Programming "Things"

Zumo Search & Rescue Assignment

Connor Jenkins

Contents

Introduction

This document overviews the steps taken to complete the Programming "Things" zumo search & rescue assignment. In this document you will be able to preview five different sections. An overview of these sections has been provided below.

Search & rescue GUI & Usage: This section overviews the development choices made for the GUI, involving my choice of programming language and design choices. This section also overviews how to operate the GUI.

Search & Rescue Code: This section overviews the development choices made for the zumo Arduino code, along with any resources I may have used to assist my development.

Achievements: This section overviews which tasks I were able to succeed at, and how they were completed.

Complications: This section overviews which tasks I was unable to complete and why I couldn't achieve them. Any bugs that occur within the program and additional challenges are included here.

Bibliography: The bibliography includes references to any code or resources I used to assist in my development.

Search & Rescue GUI & Usage

The graphical user interface was developed in C# using the windows forms app (.net framework), within the visual studio IDE. The decision to use this programming language, stems from my familiarity with it, and the development of windows form apps using this programming language. In particular, the simplicity of the technology proves to be an addition benefit. Below is a full preview of the GUI, when no connection is made.

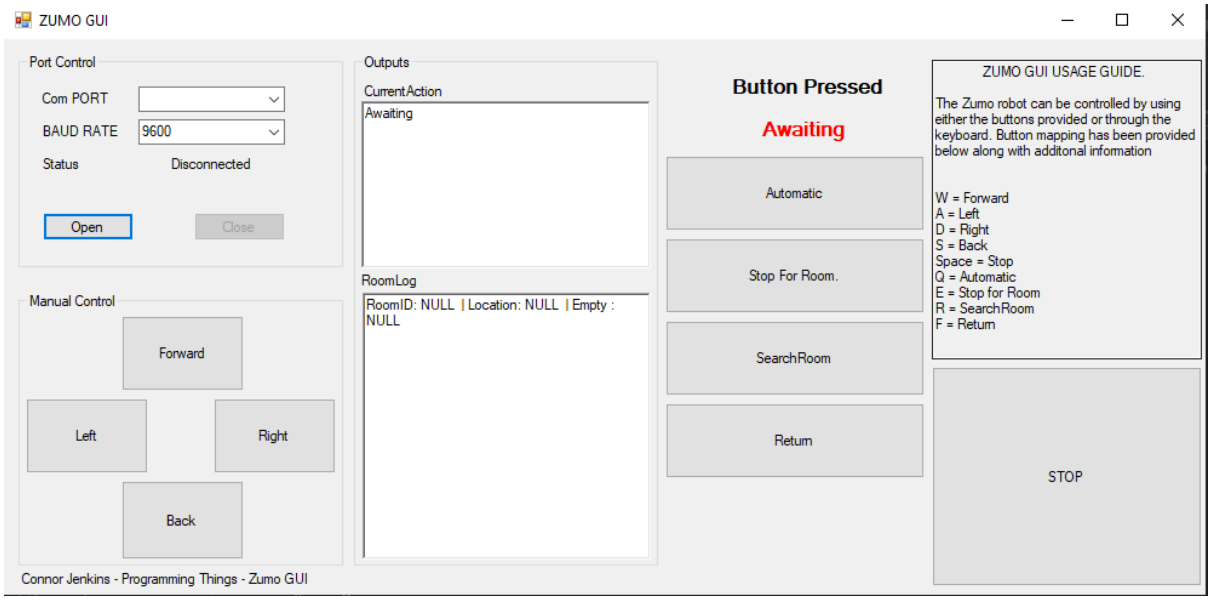

Figure 1ConnorJenkins-ZumoSearc&RescueGUI

The GUI is simple. From left to right the user is presented with a range of options.

- They're able to control the port, and the BAUD rate. As well as being able to open and close the connection. If no port is selected an error is shown. Present in figures 2

-They're able to control the zumo manually. Moving the Zumo forward, Backwards, Left and Right.

- Any ongoing or current actions are displayed within the CurrentActions box. Present in figure - 3

-Any rooms which have been searched, which be presented in the roomlog box.

-The user can enable the automatic process. This puts the zumo into an autonomous state.

