CG2111A Engineering Principle and Practice

Semester 2 2022/2023

"Alex to the Rescue"

Final Report

Team: B04-5B

Name	Student No.	Main Role
Davian Kho Yong Quan	A0252623L	Software, Firmware, Hardware
Daryl Tay Chin Kian	A0252257E	Firmware
Sparsh	A0265162H	Software, Hardware
Chen Xiao Kang	A0254466X	Software

Link to Animated CAD Render: https://youtu.be/AahvEI9D77U

Section 1 Introduction

PROJECT OBJECTIVE

Earthquakes often happen instantly and devastate the lives of many, as observed from the most recent Turkey-Syria Earthquake that claimed the lives of up to 47,000 in 2023 [1]. This illustrates the constant need for an advanced search and rescue robot that can scan the rubble of the devastated area quickly to look for casualties. Thus, “Alex”, our search and rescue robot is designed to navigate around obstacles by mapping the surroundings accurately, controlled remotely to avoid sending humans into dangerous environments and to identify victims that may be in the vicinity.

MAIN DESIRABLE FUNCTIONALITY

eatures	Description
Remote Networking	Remote Networking would allow ALEX to operate without the physical presence of a human, allowing it to venture into places humans cannot.
Environment Mapping	The LIDAR sensor should scan the environment accurately, detecting any walls or objects. We should be able to generate an accurate and detailed map for the user to use to navigate ALEX.
Stability & Mobility	ALEX should be able to move and turn quickly to navigate smoothly and safely around a potentially dangerous environment without overturning.
Energy Efficiency	ALEX should ideally be as energy efficient as possible, to allow for longer operation lifespan on limited battery power.
User Friendliness	ALEX should be easy to use and not require complicated commands to control. The user interface should be easily comprehended by the user.

Figure. 1 List of functionalities

Section 2 Review of State of the Art

1. ***Delta extreme*** - The Delta Extreme is an intuitive, remotely operated robotic system designed for reconnaissance during search and rescue missions capable of fitting through confined spaces and hazardous environments [2]. It comes outfitted with a Colour ¼" EX View CCD camera with high-intensity LED to easily observe the terrain around it and can be operated using a remote proportional joystick.

Its strengths include being able to navigate tight spaces due to its shape-changing capabilities and being waterproof up to 10m. It also comes outfitted with specially designed track belts to provide optimal traction in a variety of environments. It is also very portable and is light enough to be carried via a single backpack. However, it has a limited battery life of around 2 hours, and its small stature means that it is unable to tow heavier weights.

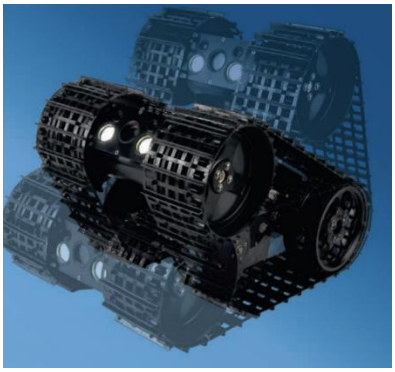

Figure. 2 A model of Delta Extreme

2. ***Thermite RS1-T2*** - A remote firefighting robot created by Howe and Howe Technologies to assist firefighters in their search and rescue missions involving hazardous environments [3]. It comes with industrial-grade treads that can ascent slopes up to 70% gradient and is capable of outputting more than 4730 litres of water per minute. It comes fitted with an inbuilt LCD monitor and offers both remote and tethered controls.

Its strengths include the capability to tow up to 800kg in weight and come with heat shields, lessening the physical burden on actual firefighters in actual scenarios. The remote distance is around 300m, allowing operators to rescue victims from a safe distance. However, it does come at a relatively pricey 90 thousand dollars and may not fit in very small hallways due to its size.

