ELEC1601 Project Report

ELEC1601 Team Report

SIDs:
500516975
500552548
500216307
500220500
500136245
500478204
Group: W11A 10

Faculty of Engineering
Nov, 2020

ELEC Team Report

1. Introduction

For our project we have chosen to undertake Task 2: Maze Navigation Competition. Our attempt at completing this task consisted of placing primary importance to implementing the initial and secondary design goals first and then choosing whether or not to complete the tertiary design. Our design process (further detailed) underwent many challenges and attempts at trying new implementation techniques and testing until we were able to achieve the initial and secondary goals. The final prototype of our robot is designed to receive commands via bluetooth and begin to move through a maze by making its own decision in its movement by detecting the walls around it. It is programmed to handle many different turning situations, including T- intersections, right turn, left turn and rotating 180 degrees if it reaches a dead end. Once it finally reaches the desired object (ping pong ball), it will pick it up using an externally built arm attached to a servo motor, and proceed back to the original starting position. Further features for the secondary design goal were added to the robot where it can store the route of the maze (or any scenario) in its memory during manual control and can use this data to redo the entire route automatically. Our robot design can also satisfy most of the tertiary requirements, although this required was still incomplete.

2. Project Processes

2.1. Details about how the project was managed

Our group communicates through a social tool called WeChat, and for this task we created a group. In it, we can send messages and related pictures to exchange information. In addition, We had 5 meetings, including 4 zoom meetings and one offline meeting. On November 3rd and November 10th, we had two meetings involving all members. Then, an offline meeting involving offline members was held on November 15. Then there was a zoom meeting on the 17th and 19th. In terms of announcements, we also posted our plans to Trello. We upload all the materials that we use, are using and will be used in the task to our trello board. In addition, any design plans are uploaded to trello. We upload all the materials that we use, are using and will be used in the task to our trello board. In addition, any design plans are uploaded to trello. Our group's trello is divided into 5 lists, the first is to do, the second is doing, the third is done, and the fourth and fifth are component list and meeting dates and times respectively.

2.2. Details about task division

For this task, after we fully understand the tasks that need to be performed through the zoom meeting, our team has made a clear division of labor. Since our task is to control the robot through Bluetooth communication, and let the robot realize straight-line entry, 90-degree turn, T-junction and side branches from the beginning. Then, Zhou Hongyu, Zhang Wenhan and Yang Hans built a 3D model for the ball pickup robot arm and the pickup box. Hans took charge of ping pong ball detection, and Jingyu Chen completed the review code and streamlined the sensor part. In addition to this, there is the general detection part (this is the task of letting the machine choose the other side in the case of the front and the right) Shoudi have completed the task of the robot part. Danush is working on storing robot movement in memory.Taking into account that it is difficult for online learning to cooperate on the spot, it is more about writing tasks.

2.3. details on project progress

During the task, our team encountered various problems. The first one was the replacement of the ir sensor. When we used the ir sensor, we found that it could not effectively identify the robot for steering and we chose to replace it with ultrasonic sensors as it is more accurate in distance measurement between obstacle and sensor. In addition, because the robot doesn't turn well every time when it turns 90 degrees, it won't go well every time when turning the maze to the right. When trying to stop it, it kept moving. When the power is on, the reason is that the motor on the wheels will vibrate. We disconnect the power to the motor when we need to stop, which solves the motor shaking problem. And Han encountered some problems during detection, so he made a prototype. But the wood also has a prototype. Don't distinguish when identifying. Later, deep learning was used to solve this problem. Finally, the ping pong ball is recognized. There are some robot training issues.

3. Materials

3.1. Arduino UNO Board

Arduino UNO board is an open-source platform based on Arduino IDE, which is able to read from and output to the analog pins. It's widely equipped in robots as the microcontroller.

3.2. Arduino Mega Board

Arduino Mega Board is a microcontroller which is compatible with most shields designed for the UNO board. In this project, it's connected to the JoyStick controller and receives commands from Bluetooth Master.

3.3. Raspberry Pi

Raspberry Pi is a series of small Linux-based single-board computers with more computation capacity. In this project, the streaming media server receives the video streaming from the camera to infer the obstacle with the model pre-stored in the streaming media server of Raspberry Pi.

3.4. Wire

It's used to connect all the components into a circuit which has negligible resistance.

3.5. Servo Motor:

Servo Motor is able to turn its shaft by a given angle through servo.write() function in Arduino. In the real-world application, servo motors are commonly used in robots to perform the precise angle of movement. For this project, only one servo motor is required to control the rotational angle of the robot carrier to lift up an object.

3.6. Ultrasonic Sensor

Ultrasonic sensor determines the distance of an object via sonor. It's not affected by sunlight or black material. Ultrasonic devices are widely applied to detect invisible flaws in products. In this project, it was chosen to identify different types of corners (90-degree turn, T-conjunction, side branch, etc).

