Maze Runner
Github: janviig

I. Executive Summary

The purpose of this project is for the Zumo robot to develop navigation and pathfinding capabilities. In the daily life both commercially and personally, these functionalities can be deemed useful to help research in complex areas where the human eye could cause complications in researching. For instance, somewhere where the chance of damage to an individual's life is present or the space is too small for humans to reach. The robot should be able to follow through and explore a maze.

Technical approach:

Hardware required:
-Pololu Zumo robot
-Arduino Uno microcontroller
-laptop with VM installed
-Arduino USB Type-C® Cable 2-in1
-4x AA batteries for Zumo robot motor
-Ultrasonic sensors x3
-14 jumper wires (can be colour coded into 4 categories)

Building physical maze:
-Cardboard
-Hot glue gun
-Hot glue sticks
-Scissors

Software:
-Have Ubuntu installed on a virtual machine (VM); could be on VirtualBox or Parallels
-Arduino IDE installed on the virtual machine
-Laptop with Arduino IDE installed
-Have a text editor installed on the virtual machine (optional); Visual Studio Code or Sublime Text

The significant outcomes from this project included the robot being able to avoid the walls of the maze – obstacle avoidance being implemented for maze walls. The robot was also able to explore the maze at a steady rate, integrating different turns for the maze with appropriate time delays. This let to the robot almost coming out of the maze by itself.

II. Introduction

This project addresses problems of sensors, navigation and pathfinding capabilities as mentioned. A maze solving robot in the context of this project, although on a smaller scale – if presented on a larger scale, has the potential to benefit in advancement of automation and future robotics. This project outlines how a robot can solve and explore a physical maze, and be able to detect any obstacles and the walls in its way through the integration of ROS2 and Arduino using ultrasonic sensors.

Background:

Robots can be extremely helpful due to their navigational capabilities able to fit in small spaces and advocate for

scientific research. If a robot can be programmed to learn how to direct turns it can explore these spaces. In today's world, something that is more commonly used in everyday households is robotic vacuum cleaners that map your house' blueprint in order to learn the path of vacumming the house. This causes the robot to take many turns and avoid obstacles, facilitating for being navigational and coordinate with different turns. Furthermore, being able to implement shortest path algorithm causes robots to be more effective and efficient, reducing run time and providing results quicker in a less amount of time.

Objectives and scope of project:

The objectives and scope of this project include using ROS2 and Arduino to have the Zumo robot avoiding hitting into any walls of a maze, navigating and exploring a maze, and eventually be able to solve a simple maze.

III. Technical Approach

1. Pass Level Feature: *robot was able to not run into any walls and detect obstacles*

A. System Overview

i. System Architecture

Figure 1 displays the system architecture of the pass level criteria for the robot – detecting maze walls and being able to avoid them. This was executed and achieved through the use of ROS2. As seen through the diagram the robot communicates with ROS2 to publish and subscribe to sensor data and consequently moving motors commands.

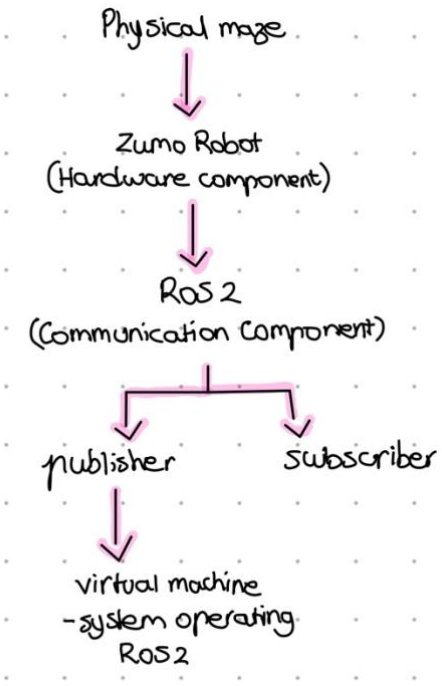

Figure 1: Architecture diagram obstacle avoidance – pass feature

ii. Flowchart of design mechanism

Figure 2 flowchart represents how the robot communicates using the pub / sub method from ROS2 to Arduino via the Zumo robot. The wall avoidance is implemented through the use of the robots ultrasonic sensors communicating the distance and causing the robot to either continue running the program, or avoid the wall depending on which directions ultrasonic sensor has recognised a hurdle.

