Zumo Search and Rescue Reflection Report

# Module: Programming things

## Student: Benjamin Sinyard B7020156

# Introduction

The task for this assignment was to develop a search and rescue robot with the Arduino board Zumo 32U4. This had to be done wirelessly via an Xbee module using the serial ports, which meant that it could follow the commands wirelessly. A GUI was added to aid the use of this robot.

# The tasks.

There was a total of 7 different tasks to be completed. I managed to complete a total 6 out of the 7 available.

The tasks were as follows.

1. Task 1 – Manually control the Zumo via W,A,S,D wirelessly with a basic GUI.
2. Task 2 – Autonomous control of the Zumo. It follows the corridor by keeping within the lines, and the zumo stops when it reaches the far wall.
3. Task 3 – Turning Corners, the Zumo reaches a corner and stops. It returns manual control to the user, who will then navigate around the corner.
4. Task 4 – The Zumo turns autonomously around a corner. The zumo recognises the corner and then will initiate a 90 degree turn after prompting the user to either go “L” (left) or “R” (right).
5. Task 5 – The Zumo searches a room. The human navigator will stop the Zumo outside the room and then signal to the Zumo which way the room is, and then the Zumo will automatically go into the room and search before alerting the user if somebody is found.
6. Task 6 – The T-Junction. The Zumo will allow the user to turn Right or Left down the corridor at the junction, and once it reaches the end will prompt the user to press ‘b’. When they do the Zumo turns 180 degrees and continues to the other end of the corridor.
7. Task 7 – The return journey. The Zumo uses the room data to find the quickest route back, while going to each room and making a sound to let rescuers know where each person is located.

# The fulfilled tasks.

Out of the 7 total tasks, I developed working code for 6 of them these are:

### Task 1 – Manual Control

I programmed the Zumo to work in a manual mode. This mode allowed for complete control of the Zumo robot via the keyboard below is a table illustrating the different key presses and their uses:

|  |  |
| --- | --- |
| Key | Action |
| ‘w’ ‘a’ ‘s’ ‘d’ | These presses moved the Zumo around, in the following directions: Forward, Left, Backwards, Right. |
| r | Informs the Zumo you want to log a room, first you press ‘r’ and then you signal whether the room is on the Right (‘r’) or Left (‘l’) |
| L | Performs the room search using the last recorded room direction. The Zumo moves 90 degrees in the direction of the room, before moving forward and sweeping left or right with the IR sensors. If a person is found an audible “BEEP” is given off and then the Zumo returns the corridor by reversing and turning 90 degrees in the opposite way from the start. |
| ‘z’ | Emergency stop, which will stop the Zumo from moving, this also works in auto mode. |
| ‘q’ , ‘e’ | Turns the Zumo 90 degrees to the LEFT (q) or RIGHT (e) using the motors encoders to ensure that the turn is accurate to near 90 degrees. |
| ‘h’ | Makes the Zumo play a beeping noise as a horn. |
| ‘1’ | Switches the mode to the Autonomous mode |
| ‘c’ | At the start of the program the user calibrates the Line Sensors with c. |

### Task 2 – Autonomous control

In this task I created a piece of code that after calibrating the Zumo’s line sensors will allow the robot to navigate between two black lines. I looked at the Maze Solver example and Line Follower examples from the Zumo32u4 libraries.

The way the Zumo navigates away from the lines is that it scans the floor it’s moving along using 3-line sensors (1 left, 1 middle and 1 right). When one of these sensors goes over the calibrated value of the white background it will initiate a turn away from the wall. Once the correction is done the Zumo will continue to move forward.

Once it reaches the end of the corridor it will check for a repeated hit on the most right/left sensor. If it hits the outer sensors more than twice in a row it will stop the Zumo and correct it’s turning and reverse to move away from the wall.

This will also occur if the middle sensor goes over the black line.

## Task 3 – Zumo Turns Corners 90 degrees

Once the Zumo hit the wall using the line following algorithm and stops, the user can then turn around the corner and turn the Autonomous mode back on using the ‘1’ key.

Prompting messages are sent over the Xbee to the GUI so that the user can follow the instructions given.

## Task 4 – Zumo turns Autonomously 90 degrees at the corner.

I have implemented an algorithm using the Zumo motor encoders. The encoders mean that the motors will log how much the motors have moved over time, and the position the motors are in. I was able to figure out that an encoder number of around 650 mean that the Zumo had turned approximately 90 degrees on the right or left depending on the desired direction.

I used a boolean that allows gives the function knowledge to which direction the motors should turn. By using this algorithm, it meant that at the end of the corridor the Zumo can turn 90 degrees.

When the Zumo reaches the end of the corridor it will send a message over the Xbee wireless to the GUI. The GUI will inform the user that it is waiting for a ‘l’ or ‘r’ key press to turn either left or right. Once this is done, it will perform the 90 degrees turn and continue along the path.

