Programming Things Coursework 1

report

Liam Sykes

2019

Contents

No table of contents entries found.

# What I Have Achieved

In this section I will discuss what I successfully achieved.

## Task 1: Manual Control

This task required me to have wireless manual control of the Zumo.

### Wireless Communication

The first task which was to control the robot manually via wireless communication over the xBees has been successfully completed. This has been achieved by using the G4P\_guistart example provided on Blackboard. This example uses the G4P\_Controls library in order to create the GUI in Processing. It also uses the Processing.Serial library. For the manual controls to be sent via the xBees in the Processing guiStart file I declared an instance of the Serial class called serialPort and then I create it in the setup function. When I create the serialPort I pass in this for the parent, the xBeePort which a string set to COM4 which is the COM port where I connect an xBee to my laptop and I pass in 9600 which is the baud rate I want to use. In the Arduino code it reads the value of the Serial into a string variable called incomingBytes and there is an if statement that is checking to see if the value of incomingBytes equates to a command for the Zumo.

### Libraries and Examples Used to Complete This Task

In order to complete this task, I used the ZumoShield library so that I had access to the ZumoShield class and could set the speeds of the motors. I learnt how to set the speeds of the motors in many of the tutorials on the Zumo which is to use the setSpeeds function.

### Forward Movement

For the forward manual movement, I have a button which has the text ‘forward’ on the GUI which when clicked writes the letter ‘w’ to the serial and is accepted in the Ardunio code where it calls a function named forward which is declared in the Movement class in the movement.h. When this function is called it sets both the motors in the Zumo to a speed which is passed in the forward direction for manual control the speed is set at 150.

### Right Movement

For the right manual movement, I have a button which has the text ‘right on the GUI which when clicked writes the letter ‘d’ to the serial and is accepted in the Ardunio code where it calls a function named right which is declared in the Movement class in the movement.h. When this function is called it sets the left motor in the Zumo to a speed which is passed in the forward direction and the right motor to the speed passed in but in reverse for manual control the speed is set at 125.

### Left Movement

For the left manual movement, I have a button which has the text ‘left’ on the GUI which when clicked writes the letter ‘a’ to the serial and is accepted in the Ardunio code where it calls a function named left which is declared in the Movement class in the movement.h. When this function is called it sets the right motor in the Zumo to a speed which is passed in the forward direction and the left motor to the speed passed in but in reverse for manual control the speed is set at 125.

### Backward Movement

For the backward manual movement, I have a button which has the text ‘back’ on the GUI which when clicked writes the letter ‘s’ to the serial and is accepted in the Ardunio code where it calls a function named backward which is declared in the Movement class in the movement.h. When this function is called it sets both the motors in the Zumo to a speed which is passed in the reverse direction for manual control the speed is set at 150.

### Stop Movement

In order to stop the Zumo in manual movement I have added a button with the text ‘stop’ to the GUI and when this is clicked it writes a space to the serial which is accepted in the Arduino code where it calls a function called halt in the Movement class in the movement.h. When this function is called it sets both the motors in the Zumo to a speed of 0 to bring the Zumo to a stop.

### Issues in this task

One of the major issues with this task was although the GUI was writing messages to the Serial they were not always read from the Serial in the Arduino code. Having done some research realised that the xBee may be holding on to the Serial and therefore I need to add the if Serial.available() > 0 so the Arduino knew when there was something to be read. After that fix everything worked exactly as expected.

## Task 2: Autonomous control of the Zumo

This task required the Zumo to move autonomously along the map until a command is input to make it do another task.

### Libraries and Examples Used to Complete This Task

I used the BorderDetect example in order to help me with this task however it was adapted so that when a wall is hit head on the Zumo comes to a stop. I used the ZumoShield library again for this task as I needed access to the ZumoShieldReflectanceArray class.

### Autonomous Control

Firstly, I created an instance of the ZumoShieldReflectanceArray class called sesnors and I pass in an argument of QTR\_NO\_EMITTER\_PIN so that the reflectance array does not use pin 2 as that is required by ultrasonic sensor. I define constant called NUM\_SENSORS which has a value of 6 as this is how many sensors the reflectance array has. I then declare an array of type unsigned int, as I don’t want the reflectance array to read negative values, called sensor\_values which is set to the size of the NUM\_SENSORS constant. This array will be used to decide if any of the sensor values mean that the robot is detecting the white card or the black tape. Then in my autonomousMovement function the values of the sensor are read into the sensor\_values array.