Figure. 3 A model of Thermite RS1-T2

Section 3 System Architecture

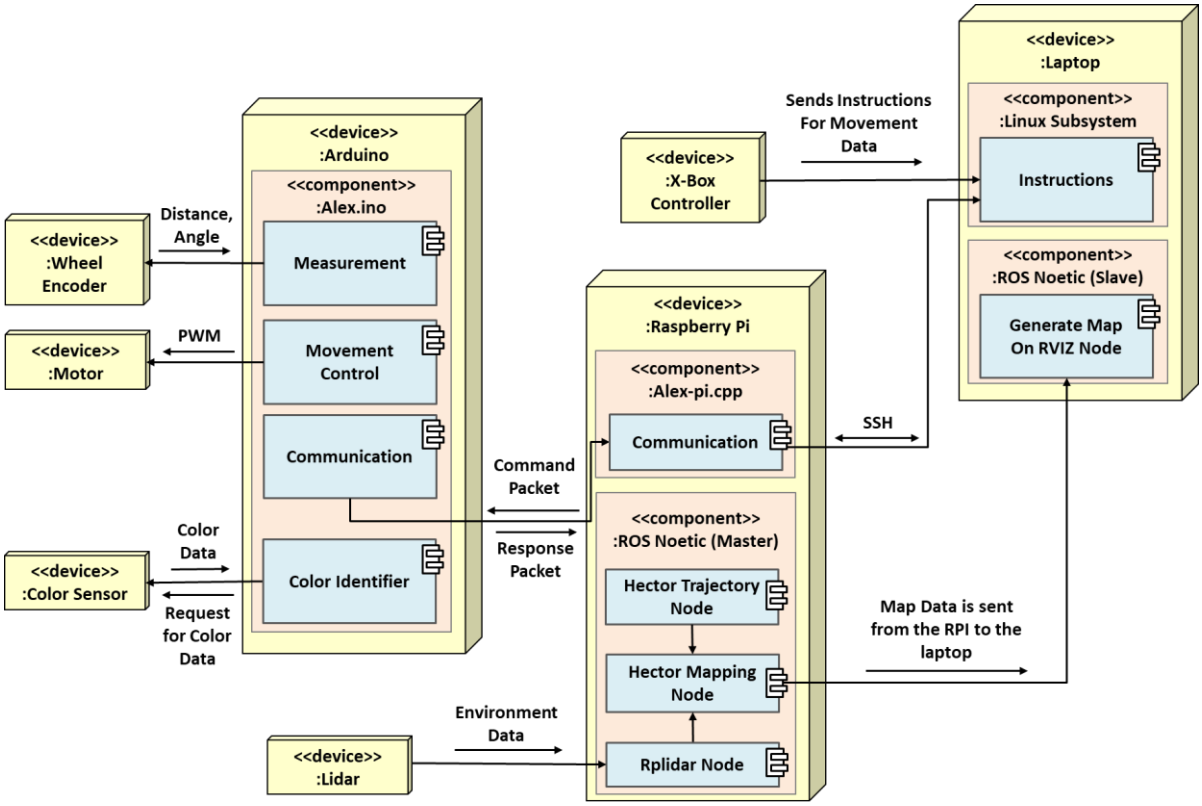

Figure. 4 Simplified UML Deployment Diagram

Section 4 Hardware Design

Figure 5. CAD render of ALEX

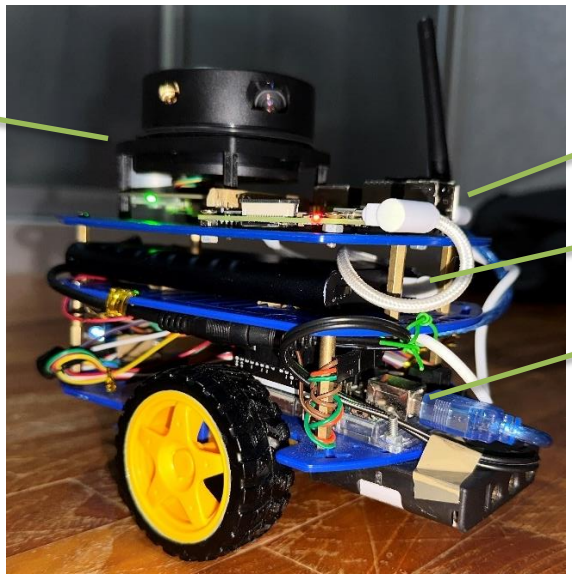

Figure 6. Left View of ALEX

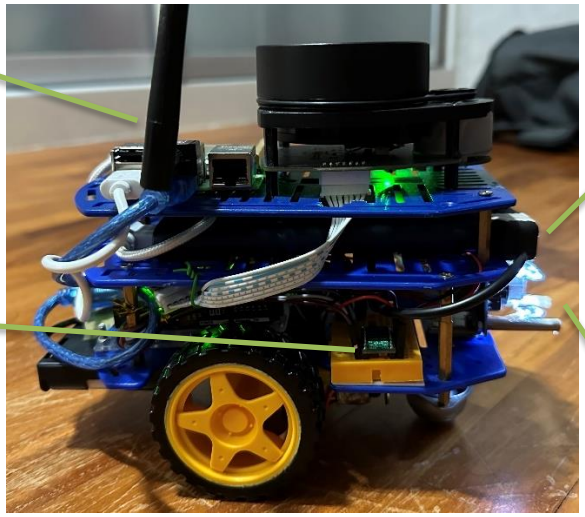

Figure 7. Right View of ALEX

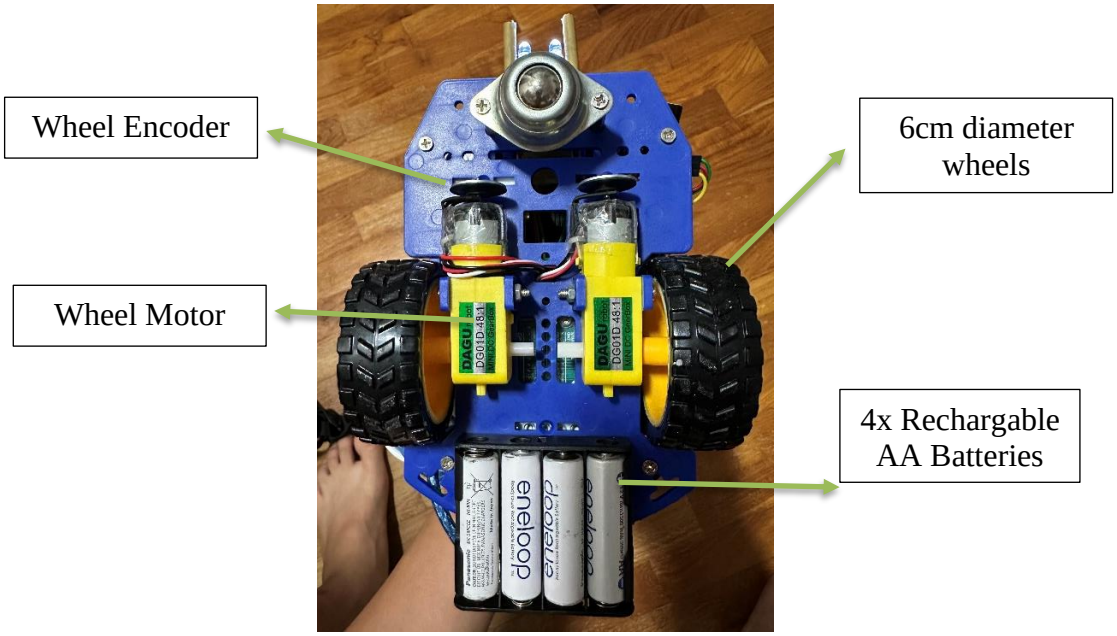

Figure 8. Bottom view of ALEX

Raspberry PI (Top Layer)

The Raspberry PI(RPI) was chosen as our primary lightweight single board computer due to its small footprint yet large flexibility in usage. Its onboard CPU and RAM are powerful enough to support complex algorithms that may be required during Search and Rescue, yet light enough to not require too much power. The onboard Wi-Fi module makes remote communication possible, and therefore increases the versatility and usage of ALEX. It is also the main central point of communication, receiving data from the Arduino, LIDAR sensor while transmitting data to the laptop for the user to interpret. It is also responsible for running the SLAM mapping algorithm.

Arduino Uno (Bottom Layer)

The Arduino Uno helps to decentralize computing from the RPI, taking in data from the wheel encoder, motor, and colour sensor. Instructions from the RPI that are sent to the Arduino will be read by the Arduino and a signal will be sent to the respective devices which are the wheel encoder, motor, and colour sensor. The Arduino also allows for easier connections due to its ability to interface with pins and headers via its GPIO slots, while allowing for simple bare-metal programming to maximize efficiency.