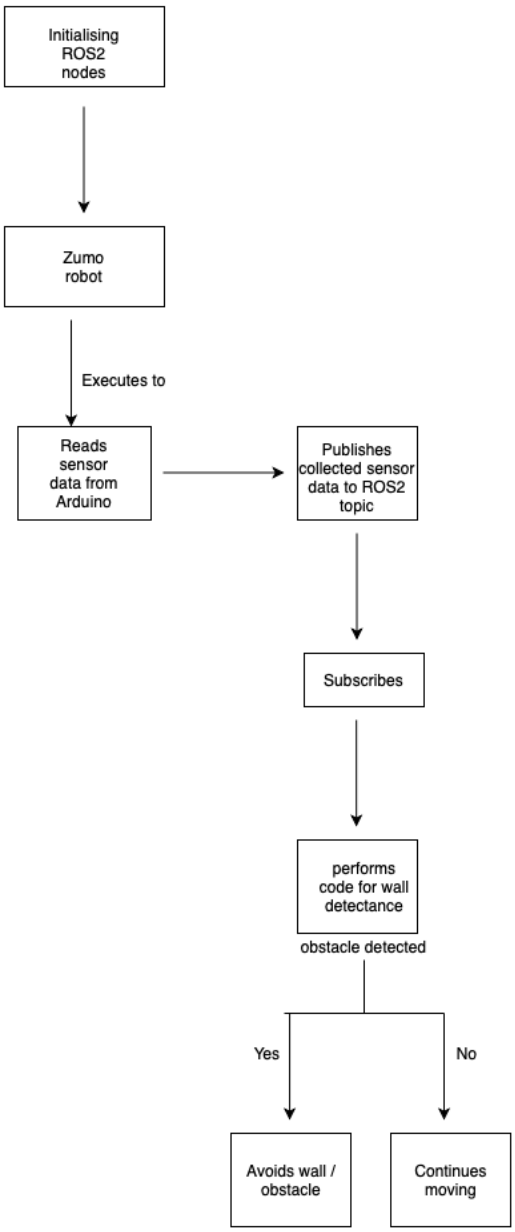

Figure 2: Flowchart of wall avoidance - pass feature

iii. Overview of system & key design decisions

The overview of this system included embedding a Pololu Zumo Robot for the hardware component, the ROS2 paradigm as the communication component and using the Arduino IDE and python scripts as the software components to execute the hardware. Sensor integration using the ultrasonic sensors was also achieved and allowed the robot to recognise objects in the front, left and right of it's movement allowing it to then avoid these obstacles through turning in opposite directions and continuing.

iv. Approach taken to design and implement application

The approach taken to design and implement this application was adding more functions to the class code 'motor_control_ultrasonic.ino' to ensure there is a separate function to cater to the vast variety of turns to the robot. For the Python script code, there was functions added so that the ultrasonic sensors would be able to recognise movement from the left, right and front directions and where the distance was less than the sensors threshold it would turn ie) wall on the left, turn to the right and vice versa.

v. Hardware and software used

Hardware:
-Zumo robot
-Arduino Uno microcontroller
-laptop with VM installed
-Arduino USB Type-C® Cable 2-in1
-4x AA batteries for Zumo robot motor
-Ultrasonic sensors x3
-14 jumper wires (can be colour coded into 4 categories)
-physical maze (materials can be found in introduction section)

Software:

-Have Ubuntu installed on a virtual machine (VM); could be on VirtualBox or Parallels
-Arduino IDE installed on the virtual machine
-Have a text editor installed on the virtual machine (optional); Visual Studio Code or Sublime Text

vi. Step-by-step guide on how everything works

1) Zumo robot connects to the virtual machine via Arduino USB
2) Compile and upload the Arduino code to the Arduino
3) Observe the serial monitor on Arduino to ensure ultrasonic sensors are recognising distance and working
4) Build, compile and execute the code on ROS2, use script code robot_bridge for the publisher
5) Run python script on the second terminal
6) Slide switch to 'on' for the robot and wait for robot to start moving
7) Test robot by placing an obstacle in front, left or right of it – once working let the robot detect the maze walls.

