# Programming Things

## Assignment 1

Jack Woodhouse
Software Engineering
B5027334

## Introduction

In this document I will assess what I have achieved within the first assignment of the programming ‘things’ module. I will also describe what I failed to achieve and why I think I failed to complete the tasks set to me. I was able to complete 5 of the 6 tasks set in the assignment brief and I am pleased with the final version of my program.

## Task 1: Manual Control

The first task in the zumo assignment required me to manually control the robot from a GUI that I created in Processing. I used Processing to create my GUI as it was an environment I had used before I started the assignment and Java is a language I am fairly comfortable using so i was confident that I could create something suited to the purpose of allowing easy and quick control of the zumo as it navigated its way around the maze.

An example was provided to me that used the G4P_Controlls library created by Peter Lager. It provided a strong base that I was able to expand and use for the rest of the assignment. The GUI was written in Java and made use of simple buttons, text fields and text boxes to allow the user to write commands and receive messages from the zumo in real time. The commands sent to the zumo, that allowed the user to move, stop and turn the zumo, were handled by two XBee’s.

To help me achieve the task of controlling the zumo manually I needed the help of some external code and libraries that I got from the Arduino forums and Processing. The code that I used from this source allowed me to read, write and access information within the XBee from my usb port within Processing. (1) (2).

With the help of the provided GUI example and the code I used from the Arduino Forums I was able to move the zumo, read and write messages to it in real time which would later allow me to tell the user when it has found an object in the room it is scanning, that it has found a dead end or that it is simply turning a corner. I had very little issue completing the first basic task and it allowed me to form a strong basis to complete the rest of the assignment.

Task 2: Autonomous control of the Zumo

The second task of the assignment was to allow the zumo to move autonomously within the corridors of the maze, stay within the borders and stop at any corners that it found. This task proved to be the first that gave me some real issues to overcome and account for when programing these individual components.

To complete this task I had to use a Zumo Reflectance Array. Within the array were 6 small LED sensors that I could use to detect the difference in light from a surface underneath it, it that was attached to the front of the zumo just in front of the wheels. The code which allowed me to detect the edges of the maze made use of the outermost sensors and making the zumo stop at a corner use a combination of all the sensors. I this to ensure that I got consistent results and will help the zumo to cope better with different surfaces and lighting conditions.

To help with this task I used some examples from a zumo library provided within the arduino software (3). These were LineFollower and BorderDetect, I used these examples as they gave me a good idea of how to make the zumo stay within a set border and helped me to understand how to set the Zumo Reflectance Array to detect black lines on a white base. This would go on to become the base of this task and is what allowed me to complete it to a high standard

During this task I started to face some major tasks with hardware and within the code itself. The first key issue that I faced was that I was unable to reliably keep the zumo within the corridors of the maze as on some runs it worked flawlessly and on some other runs it would fail to detect the lines and run over them. To overcome this issue i changed the sensor threshold to a lower value which was more appropriate to the maze I had created and I slowed the speed of the zumo as it navigated the track to ensure it had enough time to read the sensor values from the reflectance array and compare it to the values I had outlined in the program.

A second issue I found was getting the border detect and the end stop to work together in a reliable manner. On the tests I had setup the zumo would keep itself within the border as expected using the outermost sensors but would not stop when it came to a corner or dead end as I would like. To overcome this issue I first thought it would be better the use the out sensors for the edges and use the middle sensors for the dead ends. Doing this helped me progress but didn't not fully solve my problem. I would not be able to solve this issue and complete the task until it was suggested to me that I try using all the sensors together with more frequent readings when trying to stop the zum rather than using only the outer sensors or only the middle sensors. I used a statement that checked all the sensors together and this is allowed me to get reliable results inline with the task.

Task 3: Turning Corners

The third task set was to allow the zumo to detect a wall turn left or right and then continue the maze. This task was fairly simple and I was able to complete the task quick and easily with minimal issues.

Completing this task would require a combination of Task 1 and Task 2 to allow the zumo to stop when required, let the user input a command such as turn right or left and then to tell the zumo that it can continue. I started by creating a bool in the main loop of the program to help me turn on and off the main border detection function that I had created in task 2.

When the zumo it a wall it would stop and reverse slightly and then output a message to Processing gui “you hit a wall!” and then to ask the user to input a command. The user would manually turn the zumo left or right and then enter ‘c’ to then reactivate the border detect function and allow the zumo to make its way through the rest of the maze.

I did not need to use any external libraries or examples for this task as I found it fairly simple however I did try to a Compass example to help make the left and right turns as accurate as possible but i decided that it better to allow the user to input their own left and right turns and stop with the use of the GUI that I had created earlier. I found that the Compass was influenced by its surroundings depending on where the map was placed and I think that allowing the user manual control of the turn was more suitable for the program.

Task 4: The Zumo searches a room.

The fourth task set for this assignment was to allow the zumo to search inside a room for an object, identify if an object has been found or has not been found and store that information to display to the user.

Searching a room within the maze is key part of the assignment and required me to make use of an ultrasonic sensor that was attached to the front of the zumo. This sensor allowed my to detect objects in front of the zumo using sound waves emitted from the sensor.

First the robot was stopped outside the room by the user with the GUI I created earlier. They could then enter a ‘L’ or ‘R’ depending on which side of the map the room was located on. This location is then stored in an array for use later. The zumo then rotated on the spot to scan the room and search for an object and the result of this scan is then also stored in an array to be used later.

Within this task I encountered a few key issues that slow down the overall progress of the assignment. The first of these issue was getting the zumo to reliably scan the rooming use the ultrasonic sensor that was provided to me for use in this project. I found that the readings from the sensor were sometimes inconsistent which in turn gave me false readings when looking for objects inside rooms. To solve this issue I first safely secured the sensor to the front of the zumo to ensure it was looking in correct direction at all times and then i made use of the serial monitor in the arduino software to look at the readings the sensor was giving me to help identify a faulty sensor.

A second issue I found during this task was getting the zumo to rotate when inside the room so that it could scan from side to side and make sure no areas of the room were missed and therefore objects placed inside. To overcome this issue I used a basic sweep loop that I found inside the LineFollower example. It allowed me to rotate from

left to right and back again while also scanning the room and storing an objects placed inside.

Task 5: The T-junction

The last and final task that I completed in this assignment proved to be another difficult task with some of the smaller details giving some trouble. This task required me to take everything I had learnt from the previous 4 tasks and put them all together and work a cohesive unit.

The basis of the task was that the zumo was to reach an end of a corridor and then turn left or right, proceed to the end searching an rooms and then check the other end of the corridor in turn. The user could control the robot with the basic commands that they had been using throughout the other tasks however i was required to make the robot ignore commands on the second half of the corridor which came to be an issue for me when trying to complete the task.

The first main issue that I encountered and had to overcome was forcing the robot the ignore commands on the second half of the corridor but still acknowledge that they were entered by the user in the GUI. I decided that it was best to create a bool that allowed me to tell the zumo when it had passed both a room and corridor and therefore activate or deactivate controls sent by the user.

Sources


1. http://forum.arduino.cc/index.php?topic=164087.0
2. https://processing.org/reference/libraries/serial/Serial.html
3. https://github.com/pololu/zumo-shield/blob/master/ZumoExamples/examples/BorderDetect/BorderDetect.ino


Github Repository

The github link below contains a link to the repository containing the code files for the arduino and GUI. It also contains a video of the completed assignment.

https://github.com/jackwoodhouse/Programming-Things