LIDAR Sensor (Top Layer)

The LIDAR sensor is used to scan and provide the environment data to the RPI to generate a map so that the user can navigate their way through the environment. This will allow the robot to have high mobility as the map generated from the environment data obtained by the LIDAR sensor will allow the user to avoid obstacles and move around with ease.

Colour Sensor (Bottom Layer)

The colour sensor provides the object's colour to the user controlling the robot. It detects the object's colour and sends colour pulse width (PW) values to the Arduino. The Arduino will translate the PW values into a colour wheel and sorts the colour before sending it to the RPI. The RPI will state the colour of the object by matching with existing colour data. This is so that the victims can be identified correctly using the colour sensor.

Laptop

The laptop is used to set up an SSH connection with the RPI for remote command and control. It will also run Rviz (An Imaging Software). Any instructions that are sent from the laptop will be sent to the RPI. The RPI will then read the instructions and send commands to the respective devices. We chose to use the laptop to run Rviz, in doing so reducing the workload of the CPU and latency on the RPI.

Wheel Encoder (Bottom Layer)

The wheel encoder provides the distance the robot has traversed to the Arduino. Every time ALEX moves, the wheel encoder sends a signal to the Arduino, which will keep track of the signal in the form of ticks. When the RPI sends a command to the Arduino to obtain distance of the robot, the number of ticks stored will be returned.