2. Credit Level feature: *robot was able to explore the maze*

i) System architecture

Figure 3 represents the architecture of the robot exploring and solving the maze for both the credit and distinction features. The diagram represents that the source code is used by the Zumo robot to work with the motors and the push button of the robot respectively. The code then allows the robot to explore the maze.

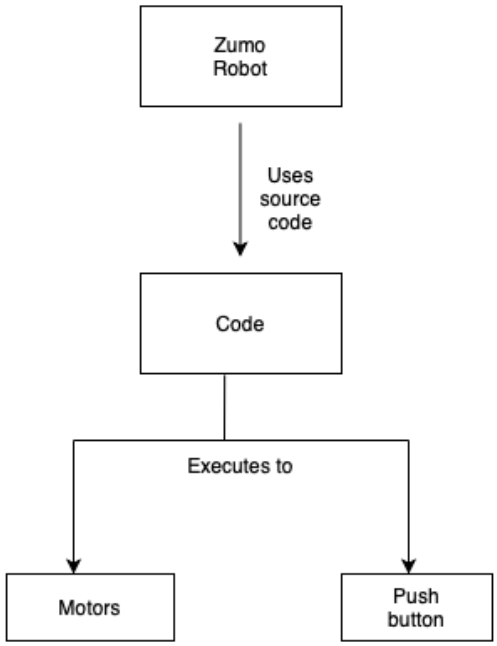

Figure 3: Architecture diagram robot exploring & solving maze: credit & distinction features

ii) Flowchart of design mechanism

Figure 4 displays the path the robot takes to explore the maze and where it stops at.

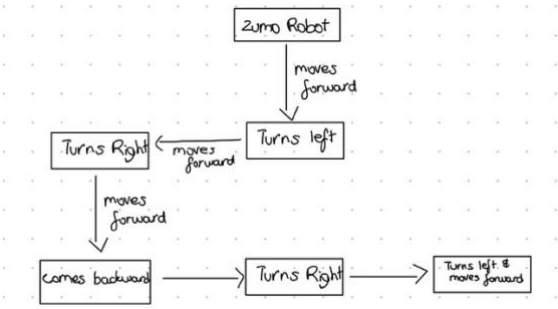

Figure 4: Flowchart of credit feature – robot exploring the maze

iii) Overview of system & key design decisions

The system overview of this feature was again using the Zumo robot, however without ROS2 in this instance. ROS2 caused problems that I have never experienced before and was eventually not able to solve, so I decided to implement the credit feature on Arduino without using the VM. This also provided the advantage of using the robot wireless, causing less restrictions for movement.

iv) Approach taken to design and implement application

The approach for the designing and implementing this application was writing a new code script for the robot. I defined the robot motors and the pushbutton allowing the robot to start. I then defined different motor movement in separate methods; forward, backward, left, right and stop. In the setup method, the motors and button was set up followed by the loop method, which called in the motor directions methods catering to the maze with time delays allowing the robot to take a pause before moving forward.

v) Hardware and software used

Hardware:
-Zumo robot
-Arduino Uno microcontroller
-Arduino USB Type-C® Cable 2-in1
-4x AA batteries for Zumo robot motor
-physical maze (materials can be found in introduction section)

Software:
-laptop with Arduino IDE installed

vi) Step-by-step guide on how everything works

1) Connect Zumo robot to a device through the Arduino USB
2) Compile and execute the code, once successfully uploaded unplug the robot
3) Slide switch to 'on' for the robot
4) Press the pushbutton once and wait for the robot to start moving
5) Robot will navigate through the maze as laid out in the code and therefore might take longer pauses at some turns as opposed to others.

