is for the functional trajectories while this project is for the movement trajectories, which reflects the fact that this report suggests some possible approaches to areas that remain to be explored.

4.4 Conclusion

This project is about the creation of a navigation desktop robot with edge avoidance, which is mainly based on infrared sensors and the signal values from them. Through this project, the functions and features of infrared sensors and photo interrupter sensors have been learnt. In other words, the practical applications of these two optical-type sensors were explored in this project. However, this project encounters many expected errors including human errors and experimental errors. To be more specific, the human errors are mainly caused by misunderstanding of the functions of components while the experimental errors are caused by the sensors and jump wires, which cannot be fixed by the current authors' knowledge base. Despite these problems and errors, the project achieves the desired results to some extent, although there are still some unfinished tasks such as running the robot on different coloured table tops and achieving the same goals.

References

[1] J. Terra, "The Future of Robotics: How Robots Will Transform Our Lives," Simplilearn, Feb 7, 2023, Accessed: May 4, 2023. [Online]. Available: https://www.simplilearn.com/future-of-robotics-article

[2] F. Merlo, G. Vazzoler and G. Berselli, "Eco-programming of industrial robots for sustainable manufacturing via dynamic time scaling of trajectories," *Robotics and Computer-Integrated Manufacturing*, vol. 79, Feb. 2023 Art. no. 102420, doi: 10.1016/j.rcim.2022.102420.

[3] Arduino, "UNO R3," *Arduino Documentation*, Accessed: May 4, 2023. [Online]. Available: https://docs.arduino.cc/hardware/uno-rev3

[4] L. Ada, "Overview" Adafruit Motor Shield V2, July 08, 2013. Accessed: May 4, 2023. [Online]. Available: https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino

[5] "IR Sensor Working and Applications," ROBU.IN, https://robu.in/ir-sensor-working/ (accessed May 4, 2023).

[6] "What is a Photointerrupter?" *ROHM*. Accessed: May 4, 2023. [Online]. Available: https://www.rohm.com/electronics-basics/photointerrupters/what-is-a-photointerrupter

[7] J. O'Brien-Carelli, "Assemble the Chassis," Tri-Layer Mini Round Robot Chassis Kit, Oct. 26, 2016. Accessed: May 5, 2023. [Online]. Available: https://learn.adafruit.com/tri-layer-mini-round-robot-chassis-kit/assemble-the-chassis

[8] S. Suresh, *How to Make Arduino Based Edge Avoiding Robot using IR Sensor,* (June 29, 2019). Accessed: Feb. 2, 2023. [Online Video]. Available: https://www.youtube.com/watch?v=d7J0Bb78y2s&t=388s

Appendices

Appendix A

This appendix is for the project management formats.

A.1 Role allocation/responsibility matrix

Table 2. Role Allocation (Created by Nour Almosilli)

	Member Name	**Title(s)**
1	Nour Almosilli	• Project manager • Technical writer • Implementer
2	Tom Roberts	• Technical writer • Designer • Implementer
3	Tianxing JI	• Designer • Implementer
4	Abdulmalik Almarhoubi	• Designer • Implementer
5	Yuhang Long	Absent for the whole project

Table 3. Responsibility Matrix (Created by Nour Almosilli)

Title	**Project Activity**				**Deliverables**			
	Requirements/ Scope	**Design**	**Implementing**	**Testing**	**Poster**	**Blog**	**Bench**	**Report**
Project Manager	R	C	S	S	S	S	S	S/R
Designer	C	R	S	S	S	S	S	S
Implementer/ Developer	C	S	R	S	C	C	R	S
Technical Writer	C	S	S	S	R	R	C	R

KEY:

R – Responsible (accountable) for completion of task. (Task can be delegated to this person.)

S – Supports task.

C – Requires communication about the task.

Table 4. Typical Roles (Created by Nour Almosilli)

Title	**Role**	**Responsibilities**
Project Manager	Responsible for developing, in conjunction with the supervisor, the project scope. The Project Manager ensures that the project is delivered on time and to the required standards.	• Managing and lead the project team. • Managing the coordination of the partners and the working groups.