Xbox Controller

The Xbox controller is connected to the laptop with useful controls like movement, power setting, and colour detection mapped to different buttons on the controller. The controller allows easier and more intuitive controlling of Alex. The controller movements are shown in the appendix.

Additional Components

Component	Feature
4 AA Batteries and power bank (Batteries are beneath the bottom layer and the power bank is on the second layer.)	The batteries supply power to the motors and the power bank supplies power to the RPI.
DRV8833 Dual Motor Driver (It is at the bottom layer.)	It is used to control the 2 motors.
Motor and wheels (Motor is attached to the bottom layer of the robot and wheels are connected to the motor.)	To allow the robot to move.

Section 5 Firmware Design

A high-level algorithm is being used on the Arduino and there is a communication protocol that is being followed for the Arduino to communicate with the RPI. Figure 6 shows the algorithm that Arduino uses when it receives command packets from the RPI.

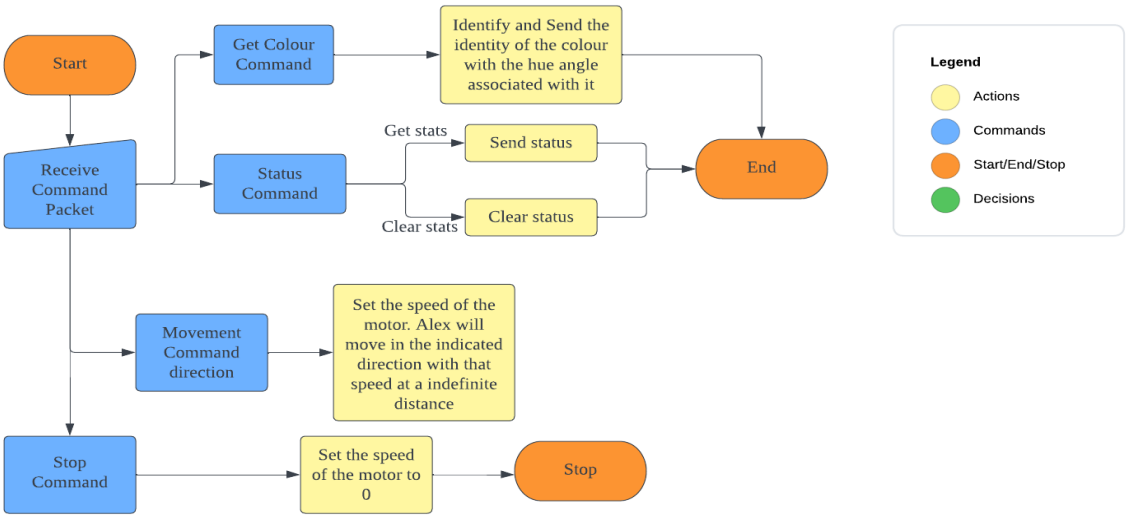

Figure. 9 Arduino Receiving Command Packet

When the user inputs 'o' on the keyboard or 'Y' on the Xbox controller, a command to get colour will be sent from the laptop to the RPI and then to the Arduino. The colour sensor will detect the colour and send RGB values back to the Arduino. We implemented an algorithm to convert the RGB values to a hue angle which corresponds to the colour identified on the colour wheel. The Arduino will then send a response back to the RPI which includes both the hue angle and the colour identified.

The laptop will send commands to the RPI and the RPI will send commands to the Arduino to execute. Arduino is connected to the RPI via USB. The baud rate between the Arduino and the RPI is 9600. Table 1 shows the structure of the packets being sent from the RPI to the Arduino and vice versa.

Packet variable	Description
Char packetType	The type of packet(response packet, command packet, etc)
Char command	It stores the commands/responses
Char dummy[2]	Padding to make up 4 bytes
Char data[32]	It stores a string data
Uint32_t params[16]	It stores parameters as 32 bit integers.

Table 1: Packet variables

Packet Type	ID associated with packet type
PACKET_TYPE_COMMAND	0
PACKET_TYPE_RESPONSE	1
PACKET_TYPE_ERROR	2
PACKET_TYPE_MESSAGE	3
PACKET_TYPE_HELLO	4

Table 2: Packet types and their associated ID

Each packet type has an ID assigned to it. The packet types are stored in the packetType variable in the main TPacket. This allows the Arduino to understand what actions need to be taken for the type of packet received. Before sending commands over to the Arduino, the RPI will serialize the command packet into a stream of bytes. This is done by getting a pointer to the structure of the packet, copying it into an array of char and including the information on packet length and checksum. Then it will be sent to the serial port. The Arduino will then receive the stream of bytes and convert it back to the packet structure by deserializing the stream of bytes. This is done by getting a pointer to the structure and copying the buffer of bytes to that pointer. The checksum will also be computed. If the checksum is not correct, "bad checksum" will be printed. If the command packet sent from the RPI to Arduino requires a response, a response packet will be sent back from the Arduino to the RPI.