3. Distinction Level feature: *robot was able to solve a simple maze*

i) System architecture

As the distinction level feature is an expansion of the credit level feature, the system architecture is the same – hence, Figure 3 can be referred to.

ii) Flowchart of design mechanism

Figure 5 and figure 6 exemplify the path the robot would take to solve the maze in a flow chart diagram and a 2d diagram of the which arrows displaying the flow of the robot navigating through the maze.

Figure 6 presents 3 sets of colour-coded arrows based off directions it is moving in the maze, they have been grouped.

1st set of instructions – blue arrows

2nd set of instructions – red arrows

3rd set of instructions – purple arrows

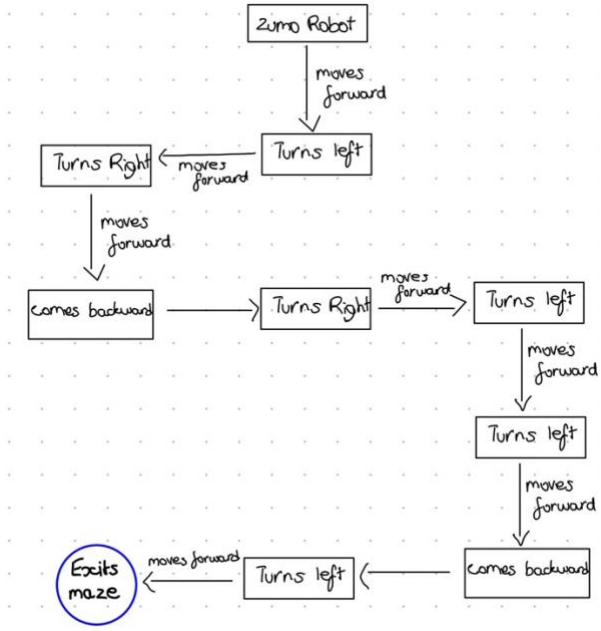

Figure 5: Robot solving the maze, path it takes to exit and solve the maze.

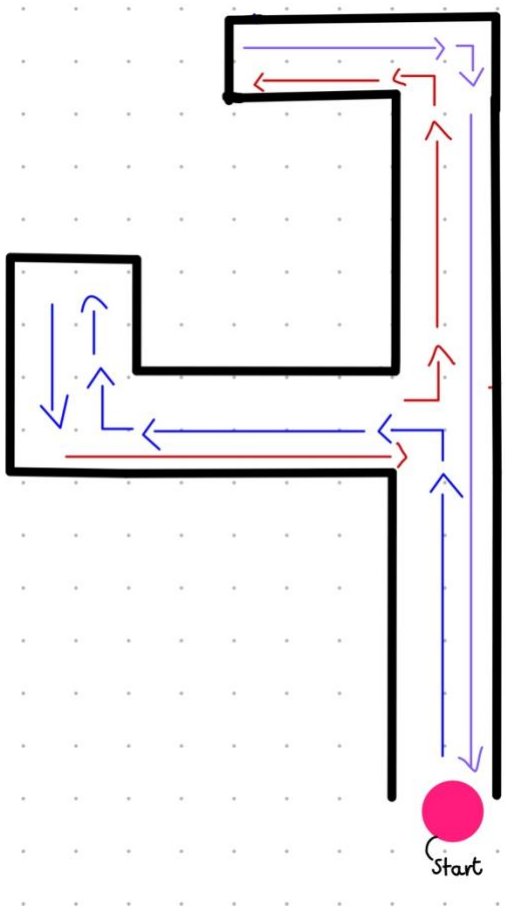

Figure 6: Diagram of the maze

iii) Overview of system & key design decisions:

The overview of this system expanded from the credit level feature. Although this feature was not fully achieved the code is correct – however due to the time delays causing issues it wasn't fully able to get out of the maze. This feature also relied on the buzzer being able to play once reached the end to indicate the path has finished and the robot has solved the maze, due to the robot not finishing in the demonstration this didn't occur.

iv) Approach taken to design and implement application

The approach that was taken to design the distinction feature was expanded from the credit features source code however, because the credit feature defined each movement direction in different methods, in the distinction feature I decided to call in the motor speed separately catering to the different turns. Depending on angles for different turns, motor speeds would have to be adjusted along with different runtime of each direction that would need to be done. Therefore, the code was differentiated a lot from the credit features source code. The maze code from the Arduino side doesn't implement an algorithm and therefore is catered to the maze created for this project only, limiting its capabilities to solve other mazes.