3.7. JoyStick controller

A JoyStick controller consists of a stick which can rotate and pivot for a certain angle. It's an input device which reports its angle to the device it's controlling. It is commonly used in the cockpit of many military aircrafts. In this project, it was used to implement the remote control of manual navigation via Bluetooth.

3.8. Camera

The specification of the camera in this project is 8fps. It transmits the immediate image with bias of 0.125s to the streaming media server of Raspberry Pi, to help it infer the obstacle according to the models stored in Raspberry Pi.

3.9. HM-10 Bluetooth Module

The HM-10 Bluetooth Module is a serial 4.0 BLE module which has low power consumption. It is widely used for wireless communication via cell phones and other electrical equipment. In this project, two HM-10 Bluetooth Modules are used, Master and Slave, which are connected to Mega board and UNO board correspondingly. In this project, HM-10 Bluetooth Modules implement the whole process of bluetooth communication, such as manual navigation in the maze,etc.

3.10. BOE Shield-Bot

BOE Shield-Bot is a mobile programmable robot which is based on Arduino technology. In the real-world application, it's a prototype of many robots using the Arduino environment and Arduino UNO board. In this project, it's the robot based on which we equip with many external components and implement the commands from the Arduino IDE.

3.11. GPU Server

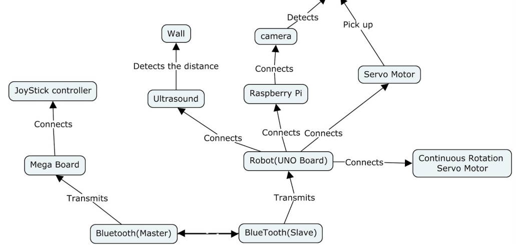
The CUDA based GPU is used for object detection model training and inference. In this project, we use it to inference the object bounding box from video flow.

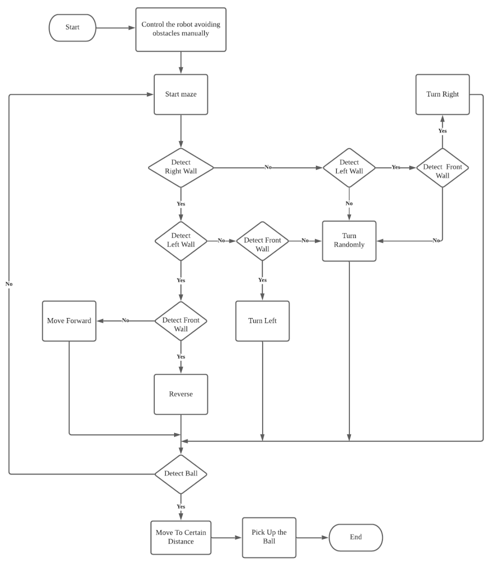
3.12. Faster-RCNN Toolkit

Faster-RCNN is a wide-used detection model in the deep learning industry with incredible performance. We used the Faster-RCNN model which pre-trained on Microsoft COCO dataset, and re-trained on our own annotated Ping-pong dataset.

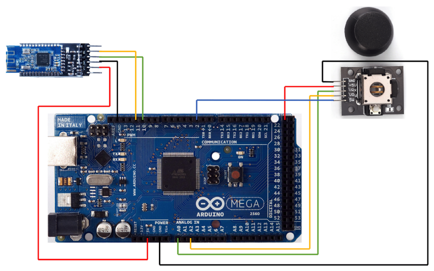
4. Implementation/Analysis

4.1. Flow chart representation

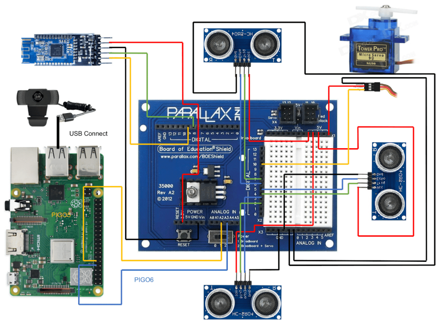




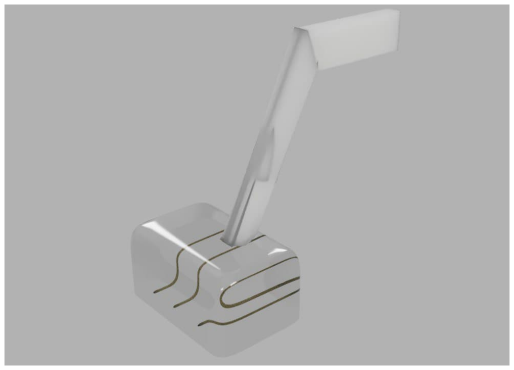
4.2. Master Board Circuit Drawing



4.3. Robot Circuit (Slave board) Drawing



4.4. Robotic arm 3D model



4.5. Pseudo code

Algorithm 1 Master and Slave Board Set up

```
1: Connect Arduino Mega board, joystick controller and HM-10 master Bluetooth with in a circuit as Master board circuit drawing indicate for Master board set up.
2: Connect three ultrasonic sensors, slave Bluetooth, two parallax continue rotation motor and one servo motor with in a circuit as Robot circuit Drawing.
```