Next if the left most and right most sensors are reading values above the QTR\_THRESHOLD, which is a constant defined and set to 800, or the two right most sensors are reading values above the QTR\_THRESHOLD or the two left most sensors are reading values above the QTR\_THRESHOLD then the halt method is called in the Movement class. The autonomous bool also gets set t false, this bool is used to determine whether the Zumo is autonomous being control or whether it is being control manually. Else there is a delay for 50 milliseconds and if the left most sensor is reading a value above the QTR\_THRESHOLD then the halt function is called to stop the Zumo there is a 500 milliseconds delay then the adjust function is called with a speed of 150 for the left motor and a speed of 0 for the right, this is a function in the Movement class where two different speeds can be passed in. This will cause the Zumo to move slightly right to move it back on to the white card. After that there is a delay of 250 milliseconds. Else if the right most sensor is reading a value above the QTR\_THRESHOLD then the halt function is called to stop the Zumo there is a 500 milliseconds delay then the adjust function is called with a speed of 150 for the right motor and a speed of 0 for the left. This will cause the Zumo to move slightly left to move it back on to the white card. After that there is a delay of 250 milliseconds.

### Issues in this task

There was an issue when trying to make the autonomousMovement function where the Zumo would just ignore the black tape and drive over it completely. I realised that the reason for this was that I had set the speed of the Zumo to 75 and this meant that the delays I had put in were two small so I increased the speed to 100 which resolved the issue. There was also an issue where when the Zumo hit a wall head on it would just adjust rather than stop and this was because sometimes both the right most and left most sensors would not be on the tape. To resolve this issue, I added the checks for the two left most and two right most sensors. Another issue when developing this section of the code was that on every test run the reflectance array would read different values and I could not set the QTR\_THRESHOLD to something suitable. When I asked my tutor about this, he said it was because I was not calibrating the sensor so to resolve this, I used the line follower example for help and I now calibrate the sensor in the setup method of the Arduino code.

## Task 3: Turning Corners

This task required me to make the Zumo be able to handle corners manually and then resume autonomous control.

### Libraries and Examples Used to Complete This Task

No additional libraries were used for this example and no examples were used to assist with the development of this task.

### Turning Corners

When the Zumo is in autonomous and it reaches the wall at a corner and comes to a stop using the halt function. When this happens I added the code for a message to be sent from the Zumo stating it has hit a wall or a corner and it requires a command and to deactivate the autonomous functionality so that the user could manually turning the corner without the autonomous functionality interfering. The user manually controls the Zumo around the corner using the control buttons on the GUI. Once they have successfully turned the robot around the corner. They send the ‘c’ character using the Send Messages to The Zumo text field on the GUI. In the code it gets the text entered into the field using the getText function and sets that into a string variable called text. The text variable then gets passed into the serialPort.Write method which sends it over the xBees to the Arduino then the textfield gets set to empty string ready for the next message to be input. When the Ardunio accepts the ‘c’ character it sends a message from the Zumo using Serial into the Message From Zumo text area on the GUI stating that it continuing autonomously and sets the autonomous bool to true which reenables the autonomous functionality of task 2.

### Issues in this task

There were no issues in this task.

## Task 4: The Zumo searches a room.

This task required me to make the Zumo turn into a room and use a ultrasonic sensor to detect objects in that room.

### Libraries and Examples Used to Complete This Task

To complete this task I used the NewPing library so that I had access NewPing class so that I could read values from the ultrasonic sensor attached to my Zumo. In order to complete this task, I adapted the NewPingExample file so that I could read the distance of an object using the ultrasonic sensor. I used the LineFollower example to get the sweeping motion correct inside of a room when tying to detect and object.

### Searching a Room

Firstly I define three constants, the first being TRIGGER\_PIN which is the pin that the ultrasonic sensor trigger is connected to on the board which is 6, the second being ECHO\_PIN which is the pin that the ultrasonic sensor echo is connected to on the board which is 2 and the last being MAX\_DISTANCE which is the maximum distance I want to the sensor to detect which I have set to 30. Next, I create an instance of the NewPing class called sonar and pass the three constants into the constructor. I then allowed the messages ‘Ro L’ and ‘Ro R’ to be sent from the GUI and accepted in Arduino.

When ‘Ro L’ is sent the Arduino code turns the Zumo left at a speed of 150 and then delays for 800 milliseconds so that it turns into the room before calling the roomSearch method passing through the room location of left so that the message written in the function relate to a room on the left. When ‘Ro R’ is sent the Arduino code turns the Zumo right at a speed of 150 and then delays for 800 milliseconds so that it turns into the room before calling the roomSearch method passing through the room location of right so that the message written in the function relate to a room on the right. Inside the roomSearch function. A bool called object gets declared and set to false as no object has been detected yet. An int called roomNumber that is in the global scope gets incremented for every room entered so that a count of the rooms can be stored and used in some of the messages. The int then gets converted to a string so that it can be used in the following message: Here is room number <RoomNumber> and is located on the <left or right>. This message gets output to the GUI so the user knows the information. This information also gets stored using a method called storeRoomLocations I pass through the room number and the location. In this method I store the room number and location into an array called roomsAndLocations.