v) Hardware and software used

Hardware:
-Zumo robot
-Arduino Uno microcontroller
-Arduino USB Type-C® Cable 2-in1
-4x AA batteries for Zumo robot motor
-physical maze (materials can be found in introduction section)

Software:
-laptop with Arduino IDE installed

vi) Algorithms

Although I wasn't able to implement any algorithms in my code, this is one I researched on that I found highly intriguing and, in the future, would implement this.

LSRB Algorithm:

Left, Straight, Right, Backward algorithm works in the following order, this can also be interpreted as the pseudocode of the LSRB algorithm:

1. Start at entrance of maze
2. Left**,** turn left if there is a turn
3. If there is no left turn, continue driving straight
4. If there is no left turn or can't continue straight, turn right
5. If there is not a left turn, straight or right turn – the robot has reached a dead end and will have to go backward ie) u-turn to the last intersection
6. Process will be repeated till the robot can solve the maze

Figure 7 displays a series of diagrams of how a LSRB algorithm works to solve a path given intersections:

PATH GIVEN:
$S \rightarrow R \rightarrow L \rightarrow B \rightarrow L \rightarrow R \rightarrow L \rightarrow R$

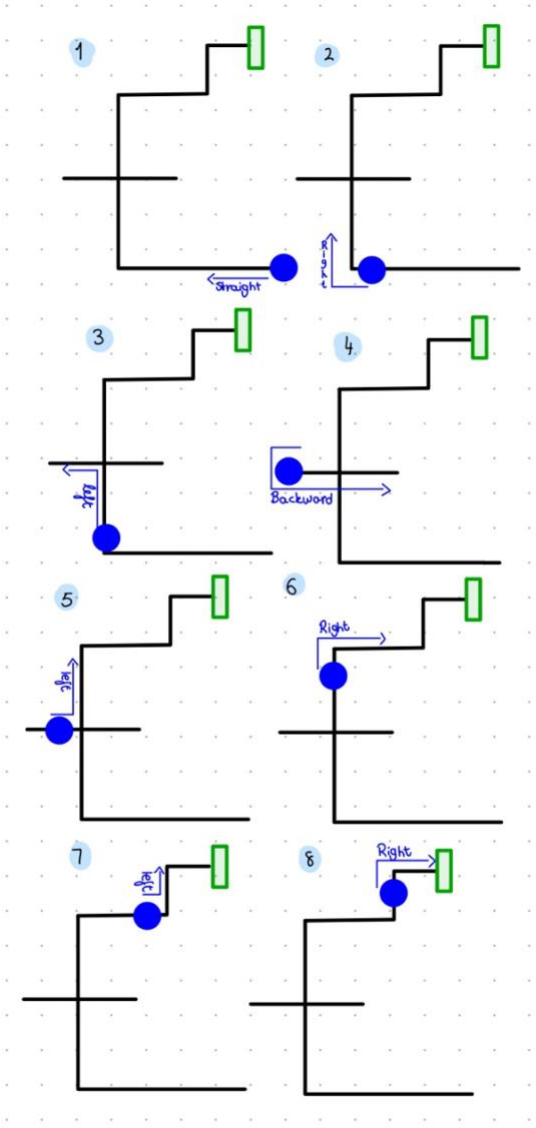

Figure 7: LSRB Algorithm path

vii) Step-by-step guide on how everything works

Steps are all the same as C-level feature – please refer to 2. C-level feature

IV. Discussion

My maze was supposed to be one that can be followed through the infrared sensors rather than a physical maze, however the line following sketch suddenly stopped working on my robot, so I decided to build a physical one instead. Building a cardboard maze from scratch although seems easy took more critical thinking and problem solving than anticipated. Once the maze was built it had to constantly be reformed again after the robot kept driving into every wall. Once the robot was able to detect walls using the ultrasonic sensors, it was able to stay in the maze without hitting anything exactly as I wanted.