-The user can stop the zumo upon reaching a room, they will then be presented with a selection. Presented in figure - 4

-The user can then tell the zumo to search the room if the wish. If they do not, they can turn the robot around manually, then begin the automatic process again.

-The user can press the return button to turn the robot 180 degrees, then begin the automatic process again.

-The user can stop the robot.

- The user can control the robot with keyboard controls. The keyboard inputs can be previewed in the ZUMO GUI USAGE GUIDE text box.

Below are figures for certain functions, Demonstrations for functions without provided figures can be found within the video.

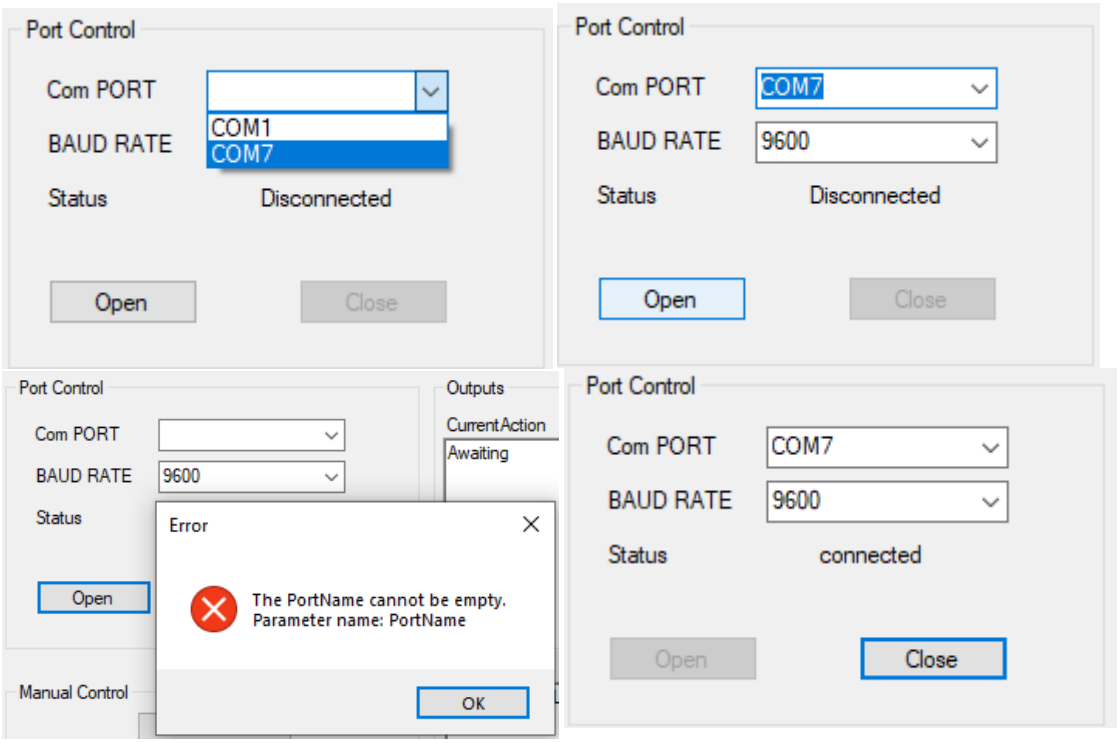

Figure 2 Port Control demonstration

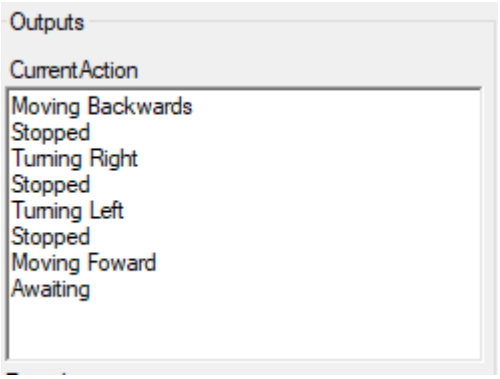

Figure 3 CurrentAction Demonstration

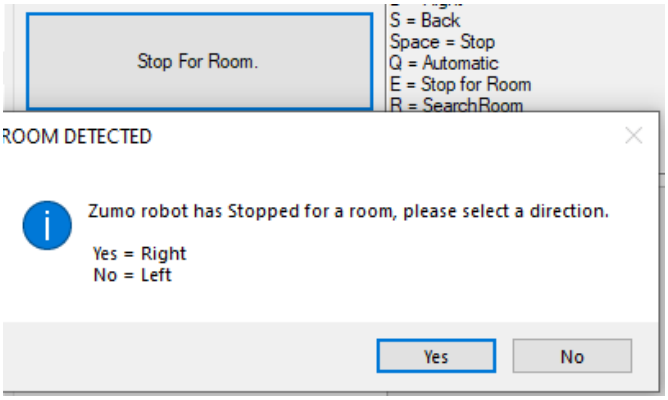

Figure 4 Stop for room demonstration

To use the ZUMO GUI follow the following Instructions.

-Ensure the Xbee module is plugged into the host device.

-Open the ZUMO GUI.exe.

-Select the Com PORT with the Xbee Module, the baud rate doesn't need changing.

If the Com port with the Xbee module cannot be found, re-plug the Xbee module, and restart the GUI

-Once the com Port has been selected, click Open. The Xbee module and your zumo should then been connected.

You are then able to use any of the functions available on the GUI.

Search & Rescue Code

This section overviews the Search & rescue code for the ZUMO robot. Below I have documented specific parts of my code in relations to tasks, and any resources I used in relations to those pieces of code.

Task 1:

Task 1 involved introducing manual control over the zumo robot. Within my code this is achieved with a switch, which changes depending on what is read by the serial port. Information is passed to the serialport by the GUI. Figure – 5 & 6

The zumorobot can be controlled either by pressing the button manually or through keyboard control.

```
protected override bool ProcessCmdKey(ref Message msg, Keys keyData) //Keyboard i
{
    switch(keyData)
    {
        case Keys.W:
            btnForward.PerformClick();
            return true;
        case Keys.A:
            btnLeft.PerformClick();
            return true;
        case Keys.D:
            btnRight.PerformClick();
            return true;
        case Keys.S:
            btnBack.PerformClick();
            return true;