Algorithm 2 Master and Slave Bluetooth Set up

```
1: Create master and slave Bluetooth object with “SoftwareSerial” library.
2: Redirect Serial Monitor and Bluetooth object’s output to each other.
3: Use AT command “AT+ROLEx” to set master and slave Bluetooth (Master: x=1; Slave: x=0)
4: Use “AT+MODE0” command to set master Bluetooth in auto connection mode.
```

Algorithm 3 Robot Manual Control

```
1: Use Joystick controller sent digital signal from 0 to 1023 in x and y axis to master board which correspond to forward, backward, left and right turn.
2: Redirect digital value to slave board with Bluetooth connection.
3: Slave board move the robot according to the x and y digital value with function “.writeMicroseconds()”. (Eg. y=0, Robot move forward)
4: Delay 100 microseconds for one move
5: Slave board sent the finished move back to Master board. (Eg. “F” for forward, “R” for right)
```

Algorithm 4 Manual Control Memory

```
1: Master board receive the finished move.
2: If finished move == stop (S), jump to step 12.
3: Else if the finished move == last move, last move count += 1.
4: Else if finished move != last move, append last move in the move instruction list and the last move count to the move count list. Set last move = finished move. (The last move of manual control will be appended when auto detection for Ping-Pong ball though maze start)
5: Repeat Robot Manual Control step 1 to Manual Control Memory step 4 until start auto detection or start return or restart process.
```

Algorithm 5 Maze Navigation

```
1: Receive string “A” from master board’s serial monitor.
2: Sent message to slave board to indicate maze navigation start though Bluetooth connection
3: Detect distance between wall and robot in three direction (Front, Right, Left) with ultrasonic sensor.
4: If detected distance is less than minimum allowing distance indicate there are wall nearby in that direction.
5: Append the available directions for next move to a list and randomly chose one with function “random()”
6: Slave board move the robot according to the chose direction.
7: Delay for a suitable time. (Front: 200, Left and Right: 800)
```

Algorithm 6 Maze Navigation Memory

```
1: Repeat Manual Control Memory step 1 to 4 with different move count list and move instruction list for auto detection.
2: Repeat Maze Navigation Step 2 to Maze Navigation Memory Step 1 until Ball Found.
```

Algorithm 7 Ball Detection

```
1: If ball found but far away, robot move forward, do step 2, 3, 4. Else, jump to 5.
2: Delay 200
3: Slave board sent finished move “F” back to master board.
4: Repeat step 8, 9, 10, 11 with different move count list and move instruction list for auto detection.
5: If ball found in front of robot, robot stop moving and slave board send ball found massage back to master board and robot back to manual control state. Jump to Robot Manual Control step 1.
6: Repeat step 1 to 5.
```

Algorithm 8 Pick Up Ball

```
1: Receive String “P” from master board’s serial monitor.
2: Master board send massage to slave board indicate Pick Up process start.
3: Slave board set servo motor’s angle to 0 (lay down the ball collector).
4: Delay 3000 microseconds.
5: Reset servo motor’s angle back to 130 (pull up the ball collector).
6: Jump to Robot Manual Control step 1.
```

Algorithm 9 Restart process fully automatically

```
1: Receive string “S” from master board’s serial monitor.
2: Master board generate suitable x, y value according to the direction in the move instruction list.
3: Send the x, y value continuously to the slave board until it match the move count in the move count list with same index.
4: Repeated step 2, 3 with index plus one until the last move in the list.
5: Jump to Robot Manual Control step 1.
```

Algorithm 10 Return to Starting Point

```
1: Receive string “B” from master board’s serial monitor.
2: Master board generate suitable x, y value according to the direction in the move instruction list in reverse order.
3: Send the x, y value continuously to the slave board until it match the move count in the move count list with same index.
4: Repeated step 2, 3 with index minus one until the first move in the list.
5: Jump to Robot Manual Control step 1.
```

Algorithm 11 Reflash Memory

```
1: Receive string “E” from master board’s serial monitor.
2: Empty the move instruction list and move count list for manual control and auto detection with the function “memset()”.
3: Jump to Robot Manual Control step 1.
```