Working with ROS2 was problematic with the robot detecting walls of the maze. Default errors causing it to stop working even with class code that wasn't further edited. So I decided to use ROS2 for the pass level feature of the robot not driving into walls and detecting obstacles, and Arduino for the credit and distinction features – exploring and getting out of the maze.

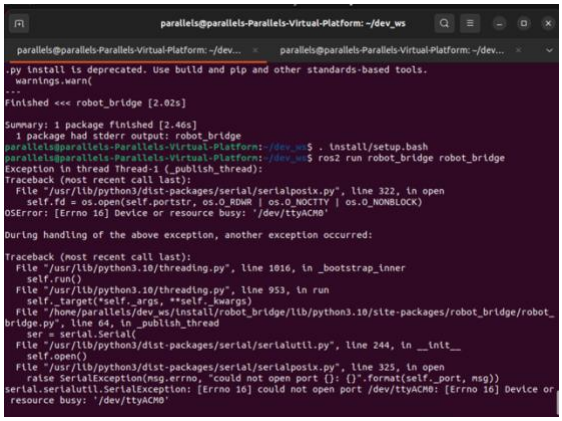

Figure 1: Default error on ROS2 that was unable to be solved.

One of the biggest limitations of this project is that there was no algorithm implemented within the code, which meant the robot was only catered to my maze. The robot also didn't use the

ultrasonic sensors for the credit and distinction level features rather used the 'delay' function to navigate through the maze. Another limitation – refer to figure 8 - the port was being used by Arduino as it was meant to be, to then be used in conjunction with the Python script however this has never occurred before and searching up error also further proved no success.

Therefore, in the future I would implement an LSRB algorithm for the maze to navigate itself through different mazes. I would also implement a shortest path algorithm for the robot to solve a maze in different ways and eventually analyse the fastest route. Using the ultrasonic sensors and even implementing an LCD screen in the future to provide updates would be useful to work with the robot and for a more intuitive prototype. I would also utilise the line sensor and infrared sensors inbuilt in the robot in the future so that testing is easier as opposed to building or remodifying a physical cardboard maze every time.

V. My Project Journey

My project journey encompassed a lot of "ups and downs". When I started researching about a robot solving mazes, I discovered the LSRB algorithm behind it which sounded so fascinating and as I researched more about the algorithm I can confidently say I now so much about this algorithm. I tried to implement it in my code – both ROS2 version and Arduino but it was difficult.

Things I have learnt from this project that I can boast about are; learning about the LSRB algorithm, building a maze, calibrating a robot and getting it to follow through a maze.

Although the distinction part is not fully shown and achieved in the video, the code exemplifies the path the robot was to take and how this would allow it to come out of the maze – however, due to time constraints from myself, the 'delay' command in the program it was not presented fully working in the demonstration video.

My favourite moments from this project were discussing the project ideas with my friends and exchanging feedback and method to improve our findings. I also enjoyed working on the code and eventually getting the robot to explore the maze, in the future I'd like to be able to get the robot to solve any maze using the LSRB algorithm on white paper with lines drawn and solve any given maze in the shortest amount of time.

VI. References

[1] "7.3. Maze Solver.." Pololu https://www.pololu.com/docs/0J57/7.e [accessed 14/5/2023]

[2] M.M Nayeem. Medium. "Coding a Line Follower Robot for Maze using LSRB Algorithm and finding its Shortest Path" https://towardinfinity.medium.com/coding-a-line-follower-robot-using-lsrb-and-finding-the-shortest-path-d906ffec71d [accessed 20/5/2023]

[3] P. Craven. Youtube. "Running the motors with the Zumo robot."

https://www.youtube.com/watch?v=k2DDa9nwVx0&t=15s&ab_channel=ProfessorCraven [accessed 24/5/2023]

[4] BenPololu. Youtube “Pololu 3pi Line Maze Solving Robot.” https://www.youtube.com/watch?v=mJV-KDqHgDQ&ab_channel=BenPololu [accessed 24/5/2023]

[5] ChatGpt https://chat.openai.com