Response types	ID associated with Response Type
RESP_OK	0
RESP_STATUS	1
RESP_BAD_PACKET	2
RESP_BAD_CHECKSUM	3
RESP_BAD_COMMAND	4
RESP_BAD_RESPONSE	5
RESP_COLOUR	6

Table 3: Type of responses

Table 3 shows the different types of response packets that the Arduino might send back to the RPI upon receiving the command packet from the RPI.

Command Types	ID associated with Command Type
COMMAND_FORWARD	0
COMMAND_REVERSE	1
COMMAND_TURN_LEFT	2
COMMAND_TURN_RIGHT	3
COMMAND_STOP	4
COMMAND_GET_STATS	5
COMMAND_CLEAR_STATS	6
COMMAND_IDENTIFY_COLOUR	7

Table 4: Type of commands

Table 4 shows the different command packet types that the RPI could send to the Arduino. The Arduino will read these commands and carry out the actions that are required.

Section 6 Software Design

Summarised High Level Steps:

1. Initialization
2. Scanning and Mapping
3. Receive User Command
4. Carry Out Command
5. Repeat steps 2 to 4 until the navigation is over

In order to carry out its search and rescue mission, ALEX needs to have environmental mapping, while using telecommunication to navigate around physically. Figure 8 below shows a flowchart describing how ALEX carries out its tasks.

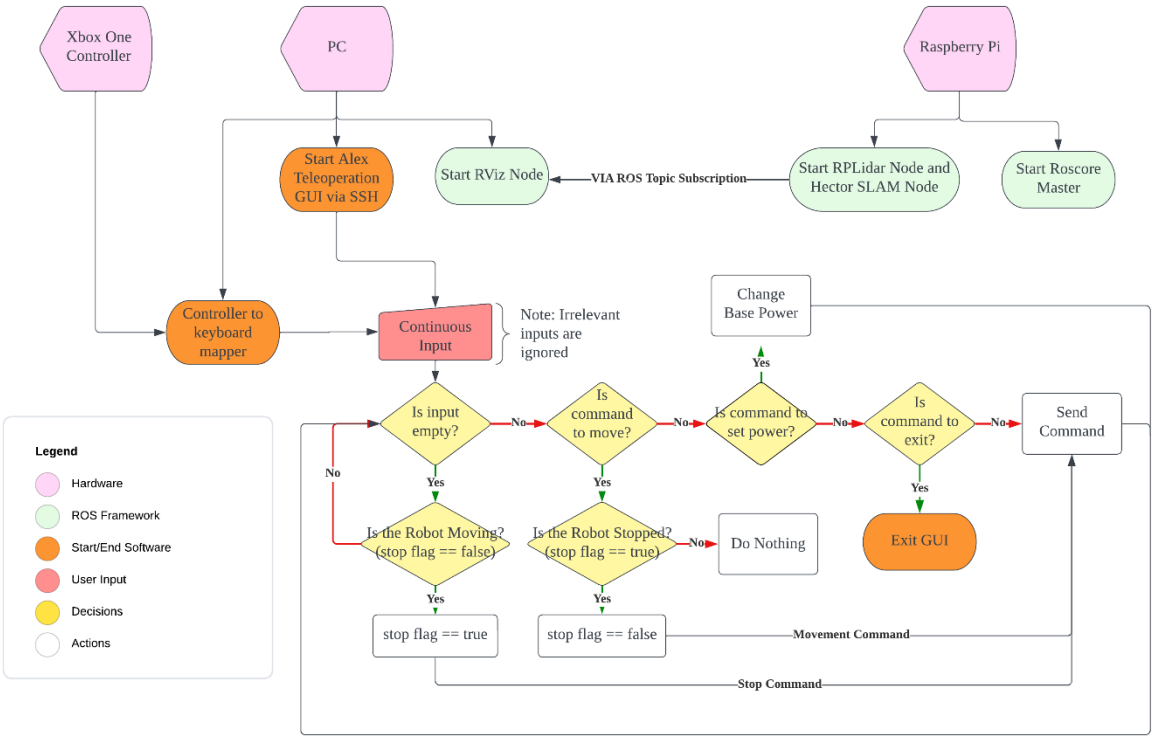

Figure 10. Control Flow for ALEX

Energy Efficiency

In view of our main desired functionality, which is for the RPi to have energy efficiency, the visualization of the environment (RViz) was offloaded onto the PC using the ROS Topic Publisher/Subscriber framework. The RPi would take raw readings with the RPLidar and process it using the Hector SLAM algorithm. The output data would then be broadcasted to all machines connected to the ROS MASTER (the RPi), which in this case is our local PC. Then, RViz can "subscribe" to that service and show us the mapped environment, as shown in Figure 10. This would create a reduced CPU usage of up to 30 Percent, hence reducing power consumption.

