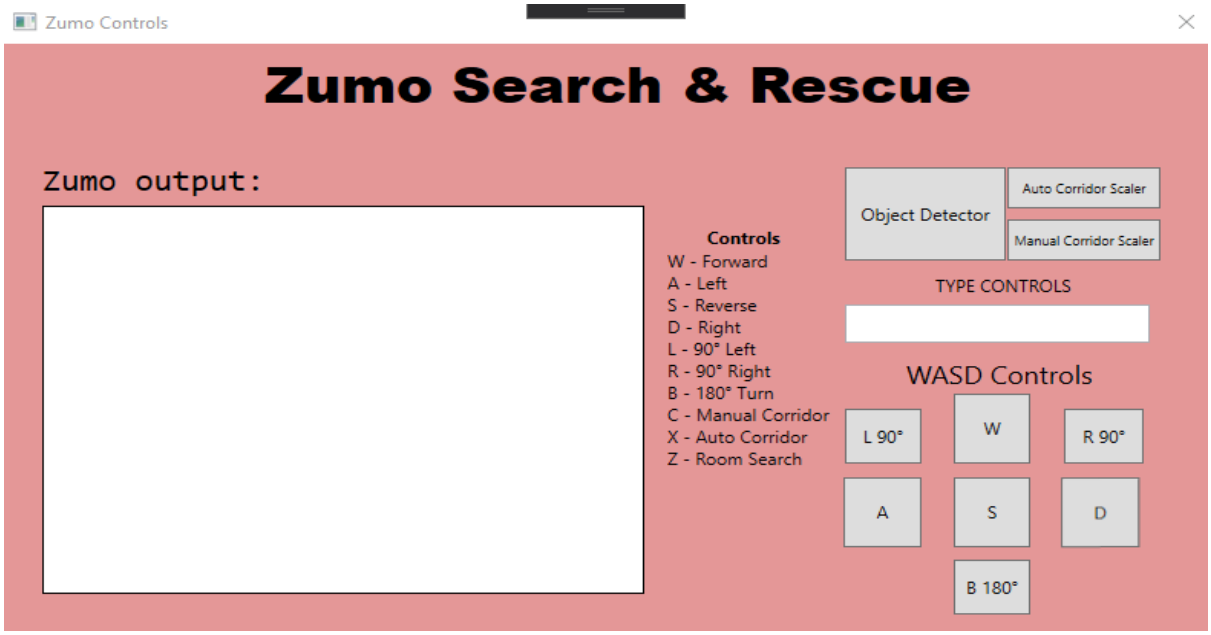
Zumo Robot Search and Rescue – Report

Tasks Achieved

From the tasks outlined in the assignment brief, my code and robot achieved the following;

- Task 1: Manual control of the Zumo.
- Task 2: Autonomous control of the Zumo
- Task 3: Turning Corners
- Task 4: The Zumo turns autonomously around a corner
- Task 5: The Zumo searches a room.

GUI Overview - C# & WPF



- The GUI, written in C# and WPF, allows for control over the robot. Each section of the GUI outputs to the serial port "COM5" where the XBEE device is connected to.

GUI Component	Behaviour
Zumo Output	Any messages sent from the robot are outputted here (for example, directions of travel, people found in a room etc.
Object Detector	This button executes the object detection function.

GUI Component	Behaviour
Auto Corridor Scaler	This button executes the automatic corridor scaler function.
Manual Corridor Scaler	This button toggles the manual corridor scaler function.
Type Controls	This is a text box where the user can input commands for the Zumo robot. The type controls consist of; W - Forward, A - Left, S - Reverse, D - Right, L - 90° Left, R - 90° Right, B - 180° Turn, C - Manual Corridor, X - Auto Corridor, Z - Room Search
WASD Controls	These buttons replicate typing the WASD movements in the form of clickable buttons. W - Forward, A - Left, S - Reverse, D - Right, L - 90° Left, R - 90° Right, B - 180°.

Function Overview

In order to traverse the corridors and perform the rescue mission, the robot has been programmed with several different functions which can each be triggered from the GUI. These are;

Function	Behaviour
Manual movement	The robot can be controlled manually through WASD controls on the GUI. This allows for free traversal of the robot without any automatic line detection etc.
Manual Corridor Scaler	The robot will begin to move up the corridor automatically using line detectors to keep it within the boundaries. At the end of the corridor, the robot will stop and prompt the user through the Zumo Output to make a manual turn of the robot. After completing this turn, the user can input 'C' into the 'TYPE CONTROLS' component to resume the function.
Auto Corridor Scaler	The robot will begin to move up the corridor automatically using line detectors to keep it within the boundaries. At the end of the corridor, the robot will stop and prompt the user for a command through the 'Zumo Output'. Using the 'TYPE CONTROLS' component, the user can enter 'L' for a left 90-degree turn, 'R' for a right 90-degree turn, or 'B' for a 180-degree turn. The robot will then make the respective turn and continue down the corridor.