Title	Role	Responsibilities
Designer	Designing the system (the circuit, the code, etc.).	• Creating the required block diagrams, circuit diagrams, flow charts, etc.
Implementer/Developer	Implementing the suggested design	• Connecting the systems/circuit • Writing the code
Technical Writer	Documenting the project progress and deliverables	• Recording and maintaining all meeting logs • Updating the logbook. • Creating project blog • Creating project poster • Writing project report

A.2 Contribution to project deliverables

Table 5. Contribution to Project Deliverables (Created by Nour Almosilli)

	Member name	Deliverable(s)	Comments
1	Nour Almosilli	1. Logbook 2. Poster 3. Components testing 4. Sensors positioning 5. Circuit diagrams and graphs 6. Report	Responsible for all the managements forms, component request forms, the risk assessment and the ethical approval forms. Created most of the poster and took part in planning hardware side of the project and testing its aspects.
2	Tianxing Ji	1. Code 2. Designing algorithms and cases	Responsible for designing the code and implementing relative methods in algorithms. Testing components and

		3. Testing and simulating components 4. Poster 5. Report	their response to the code and adjust the code accordingly. Written the code part on the poster and created a flow chart to illustrate the cases present in the code which is added to the blog.
3	Abdulmalik Almarhoubi	1. Testing components 2. Sensors positioning 3. Installing the chassis 4. Soldering pin headers 5. Report	Installed the chassis and took part in planning the positioning of the sensors by testing the validity of each plan. In addition to testing components and soldering the pin headers to the motor shield.
4	Tom Roberts	1. 3D design 2. Blog 3. Sensors positioning 4. Testing components 5. Report	Designed clips to install on the chassis and hod the sensors into position. Created and updated the blog continuously. Tested components and took part in planning the positions of the sensors.

A.2 Attendance record

Table 6. Attendance Record (Created by Nour Almosilli)

	Member name	**Attended the weekly meeting and the lab? (Yes/No)**					**Comments**
		Week 1	**Week 2**	**Week 3**	**Week 4**	**Week 5**	
1	Nour Almosilli	YES	YES	YES	YES	YES	
2	Abdulmalik Almarhoubi	YES	YES	YES	YES	YES	
3	Tom Roberts	YES	YES	YES	YES	YES	
4	Tianxing Ji	YES	YES	YES	YES	YES	
5	Yuhang Long	NO	NO	NO	NO	NO	

A.2 Supervisor weekly meeting log

Year 2 Project (ELEC222/273) – *Supervisor meeting* – *Week 1*

Date: 30/1/2023 Supervisor: Dave Mclntosh

Project Title: Desktop navigation robot

Student Names /Attendees:	1. Nour Almosilli	2. Tianxing Ji
3. Tom Roberts	4. Abdulmalik Almarhoubi	5.

Summary of week's activities:

We started off by familiarising ourselves with the components that we ordered by researching about them and reading their datasheets. Then we planned the positioning of the sensors on the robot. We established the working mechanism of the sensors and identified the inputs that will be detected by the sensors, and we decided how the sensors will act upon these inputs.

Problem, issues and concerns:

We decided to implement trigonometry in the code to make the robot's turns as accurate as possible. This will be achieved by calculating how many degrees it will need to make a turn. Trigonometry will also be used to make the robot maintain a straight path when there is no need for any turns (i.e. when the robot does not face an edge or a corner). To achieve this, we need to establish the relative positions of the sensors.

Tasks for next week/Actions for next meeting:

Research how to implement the TT motor encoders and the photo interrupter sensors in our project.

Research trigonometry that would best suit the purposes of our project.

Supervisor use only

Progress Assessment: □ Unsatisfactory □ Satisfactory □ Good Comments/Recommendations: Keep going and well done so far.

Year 2 Project (ELEC222/273) – *Supervisor meeting – Week 2*

Date: 6/2/2023 Supervisor: Dave Mclntosh

Project Title: Desktop navigation robot

Student Names /Attendees:	1. Nour Almosilli	2. Tianxing Ji
3. Tom Roberts	4. Abdulmalik Almarhoubi	5.

Summary of week's activities: We have installed the robot chassis. Then we tested the conditions that might affect the strength of the infra-red signal and recorded the results. From these results, we established the ideal working conditions of the sensors, and we adjusted the prototype environment according to some of these conditions.

Problem, issues and concerns: According to our research, we established that the Arduino needs 7V power source to operate. However, the motors need a power source with a minimum of 6V, 2-3V

for each motor. Therefore, we will use a DC power source to power the motors, and 9V battery will be used to power the Arduino.

Tasks for next week/Actions for next meeting:

Further investigation will be carried to finalise the positioning of the infra-red sensors and their quantity.