User Friendliness

To achieve User Friendliness, we opted to implement a controller, as its controls are intuitive to use (stick for movement, buttons for features). To achieve that, a controller to keyboard mapper was used, as shown in Figure 12. Since the native program taking in inputs accepts keyboard values, in the event the controllers fail, the user can fall back to a more rudimentary 'WASD' form of controlling, with other keys mapped to other functions which can be found in Table 5.

The provided source code uses a command-and-control interface style, where the user is prompted to type in individual commands, while manually setting the power and distance to travel. Using the C++ NCurses Library*, we were able to compartmentalize and organize all information relevant to ALEX and create a Graphic User Interface (GUI) with all information ready at a moment's notice. Additionally, we used a getch(); function to obtain input rather than scanf(); as it takes in user input continuously without having to press enter. This would allow pressing and holding of keys rather than continuous singular inputs. There would be a baseline power set at 75, with keys that can increment/decrement the value.

To run ROS, the source of the framework is required to be entered in bash before running any ROS-related commands. To reduce the number of commands required on the user's end, the .bashrc file in both RPI and local machine was edited, allowing the source commands to be automatically executed upon starting up bash.

```
source ~/cg2111a/devel/setup.bash
export ROS_IP=192.168.1.165
export ROS_MASTER_URI=http://192.168.1.165:11311
```

Figure 11. .bashrc configuration on RPI

```
source /opt/ros/noetic/setup.bash
source ~/Desktop/slam/devel/setup.bash
export ROS_HOSTNAME=127.0.0.1
export ROS_MASTER_URI=http://192.168.1.165:11311
```

Figure 12. .bashrc configuration on Local Machine

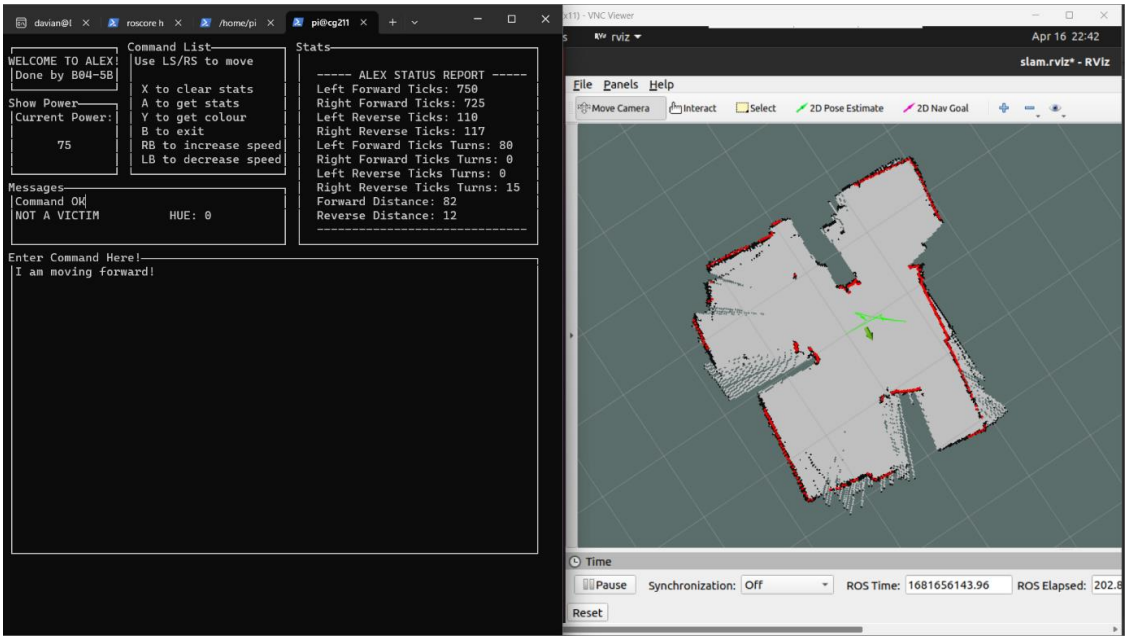

Figure 13. ALEX Program with GUI on the left, Hector SLAM visualized with RViz on the right