        case Keys.Space:
            btnSTOP.PerformClick();
            return true;
        case Keys.Q:
            btnAutomatic.PerformClick();
            return true;
        case Keys.E:
            btnStopForRoom.PerformClick();
            return true;
        case Keys.R:
            btnSearchRoom.PerformClick();
            return true;
        case Keys.F:
            btnReturn.PerformClick();
                return true;
        default:
            return base.ProcessCmdKey(ref msg, keyData);
    }
}
```

Figure 6 – keyboard controls

```
void commands(int control) // manages the users inputs from the GUI.
{
  switch (control) {
    case 'f':  //Output code f represents forward
      forward();
      break;
    case 'b': //Output code b represents reversing
      reverse();
      break;
    case 'l': //Output code l represents left
      left();
      break;
    case 'r': //Output code r represents right
      right();
      break;
    case 's': //Output code s represents stop
      hault();
      break;
```

Figure 5 – Switch case handling serial inputs.

Task 2:

Task 2 involved introducing Autonomous control, the goal of this task was for the robot to autonomously stay within the black lines and stop at a wall. If a black line is detected to the side, it should turn away and continue. I achieved this by following several tutorials although, one which I found to be particularly helpful was the Zumo Bot Arduino 8: Line detection tutorial by codePetersen. (codePetersen, 2020). This helped me achieve the goal of staying within the blacklines, however the tutorial didn't assist with wall detection. This was achieved through trial and error of two different methods.

```
void detectwall() //Includes operations to assist with wall detection.
{
  if (linesensorvalues[0] > QTR_THRESHOLD) //upon hitting the QTR_Threshhold the leftdetectedc
  {
    leftdetectedcountdown = millis();
  }

  if (linesensorvalues[numberofsensors - 1] > QTR_THRESHOLD) //upon hitting the QTR_Threshhold
  {
    rightdetectedcountdown = millis();
  }

  if (rightdetectedcountdown > 15 && leftdetectedcountdown > 15) //
  {
    walldetected = true; //If both sensors reach a value greater than 15, a wall is detected.
  }

  //if a single sensor is only hit, after 50 ms it will be reset.

  if (leftdetectedcountdown > 50)
  {
    leftdetectedcountdown = 0;
  }

  if (rightdetectedcountdown > 50)
  {
    rightdetectedcountdown = 0;
  }
}
```