The batteries, Arduino and motor shield will be installed on the chassis.

We will write the algorithm that will allow the robot to cover the area of the desk.

Supervisor use only

Progress Assessment: □ Unsatisfactory □ Satisfactory □ Good

Comments/Recommendations:

Problem with Arduino- as yet undiagnosed- new one ordered.

Year 2 Project (ELEC222/273) – *Supervisor meeting – Week 3*

Date: 16/2/2023 Supervisor: Dave Mclntosh

Project Title: Desktop navigation robot

Student Names /Attendees:	1. Nour Almosilli	2. Tianxing Ji
3. Tom Roberts	4. Abdulmalik Almarhoubi	5.

Summary of week's activities: One method to calculate the distance travelled by the robot is to set the Arduino clock to record the time needed to travel along one full

side of the desk and then using the speed of the motor to calculate the distance travelled by the robot. If this method is implemented in our project. The wheel encoders will not be needed anymore. However, this method reduces the accuracy of measurements.

Problem, issues and concerns: There are two methods to establish how much 90 degrees is. The first method is to measure the time needed to make a 90 degrees turn at one of the corners using the guidance of the infra-red sensors, so when the robot is covering inner area of the table it would spend the time determined at the corner earlier to turn 90 degrees. The second method is to count the number of notches it takes to make a 90 degrees at one of the corners and then set the robot to use the same number of notches every time a 90 degrees turn is needed.

Tasks for next week/Actions for next meeting: Connect the Arduino and the motor shield using the new pin headers and fix the sensors into their positions to run a final test. After testing individual aspects of the project (i.e.IR sensors, photo interrupter and the motors) we will fix any issues that may arise after the tests and when every part works accordingly, we will run a whole test which will test the code and the components together.

Supervisor use only

Progress Assessment: □ Unsatisfactory □ Satisfactory □ Good ✓

Comments/Recommendations:

Last week it was established that the Arduino problem (repeatedly blown+ reordered) was due to the method of connecting the motor shield - new pin headers ordered but not yet arrived. Important to learn, as a group, how to manage a project e.g. when there is a holdup with one work package, transfer your efforts to another that does not directly depend on it. When code apparently depends on the motor shield, you can actually insert dummy [fake] outputs (e.g. printing to the terminal) to test the various parts of the code.

Year 2 Project (ELEC222/273) – *Supervisor meeting – Week 4*

Date: 23/2/2023 Supervisor: Dave Mclntosh

Project Title: Desktop navigation robot

Student Names /Attendees:	1. Nour Almosilli	2. Abdulmalik Almarhoubi
3. Tianxing Ji	4. Tom Roberts	5.

Summary of week's activities: We have designed clips to hold the sensors in their positions and we will print them using 3d printer. We will solder the new pin headers to the motor shield and then connect it to the Arduino to check the connection. We will fix the batteries, the Arduino that is connected to the motor shield and the mini breadboard into their positions on the chassis so the robot is ready to function. We finished off the poster and we will print it out, so it is ready for the inspection day.

Problem, issues and concerns: We soldered the pin headers to the motor shield incorrectly so we attempted to desolder it and clean the motor shield's holes from the solder and resolder it correctly. This proved very challenging so the lab technicians helped us by asking the workshop to complete the job for us. As a result, we re-established the connection between the motor shield and the Arduino. When we run the full code, the robot followed the edges without falling off but when a full turn has finished the robot moved randomly instead of covering the rest of the desk. It is also difficult to test the code because the IR sensors are not firmly fixed.

Tasks for next week/Actions for next meeting: Fix the 3d printed clips into position and attach the sensors to the chassis. Change the code the robot covers the rest of the desk.

Supervisor use only

Progress Assessment: □ Unsatisfactory □ Satisfactory ☑ Good

Comments/Recommendations: It was a learning experience to figure out the correct way to solder on a stackable extra-long pin header. As supervisor I should have given you more guidance for this. However, you have recovered well to achieve some solid progress, albeit with more of your own time invested than would have been ideal. The prototype clips look good and I hope they can be finished in time for attaching the sensors. In the meantime, please test the 'table coverage' code another way (e.g. with summary [fake] inputs/outputs.)

Appendix B

This appendix is for a breakdown of individual contributions to the project.

Appendix C

This appendix is for the code in this project, which is written by Tianxing Ji.