Environment Mapping

The RPLidar is known to be sensitive to sudden changes, where a high rate of turn would cause unwanted artifacts in the map such as map drifting. This is most certainly undesired, and its mapping sensitivity was thus reduced by increasing its “map_update_angle_thresh” and “map_update_distance_thresh” value in the launch file. “map_resolution” was also reduced to decrease the size of each pixel and increase fidelity. Additionally, a new node called Hector Trajectory Server was added as well**, for the map to indicate its traversed path, allowing the user to be aware of areas already explored.

```
    <param name="map_update_distance_thresh" value="0.05"/> <!-- Original: 0.02 -->
    <param name="map_update_angle_thresh" value="3.0"/> <!-- Original: 0.01 -->

    <param name="map_resolution" value="0.02"/> <!-- Original: 0.05 -->
    <param name="map_size" value="1024"/>
    <param name="map_start_x" value="0.5"/>
    <param name="map_start_y" value="0.5"/>

  </node>

  <node name="hector_trajectory_server" pkg="hector_trajectory_server" type="hector_trajectory_server">
    <param name="target_frame_name" value="map"/>
    <param name="source_frame_name" value="laser"/>
  </node>

</launch>
```

Figure 14. Hectormapping.launch configuration file

For colour identification, the colour sensor was installed in the front of the chassis. The GUI processes the request and sends the request to the Arduino. With a return packet, the identified colour will be printed as a message in the “Messages” window, along with its colour hue.

Stability and Mobility

With the implementation of the controller, there are 2 available joysticks. To increase the sensitivity of the movement, we mapped the left joystick for long periods of movement and the right joystick for fine movement in tight spots. This would allow for better maneuverability in tight spaces that may be required, or when the colour sensor requires a closer measurement, while reducing the requirement for multiple inputs when moving larger distances.

*More information on the software can be found within the attached C++ code.

**Launch files from both RPI and Local Machine have been included as well.

Section 7 Lessons Learnt – Conclusion

During the past weeks of implementing novel hardware and software components to map and navigate a new environment, our team has learnt many lessons and made many mistakes as well.

Lessons Learnt

Firstly, we learnt how to troubleshoot the numerous errors we encountered when dealing with unfamiliar software systems like ROS and RViz. We had to search for the specific problem within the ROS documentation, online tutorial videos and even online forums when we were stuck as our issue may have been resolved in the past. This was important as a crucial skill in software engineers is the ability to effectively search and solve the problems they encounter.

We also learnt the value of project scope management, as the implementation of any feature can cause the complexity to increase and require more time to be properly implemented. For example, the implementation of the GUI caused issues in the previously flawless movement of Alex and required more time and effort to perfect. We realized we had to manage our time and implement features that are feasible in the given time span.

Greatest Mistakes

One major mistake we made was not planning the placement of different components of the robots before assembling them. For example, the power bank was placed at the bottom of the robot initially and it was difficult to access and charge. This resulted in us wasting a lot of time dissembling and reassembling most of the components and wires, which could have been put to better use.

Another mistake is that we did not focus heavily on the hardware features and mainly focused on the software aspects. Features like buzzers and ultrasonic sensors could have been another method to identify obstacles that are close to our robot and sound out accordingly, which could differentiate our robot as being more unique and advanced.

References

[1] “Thousands dead, millions displaced: The earthquake fallout in Turkey and Syria“, The Guardian [Online]. Available: https://www.theguardian.com/world/2023/feb/20/thousands-dead-millions-displaced-the-earthquake-fallout-in-turkey-and-syria. [Accessed April 16, 2023]

[2] “CC Delta Extreme 2016 – Nexxis", Nexxis [Online]. Available: https://nexxis.com/wp-content/uploads/sites/3/2017/09/CC-Delta-Extreme-Spec-Sheet-17.pdf. [Accessed April 16, 2023]

[3] “TREAD BOLDLY - HOWE AND HOWE”, Howe and Howe [Online]. Available: https://www.howeandhowe.com/sites/default/files/_documents/2021_Thermite%20RS-3_DataSheet_Digital.pdf. [Accessed April 16, 2023]

Appendix

Keyboard	Xbox One Controller	Function
w	Left Stick (Big Movement) Right Stick (Small Movement)	Move Forward
a		Turn Left
s		Move Backwards
d		Turn Right
q	B	Exit Program
o	Y	Get Colour
g	A	Get Stats
c	X	Clear Stats
p	Right Bumper	Add Power (+5)
m	Left Bumper	Reduce Power (-5)
r	Back Button	Force Refresh Message Window
-	Start Button	Shortcut to start program (Change Directory and ./alex-pi)
-	D-Pad Right Button	Shortcut to force close program (CTRL+C)

