CG2111A Engineering Principle and Practice
Semester 2 2022/2023

"Alex to the Rescue"
Final Report
Team: B04-2A

Name	Student #	Main Role
Samuel Chung Weng Kit	A0258878B	Hardware
Willson Han Zhekai	A0252890Y	Software, pilot for Alex
Yong Chen How	A0252647Y	Hardware
Zhu Minghui	A0257706X	Software, Firmware

Section 1 Introduction

We are tasked with creating a robot (Alex) with Search & Rescue functionalities. Alex will be placed in a 5 by 7 grid with an area of $3m^2$, which we cannot see. This simulated environment contains obstructions and walls, and even a hump, that Alex needs to navigate around. Here, Alex needs to generate a map of the environment and identify “casualties” that are either red or green objects. At the end, Alex needs to reach a designated parking spot.

Alex’s main functionality is environment mapping. To do this, Alex will be teleoperated from a laptop. We use TLS programming to create a secure connection between our laptop (client) and the R-Pi (server). This allows commands to be sent from the laptop to R-Pi and then to the Arduino via UART, allowing for movement. Alex can move forwards and backwards and turn on the spot, according to distance or turning angle respectively and speed.

To generate a map, the LIDAR captures data of the environment using SLAM via ROS. RVIZ is run on the R-Pi to generate a real-time map of the environment, allowing us to see the map to navigate Alex.

To identify the “casualties”, a colour sensor is attached to the front of Alex.

We also added some additional features. Firstly, we are teleoperating Alex with a PS4 controller. Secondly, we are using an ultrasonic sensor and buzzer to measure the distance between the casualty and Alex and buzz at the optimal distance for our colour sensor, allowing us to scan for its colour accurately.

Section 2 Review of State of the Art

The RMUS Find, Inspect, Detect and Observe (FIDO) is a tele-operated quadruped robot built on the Boston Dynamics SPOT platform. It has a multi-camera payload, with a colour ring RGB camera that rotates 360°. It also comes with a thermal camera and another RGB camera with 30x zoom. It can capture images and video, even in low light conditions, generating a scan of the environment. These cameras can be modified using its GRPC API. FIDO has an arm and gripper for carrying heavy objects and manipulating other objects. This comes with an LED illuminator and high resolution camera to inspect objects before taking action. The arm is controlled from a tablet interface and API, and control can be manual, semi-automated, or fully automated. FIDO is also equipped with high-sensitivity microphones and amplified speakers for two-way communications with people on the field. Its powerful radio communications system can withstand harsh environments.

Strengths: Able to traverse harsh environments and manipulate objects to save lives without putting operator at risk

Weaknesses: Too large to access smaller spaces, short battery life of 90mins

Drones, like the DJI Inspire platform, are tele-operated aerial platforms that can be purpose-built for Search & Rescue purposes. They are equipped with wide-angle cameras and thermal cameras to capture image and video, allowing us to generate a scan of the environment. They can be equipped with speakers to broadcast warnings, or built to carry and deliver resources rapidly. Drones can be manually operated or automated, with software that is able to detect and avoid obstacles whilst following people on the ground. FlytNow, a cloud-based drone fleet management system, provides a software platform to control multiple drones for larger scale operations. Drones in a fleet are synced over the Internet and can be controlled from a dashboard on laptops or tablets.

Strengths: Fly fast to respond quickly to situations, challenging terrains are not an issue for drones

Weaknesses: Hard to navigate in tight spaces, only able to obtain data and cannot help personnel on the ground physically