Code of Navigation Desktop Robot with edge avoidance

#include <Wire.h>

#include <Adafruit_MotorShield.h>

#include "utility/Adafruit_MS_PWMServoDriver.h" // just by default for input library, no meaning

Adafruit_MotorShield AFMS = Adafruit_MotorShield(); // communicate with arduino with I2C, by default, no meaning

// Or, create it with a different I2C address (say for stacking)

// Adafruit_MotorShield AFMS = Adafruit_MotorShield(0x61);

Adafruit_DCMotor *leftMotor = AFMS.getMotor(2); // *********3 means M3 on the motor shield, need to change before running*********

Adafruit_DCMotor *rightMotor = AFMS.getMotor(3); // *********2 means M2 on the motor shield, need to change before running********

//left and right is not compulsory, we can change them at any time

// The below is all the data and method we need

int Right_Trace_sensor_1 = 8; // ******the first IR sensor in the right side, need to change before running********

int Left_Trace_sensor_1 = 9; // ********This is the first IR sensor in the left side of the car connected to the Arduino. Need to change before running**********

int Left_Trace_sensor_2 = 10; // ********This is the first IR sensor in the left side of the car connceted to the Arduino. Need to change before running**********

// Why needs change? *** Because the number they equal is the digital pin they connected to the arduino, they may be different.