Table 5. Controller-Keyboard Mapping

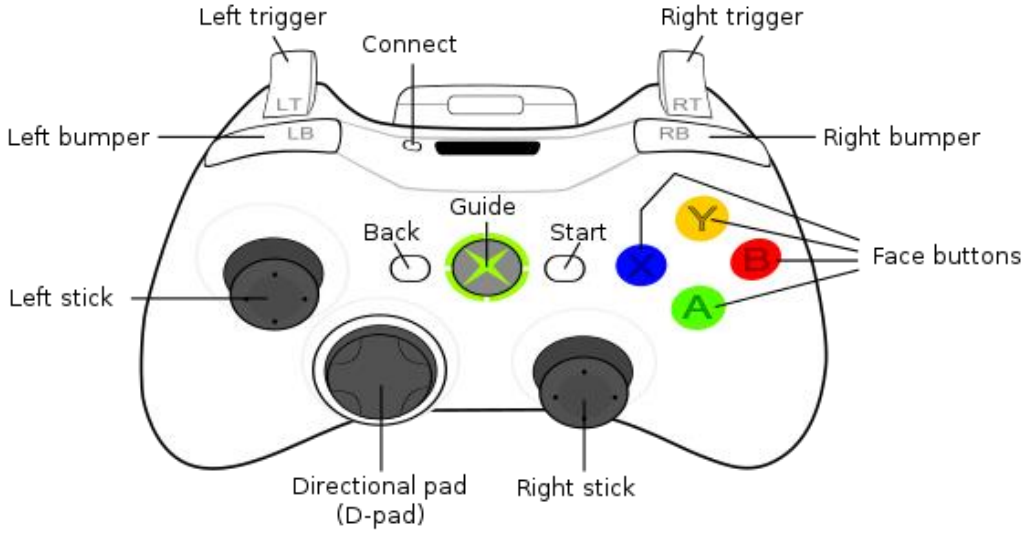

Figure 15. Controller Button Layout

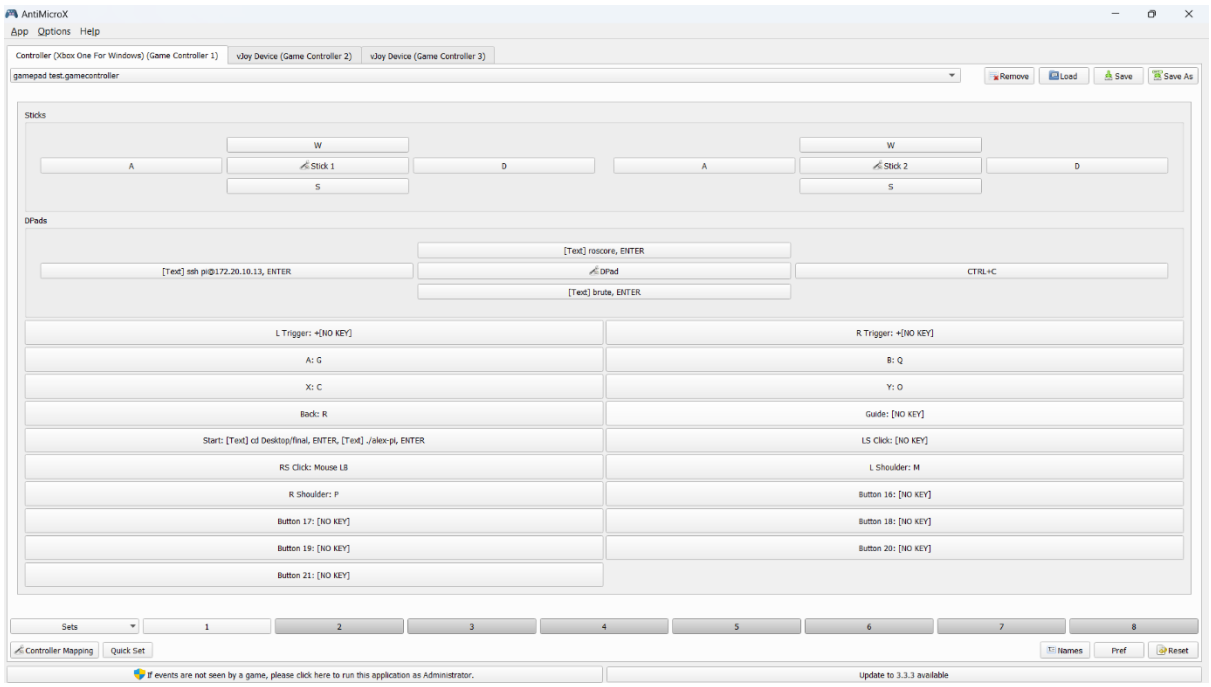

Figure 16. Controller-Keyboard Mapper