## Task 5 – The Zumo Searches a Room

While in manual mode the Zumo can reach a wall and then the user press the ‘r’ button to signify it has reached a room. The GUI will prompt them to enter either ‘l’ or ‘r’ to indicate the direction of the room. Once this is done, the user can press the ‘l’ key to begin the search. The search algorithm will look at the last logged room and use the turn90Degrees algorithm to turn towards the room.

Next the Zumo will navigate itself into the room before turning left and right repeatedly while firing IR pulses from the front facing infra-red sensors. If a person is inside the room, the Zumo will beep and then send a message to the GUI informing the user if:

1. A person is found/not found
2. The location of the room
3. The room number

Once this is complete the Zumo will reverse outside of the room, before initiating a 90 degree turn back to where it was facing in the corridor.

In autonomous mode the process is very much the same aside from when the user signifies the robot to stop with ‘r’, and after the direction in which to turn the Zumo, it will begin the search without waiting for another input by the user.

## Task 6 – The T-junction

In autonomous mode the Zumo will top at the end of the corridor (the T-junction) and prompt the user via the Xbee to the GUI whether they want to go left, or right first. Once prompted the Zumo will turn 90 degrees using the turning algorithm and go down the hallway and can be stopped to check rooms. The Zumo will also increment the stop counter by 1 to inform the program that it has reached the first stop.

Once the Zumo reaches another end of the corridor, the Zumo will know that it now has to turn 180 degrees around and search the opposite end of the corridor. It will also disable the search option, the GUI informs the user in the instructions that to search the other end of the corridor, they must signal to the Zumo that they are past the already searched area by pressing ‘p’. Once this is done the Zumo can search the rest of the corridor’s rooms. The stop counter is incremented by 1 again.

Once it reaches the end of the corridor, the Autonomous mode will stop and hand back to manual before informing the User that they have reached the end of their journey.

# **Issues while developing**

## Infra-red sensors finding targets

During development the Infra-red sensors that are used to detect people in the rooms were unpredictable. After looking into the header files, and examples I found that the Line Sensors can cause interference with the Zumo’s IR sensors. To get around this I added some tape to the IR LED’s to lessen the spread of the light, I also ensured that the Line Sensors IR sensors were turned off during the scanning. I found that often the values that the IR sensors would always be between 3-6, so I chose that 5 would signify that it is close enough to an object. I didn’t want the Robot to detect objects that were outside the boundaries of the room, so therefore I had to get the Zumo quite close to the object to detect them, this led to times when the Zumo would not find a person and thus required extra tweaking.

## Detecting the end of the corridor

The way that I developed my line avoidance algorithm was to make the Zumo turn in the opposite direction when the furthest left, or right sensors detected the line. This made detecting the end of the corridor difficult, as when it hit the end it would just try to move away from the line stead of stop as the middle sensor didn’t always make it over the black line.

To get around this I had to add a “hit” counter, in which if a sensor was still hitting a black line repeatedly after changing direction, it must mean that there is a line directly in front of it. So, I decided that 2 strikes were enough to detect this end, and then I made the Zumo turn back to face straight forward and reverse.

## Turning 90 degrees accurately.

At first, I tried to use the onboard magnetic sensor to detect the orientation of the Zumo, this worked by calibrating the Zumo’s magnetic sensor by spinning the Zumo 360 degrees. This allowed for accurate turning, most of the time. However, sometimes the motors themselves would cause inaccurate readings from the sensors and cause the Zumo to keep turning left and right trying to fight the correct position.

To get around this I decided to use the Motor encoders, as there was a Zumo32U4 library that demonstrated reading them. I was able to run a couple tests to find that when a motor has turned 650 units, it would have turned around 90 degrees. This is good because the no matter the terrain it would always turn the correct amount.

## Reading messages from the Xbee via pygame

I had to figure out a way to both get the most up to date messages from Zumo while simultaneously allowing the GUI to work with w,a,s,d to keep moving the Zumo around. I found that I could create a separate thread, which meant any updates from the Zumo could continuously update.

# Sources

The application is developed from the Pololu libraries, which include:

1. Zumo32u4Motors – Allows for simple motor controls
2. Zumo32u4Buzzer – Plays a range of notes from the Zumo buzzer
3. Zumo32u4ReflectanceSensorArray – Interaction with the Line Sensors
4. Zumo32u4Encoders – Specific movement of motors to fine tune turns
5. Zumo32u4ProximitySensors – Detection of objects in front of Zumo

The Pololu libraries also included examples, such as MazeSolver and LineFollower that helped figure out how to calibrate the Line sensors and avoid lines. Examples were also given for reading/resetting the encoders