void Forward(); // move forward

void Backward(); // move backward

void turnRight(); // turn right

void turnLeft(); // turn left

void Stop(); //stop

//all of them above should be included in the method Automatic_mode

```
void Automatic_mode_edge();

void Certain_mode();

void Automatic_mode_random();

// end of the movement of the car




// start of Photo Interrupter Sensor

//int pI = 12; // ********the number of pin connecting the sensor, need to be changed before running this code************

//int pI2 = A1; // ********the number of pin connecting the sensor, need to be changed before running this code************

//

//int val; // calibration Value

//int val2; // for the second sensor

//

//long last = 0; // used in timing, it holds the last millisecond

//long last2 = 0; // for the second sensor

//

//int stat = LOW; // used in calibration

//int stat2 = HIGH; // used in calibration, combined with stat to detect the change between sensors

//

//int stat3; // for the second sensor

//int stat4 = HIGH; // for the second sensor

//

//int counter = 0; // used to count light/dark iterations

//int counter2 = 0; //for the second sensor
```

```
//

//int sens = 1; // calibration value, when the sensor is connected to analogue pin, it outpus value from 0 to 1000.

//int slots = 20; // *******the number of slots in disc, need to count before running this code**************

//int millisecs = 1000; //time of sample


//void count (); //count the resolution


// end of the calculation of the RPM



// for the edge dectection， edge is used to remember all the edges, we only need edge [3], edge [4], edge [5], edge [6]. And for convenience, we don't care about edge [0]

long edge_length[2]; // store the tinme value of two edges

int edge_count = 0; // count the number of edges the robot travelled

long lastTime[5]; // be set to remember the time and then calculate for the distance it travels

bool edge_find = 0; // a boolean value for the state determination of wheather the car finds an edge or not

long timeForRightAngle[2]; // it is used to test out the time of turn a 90 degree

// end of the detection of four edges

long turn90right = 592; // This is the perfect time to turn a 90 degree

int counter90 = 8;


void setup() {

  // put your setup code here, to run once:

  Serial.begin(9600);           // set up Serial library at 9600 bps
```

```
  // set of each pinMode for sensors, but not the analogue ones because they don't need to set input , they are default by input
  pinMode(Left_Trace_sensor_1, INPUT);
  pinMode(Left_Trace_sensor_2, INPUT);
  pinMode(Right_Trace_sensor_1, INPUT);
  // pinMode(pI, INPUT);
  // set them all be input and then the arduino will be able to read all of the IR sensors we connect

  if (!AFMS.begin()) {         // create with the default frequency 1.6KHz
  // if (!AFMS.begin(1000)) {  // OR with a different frequency, say 1KHz
    Serial.println("Could not find Motor Shield. Check wiring.");
    while (1);
  }
//
  Serial.println("Motor Shield found.");
  leftMotor->setSpeed(65);
  rightMotor->setSpeed(65);

  // Set the speed to start, from 0 (off) to 255 (max speed)
}

void loop() {
//
// The start of testing movement of the robot
//  Forward();
//  Serial.println("turnRight");
//  turnRight();
```

```
//  delay(turn90right);
//  The end of testing movement of the robot
//
// Test for the combination of motor movements and PI sensors
//  Stop();
//  delay(2000);
//  Serial.println(counter);


//Serial.println(digitalRead(pI));
//  Serial.println("This is the digital read");
//  Serial.println(digitalRead(pI));
//  Serial.println("This is the digital read");
//The end of the test for PI sensors and motor movements
//
// The start for PI sensors to turn right 90 degrees
//  if(counter == 0){
//  timeForRightAngle[0] = millis();
//  Serial.println("Start Test");
//  }
//
//  while(counter < counter90) {
//   Serial.println("start counter");
//   count();
//   turnRight();
//  }
//  if(counter == counter90){
//    counter = 0;
```

```
//    Stop();
//    delay(5000);
//  }

// if(counter == counter90){
// timeForRightAngle[1] = millis();
// Stop();
// Serial.println("This is the time for turning right angle");
// long timeRightAngle = timeForRightAngle[1] - timeForRightAngle[0];
// Serial.println(timeRightAngle);
// Serial.println("This is the end of time for turning a right angle");
// counter = 0;
// delay(5000);
//}
// Then end of test for PI sensor to turn right 90 degrees

//
  if(edge_count < 5){  // Key command to this project
    Adutomatic_mode_edge(); // if edge_count is less than 5, then conduct this
  }
  if(edge_count == 5){
    Certain_mode(); // if edge_count is eaqual to 5, then conduct this
  }
  if(edge_count > 5){
    Automatic_mode_random(); // if edge_count is more than 5, conduct this
  }
//
```

```
//  Backward();  // used to test move backward
//  delay(200);
//  Stop();
//  delay(1000);



//   if(counter < 5){       // used to test whether the condition loop is right or not
//    for(int i = 0; i < 1; i++){
//    Serial.println("Hello world!");
//    counter ++;
//   }
//    delay(1000);
//    Serial.println("false");
//   }else if(counter == 5){
//    Serial.println("true");
//    delay(1000);
//    counter++;
//   }else if(counter > 5){
//    Serial.println("No need");
//    delay(1000);
//   }


}


// This is for the edge detection for edge_count < 8
void Adutomatic_mode_edge(){
```

```
  int stateValue = 0; // represent the state of the car by the value of integers calculated by IR sensors


  stateValue = 26 * digitalRead(Left_Trace_sensor_1) + 16 * digitalRead(Left_Trace_sensor_2) + 8 * digitalRead(Right_Trace_sensor_1) + edge_find;


  switch (stateValue) // match the different movement to different state values, which represent the state of the car


  //have obstacle -> digitalRead is 0, otherwise 1
  {
    //when the robot does not detect the edge
    case 0: // This case must exist
    Serial.println("no edge now");
    Forward(); // go to find an edge now!!
    Serial.println("go straight, case 0");
    break;


    case 16: // This case must exist
    turnRight();
    delay(200);
    Serial.println("find edge now"); // Great! you find an edge