Algorithm 12 Raspberry Pi Processing

```
Require: Webcam video $V_w$; Socket signal $S_s$; Motion video toolkit $M()$;
Ensure: GPIO signal $S_g1$ GPIO signal $S_g2$
 1: $V_r \leftarrow M(V_w)$ // Resize to 320×280
 2: $V_f \leftarrow M(V_r)$ // Fix frame rate to 8 fps
 3: $V_p \leftarrow M(V_f)$ // Public video flow to network
 4: repeat
 5:     update $S_s$
 6:     $m \leftarrow \delta(S_s)$ // Decode to string from socket signal
 7:     if $m$ = ”Existfar” then
 8:         $S_g1 \leftarrow 1$
 9:     else
10:         $S_g1 \leftarrow 0$
11:     end if
12:     if $m$ = ”Existpickup” then
13:         $S_g2 \leftarrow 1$
14:     else
15:         $S_g2 \leftarrow 0$
16:     end if
17: until break
```

Algorithm 13 Backbone Server Processing

```
Require: Network video flow $V_n$; Trained faster-rcnn model $M_f$;
Ensure: Socket signal $S_s$
 1: $V_t \leftarrow \epsilon(V_n)$ // Pre-processing
 2: $(C_v\ B_v\ P_v) \leftarrow M_f(V_t)$ // Inferenced and get the first output tuple(class, bounding box, confidence)
 3: $V_p \leftarrow M(V_f)$ // Public video flow to network
 4: repeat
 5:     if $C_v$ = ”Pingpong” then
 6:         if $P_v \geq 0.9$ then
 7:             $S_p$ append ”Exist”
 8:             $B_a \leftarrow \beta(B_v)$ // Calculate the area of bounding box
 9:             $S_e \leftarrow 6000 / B_a$ // Calculate the estimated distance
10:             if $S_e \geq 15$ then
11:                 $S_p$ append ”far”
12:             else if $S_e \leq 14$ then
13:                 $S_p$ append ”pickup”
14:             end if
15:         end if
16:     else
17:         $S_p$ append ”noball”
18:     end if
19:     $S_s \leftarrow \varphi(S_p)$ // Encode string and send by socket
20: until break
```

4.6. Description of the final prototype

Final prototype Robot contain six main components. HM-10 slave Bluetooth allow the communication between the robot's Arduino UNO board and the Master board (Arduino Mega board) with master Bluetooth. Two parallax continuous rotation motors control the speed and direction of the rotation of wheels. Three ultrasonic sensors detection the distance between walls and the robot in order to determine the available direction for maze navigation. One servo motor control the movement of Self-designed ball collector for ball picking up. Raspberry Pi was used as a client, and camera was used for record video flow. This two components works together to send video flow to GPU server to detect object bounding box by faster-rcnn deep learning model.

5. Real-World Applications

As the design of our robot is versatile, it can be used for a range of other real-world applications. A few examples of such existing applications include:

5.1. Search and Rescue Robots

Search and rescue robots are used for a wide range of situations that require rescuers to collect as much information before being able to conduct a rescue operation. Scenarios such as earthquakes or people being trapped in caves would require such robots to enter the location and proceed to navigate its way until the victims are identified.

5.2. Bomb Detection Robots

These robots have the function of being able to navigate its way through a suspected bomb location and identify any type of explosive ordnance, before signaling back to the user and wait for further instructions. This robot has a similar design and programming to our robot, as they too have the feature of being manually controlled through a range of different terrain until it reaches its desired location and identifies the object.

6. Conclusions

Overall, our robot design was able to achieve the majority of the requirements from the initial and secondary design goals. The robot successfully managed to traverse through the given route manually, was able to receive and send data to and from the user via bluetooth, inform the user once the object is detected, and automatically pick it up. Furthermore, our robot was also able to automatically reset back to its original starting point. However, the design required could have been improved through further testing and improvements, in regards to making the robot automatically navigate its way through the maze.

7. Future Work

While performing our demonstration of our robot to our tutors, it was evident that the robot was not designed properly enough when it attempted to make it's turns as it began to loop back and forth between left and right turns. As the sensors were calibrated properly and working correctly, we realised that the issue stems from the turn logic of our code as well as the physical size of the robot. We noticed that as the robot begins to make a turn, the back end collides with the wall, causing the sensors to get incorrect readings due to its positioning. Given more time, our group would have been able to successfully improve and test this section of the code by making appropriate adjustments by accounting for the size of this robot, overall resolving this issue.

8. Appendices

8.1. Trello Board

https://trello.com/invite/b/BIjuP0wz/ecc19344f935b251bd3e1ce253ce16b1/project-design

8.2. Github Repo

https://github.com/Hansxsourse/Rapberry-Pi-Object-Detection

References