Figure 7 wall detection

```
} else if (
{
  // buzzer.playNote(NOTE_A(4), 2000, 10);
  reverse(); //Upon touching a left wall the robot will reverse.
  delay(300);
  automaticright(); //After reversing the robot will turn right.
  delay(300);
  automaticforward(); //It will then continue to move foward.
}
else if (linesensorvalues[numberofsensors - 1] > QTR_THRESHOLD) //right sensor & right wall
{
  //buzzer.playNote(NOTE_B(4), 1000, 10);
  reverse(); //Upon touching a right wall the robot will reverse.
  delay(300);
  automaticleft(); //After reversing the robot will turn right.
  delay(300);
  automaticforward(); //It will then continue to move foward.
}
else
{
  forward(); //continue forward.
}
```

Figure 8 LineDetection.

The first method involved using the front line sensor on the zumo robot to detect if a line was directly ahead. However, this proved to be very ineffective. Often, the left or right sensor would pick up the wall prior the centre sensor. This resulted in the robot turning in either direction.

Noticing this, I began working on my second method, which is the one I settled on using. This involved introducing the mills() function, and cojoining it to each of the sensors. If either the right or left sensor hits a black line, a counter begins. If it reaches a certain threshold on one sensor it is reset, however if both sensors hit the threshold, we assume there is a wall, and the robot stops.

This proved to be an effective method, however there are some issues noted within the complications section.

Task 3:

Upon achieving task 2, task 3 would be simple to solve. Upon hitting a wall, the robot should return control back to the user. This was achieved by further development of the wall detection function. Once the zumo had stopped, the zumo robot would altert the user that they have hit a wall with a popup box on the GUI. This popup box provides the user with three choices, Right, left or stop. Depending on their choice, the zumo robot would react appropriately. Once the turn is made the robot would stop again. If the user wished to continue they could press the automatic button or keyboard button. The popup is triggered by the walldetection function sending the output code 1. Figure – 9 - 10

```
if (walldetected) //if both the left and right sensor has been in contact with a wall within the last 15 ms there is a wall.
{
  motors.setLeftSpeed(0);
  motors.setRightSpeed(0);
  Serial1.print("1"); //Output code 1 represents a wall has been detected.
  delay(50);
  runautomatic = false; //if a wall has been detected the automatic process will be stopped.
```

Figure 10 – WallDetected Code

Task 4:

Task 4 proved to be quite challenging, as I was unsuccessful at introducing the gyro available on the zumo. More information provided within the complications section.

Task 4 involved having accurate 90 degree turns, upon reaching a wall and a user request. To achieve this, the turn right function was edited to turn zumo for a certain amount of time. As I was unable to use the gyro function, I simulated a 90 degree turn using delay(). The zumo robot would turn either left or right for 1570 ms. This would be a common occurrence within my code for later tasks, due the being unable to access the gyro.

To have the zumo continue automatically after the turn, the automatic() function would be called once the turn had been complete within the GUI code. Figure 9 & 11.

```
private void actionpopup() //Zumo intera
{
    DialogResult UA = MessageBox.Show("Z

        if(UA == DialogResult.Yes)
        {
            btnRight.PerformClick(); //s
            serialPort1.Write("7"); //re
            serialPort1.Write("a"); //co
    }

        if(UA == DialogResult.No)
        {
            btnLeft.PerformClick(); //si
            serialPort1.Write("7"); //re
            serialPort1.Write("a"); //co
    }

        if(UA == DialogResult.Cancel)
    {
        btnSTOP.PerformClick(); //simula
        serialPort1.Write("6"); // stops
        serialPort1.Write("7"); //resets
    }
}
```

Figure 9 – GUI PopUp

Task 5:

Task 5, also provided to be troubling with multiple bugs occurring, resulting in the rewriting of code or altering of previous methods. Most commonly issues with the proximity sensors occurred, where certain objects wouldn't be detected if present, or objects would be detected in none are present. (Craven, Zumo Proximity Sensor, 2016). A resource beneficial to my attempt of this task was a YouTube tutorial by Professor Craven (Craven, Zumo Proximity Sensor, 2016) . This assisted me in getting the proximity sensors to successfully work and detect nearby objects. The code for this is present in figure 12

```
void right() //manual control - zumo turns right then stops.
{
  Serial1.print("5");
  delay(50);
  motors.setLeftSpeed(100);
  motors.setRightSpeed(-100);
  delay(1570); //robot will turn for 1570 ms.
  hault();

}
```

Figure 11

```
void Detection() //Includes operations to assist with object detection.
{
  int right_sensor = 0;
  int left_sensor = 0;
  ObjectDetection.read(); //reads the proximity sensors.
  right_sensor = ObjectDetection.countsFrontWithRightLeds(); //assigning the Right Sensor to the right_sensor variable.
  left_sensor = ObjectDetection.countsFrontWithLeftLeds();   //assigning the left Sensor to the left_sensor variable.

  if (left_sensor >= 5 || right_sensor >= 5) //If the value read, is greater than 5, we assume an object as been detected. Else we assume no object has been detected.
  {
    Serial1.print("o"); //Output code o represents an object has been detected.
    objectfound = true; //if an object has been detected, object found is set as true.
  } else
  {
    Serial1.print("p"); //Output code p represents no object has been detected.
  }
```

Figure 12

Task 6 & 7:

I was unable to attempt or successfully achieve task 6 or 7.

Achievements

Task 1 – Zumo robot can successfully be controlled with keyboard, or by using the provided buttons.

Task 2- Zumo robot can successfully navigate a hallway and stops at a wall.

Task 3- Zumo robot successfully stops at a wall, and requests a user input to either turn left, or right.

Task 4- Zumo robot successfully stops at a wall, and after a user had made their turn request continues its journey.

Task 5- Partially successful however bugs are present. The zumo robot will stop outside a room when requested, presenting the user with the option to turn left or right. The user must manually tell the robot to search the room rather than automatically. The zumo will successfully search a room given the order. Once a search has been complete the zumo robot will return to the hallway and continue its journey. The zumo is able to generate a roomid and store whether or not anobject was found. The zumo robot is unable to store the location of the object. Both the roomid and whether or not a object was found is present on the UI.

Task 6 – unachieved.

Task 7 – unachieved.

Complications

During the development of task 4, to introduce accurate 90 degree turning I attempted to introduce an example header file and cpp provided among the zumo example packs, in conjunction with following a tutorial on accurate turning. (Kevin-pololu, 2021) (Craven, Turn/Gyro Sensor Example for Pololu Zumo 32U4 Example, 2016) (pvcraven, 2016) . My attempt to do so were ultimately unsuccessful as I was unable to import the library. Receiving an error when attempting to use the L3G library. Alternatively I returned to using delay() producing timed turns. While inaccurate, I was successful at getting the robot to turn to the required position.

Bug List:

- If a level surface isn't provided it is possible for the a wall to not be detected, causing it to act as if it were a corridor.

-Wall Detection may accidently trigger if there is poor lighting.

-Turning inaccuracies, as I was unable to implement gyro for accurate turning and resulted to using delay(), turns my be overshot or too little depending on the surface.

Bibliography

codePetersen. (2020, April 15). *Zumo Bot Arduino 8: Line Detection*. Retrieved from Youtube: https://www.youtube.com/watch?v=PvFr9bUfNG8&t=338s

Craven, P. (2016, aug 15). *Turn/Gyro Sensor Example for Pololu Zumo 32U4 Example*. Retrieved from Youtube: https://www.youtube.com/watch?v=XOp22Xx7ZnU&t=1s

Craven, P. (2016, Aug 15). *Zumo Proximity Sensor*. Retrieved from Youtube: https://www.youtube.com/watch?v=ddPo6HQvxzQ

Kevin-pololu. (2021, nov 16). *RotationResist*. Retrieved from Github: https://github.com/pololu/zumo-32u4-arduino-library/tree/master/examples/RotationResist

pvcraven. (2016, aug 15). *GyroSensorExample*. Retrieved from Github: https://github.com/pvcraven/zumo_32u4_examples/tree/master/GyroSensorExample