    edge_find = 1; // make the boolean value equal to HIGH, then it will be into another pattern
    Serial.println("go straight along the road if it has an edge at first, case 16");
    break;


    case 42: // This case must exist
    Backward();
```

```
delay(200);

turnRight();

delay(100);

Serial.println("find edge now");

edge_find = 1; // it has been shown in case 16

Serial.println("turnRight after finding an edge and going out of the table, case 42");

break;


case 8: // This case must exist, however it is too difficult to determine the state after this, if we add 1 to it (make it in the other state, then it will be stuck at the same place)

Backward();

delay(200);

turnRight();

delay(750);

Serial.println("find edge now");

Serial.println("the right side of the car finds an edge, case 8");

break;


case 24: // This case must exist

turnRight();

delay(100);

Serial.println("find edge now");

Serial.println("the right side of the car finds an edge, case 24");

edge_find = 1;

break;


case 26: // This case must exist Because of the car architecture, this will not be defined as find an edge
```

```
    Backward();
    delay(100);
    turnRight();
    delay(200);
    Serial.println("case 26");
    break;

    case 34: // This case must exist
    Backward();
    delay(250);

    turnRight();
    delay(turn90right);

    Serial.println("find edge now");
    edge_find = 1; // nobody wants to know what this means, which has been shown in
case 16
    Serial.println("case 34");
    break;

    case 50: // This case must exist, hard to determine the value
    Serial.println("case 50");

    lastTime[edge_count] = millis();
    Serial.println(lastTime[edge_count]);
    edge_count ++;

    Backward();
```

```
delay(140);
Serial.println(edge_count);

turnRight();
delay(turn90right);



Serial.println("find edge now");
edge_find = 1; // it has been shown in case 16

break;

// the above is the progress to find edge, there may be more scenarios**********
And these cases exist for 100%, human can place them like that or just be created by case 0

// the below is the progress after find the edge.

case 1: // This case must happen from case 17 if the direction is a little bit to
right//must make other cases to skip this
Backward();
delay(80);
turnLeft();
delay(70);
Serial.println("case 1, you are going out of the table");
break;

case 17: // This is the perfect case that represents that the robot is on its way
Forward();
```

```
Serial.println("case 17, you are fllowing the edge, great");

break;


case 43: // This is the opposite case of case 1, this must happen from case 17 is the direction is little bit to left

Backward();

delay(200);

turnRight();

delay(100);

Serial.println("case 43, you are going left side, should correct your way");

break;


case 25:  // This will happen when the two front IR sensors are not detected at the same time, this will make an extra compensation for this scenario, otherwise, I don't know how it can happen*****

Forward();

Serial.println("case 25, No! No! you should wait for the left first IR sensor");

break;


case 27:  // This will happen when the car turns right from case 34, it should make the car continue to turn right

turnRight();

delay(200);

Serial.println("case 27, how can it happens?");

break;


case 51: // This must exist

Serial.println("case 51, the most important case.");

lastTime[edge_count] = millis();
```

```
Serial.println(lastTime[edge_count]);
edge_count ++;
Serial.println(edge_count);

Backward();
delay(140);

turnRight();
delay(turn90right);
break;

//Below are some strange cases that can will happen only when unexpected cases,
therefore, we should reset the parameter to reload the data

case 35: // This may also be caused by the case when the robot has not found an edge.
Stop(); // stop and then check the sensors
edge_count = 0; // restart to run again
delay(1000);

turnRight();
delay(turn90right);

Serial.println("case 35, I don't how can this senario happen");
break;
```

```
    case 9: // I just don't know how can this situation can happen********** This must be weak positions of IR sensors. But this is also a case that will happen, it is possible to set it to run again

    Stop();

    edge_count = 0;

    delay(1000);

    Serial.println("case 9, how can it happens? Jesus");

    break;


}

}



// A certain model for navigate the table

void Certain_mode() {



  if(edge_length[0] < edge_length[1]){

  edge_length[0] = lastTime[1] - lastTime[0] - turn90right*2; // it is the time measure of the edge, and substrated by 4/2 times turn90right

  edge_length[1] = lastTime[2] - lastTime[1] - turn90right*4;

  }else{

  edge_length[0] = lastTime[1] - lastTime[0] - turn90right*4; // it is the time measure of the edge, and substrated by 4/2 times turn90right

  edge_length[1] = lastTime[2] - lastTime[1] - turn90right*2;

  }


  Serial.println("This is the edge_length");

  Serial.println(edge_length[0]);
```

```
Serial.println(edge_length[1]);
Serial.println("This is the end of edge_length");

for(int i = 0; i< 2; i++){
  //edge 0
  Forward();
  delay(edge_length[0]/4*(3-2 * i));

  turnRight();
  delay(turn90right + 10);

 //edge 1
  Forward();
  delay(edge_length[1]/4*(3- 2 * i));

  turnRight();
  delay(turn90right + 10);

 //edge 2 navigation
  Forward();
  delay(edge_length[0]/4*(2- 2 * i));

  turnRight();
  delay(turn90right + 10);

//edge 3 navigation
  Forward();
```

```
    delay(edge_length[1]/4*(2- 2 * i));


    turnRight();
    delay(turn90right + 10);
  }


  edge_count ++;


}



// This is for the edge_count >= 8
//This is the core part of this code. Combine the IR sensors and two motors together without Photo Interrupter Sensor in this part.
void Automatic_mode_random() {


  int stateValue = 0; // represent the state of the car by the value of integers calculated by IR sensors


  stateValue = 24 * digitalRead(Left_Trace_sensor_1) + 8 * digitalRead(Right_Trace_sensor_1);


  stateValue = random(stateValue, stateValue + 3); // get a random value fromm stateValue and stateValue+3 ( stateValue+3 is not included), this will make the robot have more reactions.


  switch (stateValue) // match the different movement to different state values, which represent the state of the car


  //have obstacle -> digitalRead is 0, otherwise 1
```