Once the Zumo has established the location and the room number and this information has been stored it moves further into the room at a speed of 75 and delays for 500 milliseconds. After this is scans the room using the code from the LineFollower where it rotates left and right on the stop while using the ultra sonic sensor to detect an object where if the ultrasonic has a reading in centimetres greater than 0 it sets the object bool to true. Once it has finishing scanning the room, it comes to stop using the halt function in the Movement class. Then if an object was detected it writes a message to the GUI stating Object detected in room <RoomNumber> and it stores this using a method called storeObjectDetected where I pass through the room number. In this method I store that an object was detected and what room was searched. If no object is found, then only a message gets displayed on the GUI saying: nothing detected.

The user then manually controls the Zumo out of the room and presses ‘c’ so that it continues autonomously.

### Issues in this task

The first issue I encountered in this task was that when I was sending the message either ‘Ro L’ or ‘Ro R’ to the Arduino it was only getting part of the message. I soon realised this was because I was using the Serial.read function which only reads the first character which was fine for the single character messages for the previous tasks but now I needed it to accept strings. To resolve this issue, I changed the read function to Seria.readString which allowed me to send the ‘Ro L’ and ‘Ro R’ messages. Another issue in this task was that when I first came to detect the objects the ultrasonic sensor was not returning any values, and this was because the reflectance array and the ultrasonic sensor where conflicting with each other. To resolve this I added a sensor.init function to my setup function in the Arduino code where I pass through QTR\_NO\_EMITTER\_PIN which tells it not to use my pin 2 which the ultrasonic sensor was using and I added this to my calibrate method too.

## Task 5: The T-junction

This task required me to allow the Zumo to handle T junctions and search both sides but not search rooms it had already searched and not turn down the main corridor.

### Libraries and Examples Used to Complete This Task

No additional libraries were used for this example and no examples were used to assist with the development of this task.

### Handling the T Junction

The Zumo hits the wall at the top of the T junction and the same functionality as task 3 comes into the play it tells the user it has hit a wall, waits for a command and turns off the autonomous functionality. The user then manually turns the Zumo left or right and sends the character ‘c’ so that autonomous functionality turns back on. When the Zumo passes any rooms then the user enters the correct message so that the room search described in task 4 is completed. Once the Zumo reaches an end of the T junction it hits the wall like when hitting the top of the T junction and it asks for a command the user should press the ‘e’ character. When this is done a message is sent to the GUI from the Zumo stating: End of T junction reached. Then an int variable named ePressed gets incremented so the Zumo knows how many times an end of the T junction has been reached. It turns the autonomous functionality off so that the user can manually control the Zumo so that it is pointing the forward back down the corridor. At this point if the bools passedRoom and mainCorridor are set to false which they are initialised as then the ignoreCommands bool gets set to true. This is so that when we return down this half of the corridor the user can input the messages for entering the room and turning down the main corridor can be input so that the Zumo knows it has passed them but it does not do the actions of those inputs. When those commands are input it sets the respective bools to true so that once the Zumo has passed the main corridor the ignoreCommands bool can be set to false and rooms can be searched again. Once the second end of the T junction is reached the e character should be pressed again and then all the elements of the two arrays, roomsAndLocations and roomsWithObjects, are sent to the GUI.

### Issues in this task

When developing this task I thought I had completed the task yet upon testing when I passed the main corridor the Zumo still did not accept commands and I realised I needed to set ignoreCommands to false when I set the mainCorridor bool to true.

## Use of Mulitple Files

I decided to create the movement.h with the Movement class inside of it because I realised that I was using setSpeeds function a lot in code but changing the speeds to assist me with the tasks required. Therefore, it made sense to create methods for the different basic movements: forward, backward, left and right but to pass in the different speeds when required. The first thing in the movement.h I include the ZumoShield header file and create an instance of the ZumoShield class and call it motors so that I have access to the motors of the ZumoShield. I then use this to set the speeds as discussed previously to complete the basic movements. I pass in the required speeds for different tasks into these functions as for every task the speed is not the same. In here I use static functions so that I do not have to create an instance of the Movement class in my Arduino code and because of this the variables used in the class must be static.

# Video URL

As the video, created as evidence of the Zumo performing the tasks, has a file size too large to be placed in my GitHub repository I have uploaded it to Google Drive. Please find the link below:

<https://drive.google.com/file/d/1R3mYMUZBJ9Rdk_RTZQieYZrZ_jWTkq2z/view?usp=sharing>