Function	Behaviour
Object Detection	This function will scan a room, either to the left or the right of the corridor, for objects/people that are in there. If an object/person is found, the robot will turn to it, buzz, and record the entry in the Zumo Output. The user needs to indicate whether the room is to the left or the right using the 'TYPE CONTROLS' component and typing either L or R depending on which side the room is needing to be searched. The robot will turn, move forward into the room, scan for objects/people, indicate to the user using the 'Zumo Output' if something was in there, and then quit out the function.

Achievements & Key Issues

Achievements;

The tasks as listed in the assignment brief have been achieved on a maze scenario at home;

Task #	Achieved (Y/N)
Task 1: Manual control of the Zumo.	Y - The robot can be manually controlled using the GUI's WASD controls. The robot can traverse both corners and corridors using manual inputs.
Task 2: Autonomous control of the Zumo	Y - The robot can be triggered to traverse the corridor. When this is done, the robot will keep within the corridor walls using it's line sensors to achieve this. The robot will additionally stop when it comes to the end of the corridor, i.e a wall.
Task 3: Turning Corners	Y - The robot will perform the same as task 2, but when it reaches the end of the corridor, it will stop and ask for the user to manually turn the robot using WASD controls. Once this movement has been done, the robot can be set off again by pressing 'C'.
Task 4: The Zumo turns autonomously around a corner	Y - Using the same function from task 3, this has been achieved by differing the code to look for a 'L', 'R' or 'B' input from the user once it hits a wall. These inputs will allow the robot to make an automatic turn either 90 degrees left, 90 degrees right, or 180 degrees backwards.
Task 5: The Zumo searches a room.	Y - Using the same turning function from task 4, the robot will be told by the user that it is outside a room. The user then inputs an 'L' for left or 'R' for right, depending on which side the room is. Once a side has been chosen, the robot will move forward slightly and perform a sweeping scan of the room. If an object is found, the robot will turn towards it and buzz, along with notifying the user via. an output to the GUI.

Task #	Achieved (Y/N)
Task 6: The T-junction	N
Task 7: The return journey	N

Key issues;

Issue	Resolution/Explanation
Line detection/corridor scaling	The issue stemmed from keeping the robot within the walls of the 'maze/corridors'. To begin with, if the robot was not centred to go up the corridor, it would go over the walls and go out of bounds. By implementing code from the Zumo32U4 > BorderDetect script, the robot could then keep within the black lines of the corridors by making a slight reverse and then turning. By adjusting the motor speeds when making the turn, I made the robot make more slight turns so it could line up more appropriately rather than overturning.
End of the corridor detection	When the robot gets to the end of the corridor, it needs to stop in order to wait for the user to make the turn. Originally, the robot would not stop on the line as it detected either the left or right line sensor first and begun to make a turn. In order to resolve this, I scripted the robot to move forward ever so slightly after meeting the IF condition of either line sensor (left or right) picking up on a border. After moving forward ever so slightly, another IF condition is introduced, checking the opposite line sensor from the previous one. This way, if both IF conditions are met, it means the robot is on a line and therefore at the end of the corridor.
C# serial inputs and outputs	In order to tie the robot controls in with a GUI, wireless communication with the XBEE adapter was required. When buttons or text is entered into the different components of the GUI, it needs to be sent over the correct serial port to reach the robot. In order to achieve the wireless communication, I had to add SerialPort mySerialPort = new SerialPort("COM5", 9600); to the C# GUI to first initialize the serial port. For the rest of the code, I had to implement mySerialPort.Open(); & mySerialPort.Close(); to allow for the flow of communication. On the Robot's script, in Arduino, adding Serial1.println("Example"); allowed for inputs back to the GUI.

Acknowledgements & Sources

The .ino script utilises libraries and sections of example scripts provided by Pololu. Libraries used include; <wire.h> & <Zumo32U4.h>. Example scripts include; Zumo32U4 >

BorderDetect, Zumo32U4 > LineAndProximitySensors, Zumo32U4 > SumoColission. https://www.pololu.com/