{

case 34: // *******The front part of the car is out, take action to correct the trajectory, may have a better idea, need to change**********

Backward();

delay(500); // give the car 500 ms, which is 0.5 seconds, to move backward

turnRight();

delay(800); // give 0.8 seconds to the car to turn right

Forward();

delay(100); // go forward

break;

case 33: // *******The front part of the car is out, take action to correct the trajectory, may have a better idea, need to change**********

Backward();

delay(300); // give the car 300 ms, which is 0.3 seconds, to move backward

turnRight();

delay(600); // give 0.6 seconds to the car to turn right

Forward();

delay(100); // go forward

break;

case 32: // *******The front part of the car is out, take action to correct the trajectory, may have a better idea, need to change**********

Backward();

delay(150); // give the car 150 ms, which is 0.15 seconds, to move backward

turnRight();

delay(300); // give 0.3 seconds to the car to turn right

Forward();

delay(100); // go forward

break;

case 26: // *******The upper left corner of the car is slightly out of the table, take action to correct the trajectory, may have a better idea, need to change**********

Backward();

delay(300); // give the car 300 ms, which is 0.3 seconds, to move backward

turnRight();

delay(500); // give 0.5 seconds to the car to turn right

Forward();

delay(100); // go forward

break;

case 25: // *******The upper left corner of the car is slightly out of the table, take action to correct the trajectory, may have a better idea, need to change**********

Backward();

delay(200); // give the car 200 ms, which is 0.2 seconds, to move backward

turnRight();

delay(400); // give 0.4 seconds to the car to turn right

Forward();

delay(100); // go forward

break;

case 24: // *******The upper left corner of the car is slightly out of the table, take action to correct the trajectory, may have a better idea, need to change**********

Backward();

delay(100); // give the car 300 ms, which is 0.3 seconds, to move backward

turnRight();

delay(200); // give 0.5 seconds to the car to turn right

Forward();

delay(100); // go forward

break;

case 10: // *******The upper right corner of the car is slightly out of the table, take action to correct the trajectory, may have a better idea, need to change**********

Backward();

delay(300); // give the car 300 ms, which is 0.3 seconds, to move backward

turnLeft();

delay(500); // give 0.5 seconds to the car to turn right

Forward();

delay(100); // go forward

break;

case 9: // *******The upper right corner of the car is slightly out of the table, take action to correct the trajectory, may have a better idea, need to change**********

Backward();

delay(200); // give the car 200 ms, which is 0.2 seconds, to move backward

turnLeft();

delay(400); // give 0.4 seconds to the car to turn right

Forward();

delay(100); // go forward

break;

```
    case 8: // *******The upper right corner of the car is slightly out of the table, take action to correct the trajectory, may have a better idea, need to change**********
    Backward();
    delay(100); // give the car 100 ms, which is 0.1 seconds, to move backward
    turnLeft();
    delay(200); // give 0.2 seconds to the car to turn right
    Forward();
    delay(100); // go forward
    break;

    default:
    Forward();
    break;

  }

}

//the end of the Automatic_mode method



// there are five basic ways to move the car, which are followed as below. And we can set their speed in other mehods i.e Automatic_mode()
void Forward() {
  leftMotor->run(FORWARD);
  rightMotor->run(FORWARD);
}
```

```
void Backward() {
  leftMotor->run(BACKWARD);
  rightMotor->run(BACKWARD);
}

void turnRight() {
  leftMotor->run(FORWARD);
  rightMotor->run(BACKWARD);
}

void turnLeft() {
  leftMotor->run(BACKWARD);
  rightMotor->run(FORWARD);
}

void Stop() {
  leftMotor->run(RELEASE);
  rightMotor->run(RELEASE);
}

// end of the mothods of movements

//void count() {
//    val = digitalRead(pI); // read the value come from pI snesor 1
//
//    if(val < sens) // something has been in the gulf of pI
```

```
//    stat = LOW; // set the state to be LOW to represent the status
//    else //if nothing
//    stat = HIGH; // set the state to be HIGH to represent the status
//
//    if(stat2 != stat) // if there is a change, because state is HIGH by default, meaning nothing between the gulf.
//  {
//    counter ++; // need to add 1 to counter
//    stat2 = stat; // set for the next dectect of new change
//  }
//
//}
```