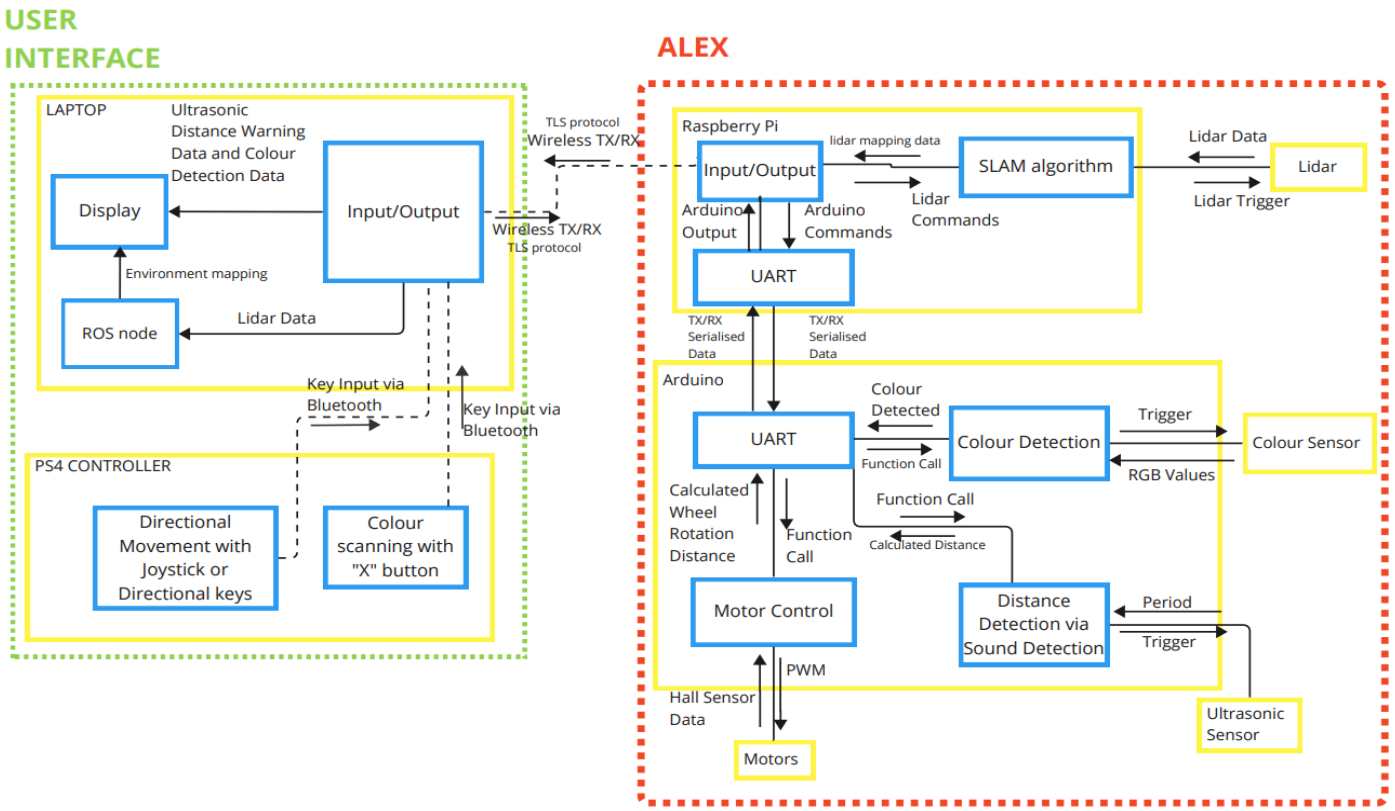
Section 3 System Architecture

3.1 Overview

Before going into the specifics of our Alex, it is worth going over the main ideas in the design of Alex. The Raspberry Pi(R-Pi) will be the main processor on Alex. The R-Pi will take inputs wirelessly from the user, inputs from the LIDAR, inputs from all the sensors on Alex via the Arduino's UART ports and provide outputs to the LIDAR, the user interface and all the sensors, again, via the USART ports. This is done in order to allow our Alex to effectively complete the relevant tasks wirelessly, without our own visual input(eyes).

3.2 High Level System Components

The following chart will show all the components in Alex and how they interact with each other to accomplish the tasks at hand. The diagram shown is colour coded such that the yellow boxes represent the hardware components along with the label of each component at the top left of each yellow box. Each blue box within the yellow boxes represents each software feature of each hardware component. The blue box also contains a description of what each feature does. The solid, black arrow headless lines show that there is a wired connection between the components whereas the dotted, black, arrow headless line shows a wireless connection between the components. In addition, the headed arrows show the direction of the data being transferred and a short description on the type of data. Lastly, a green box and a red box were used to show the components of the user interface and the Alex respectively.



3.3 Changes Made

A few slight changes were made from the last system design of Alex to aid in the reliability of the manoeuvrability of Alex throughout the course. We decided to wirelessly transmit inputs from the controller to the laptop which would then send the data via TLS protocol to the R-pi instead of using a direct connection of the controller to the R-pi. This was done to allow both the keyboard of the laptop and the controller to control Alex as a form of redundancy in the event one failed to operate.

Section 4 Hardware Design

For the hardware design, our group mainly focused on the following criteria: stability and compactability of the Alex, and effectiveness of the sensors. The details of hardware placements are shown in the Figure 4.1 to 4.6.

In order to keep the Alex stable throughout the navigation, we need to arrange the components such that the centre of gravity of Alex is located at an optimum position to prevent tipping over during the hump challenge. Through testing, we realised we need to localise the centre of gravity at the centre and towards the front of the Alex. Therefore, we place most of the heavy components - RPLIDAR and the Motor batteries at the front.

For the compactability of the Alex, we realised the different components needed to be smoothly connected so that it would be easier for us to arrange the wire. Therefore, different components are placed close to their power source and components are adjusted based on the length of their connecting wires. Additionally, wires connecting to the same component are tied together and kept away from wires to prevent tangle.

To utilise all the sensors, we considered which positions are most suitable for them to detect data as well as consolidate each other.

- RPLIDAR is placed at the highest place of Alex to ensure there is no obstructions by the other components
- The Color sensor is placed at the front as the arrow in the visualisation tool RVIZ representing the Alex is pointing to the front. Therefore, it would be more intuitive for us to monitor and adjust the pointing of the colour sensor
- Ultrasonic sensor and buzzer (Additional hardwares)
  Ultrasonic sensor is placed directly above the colour sensor as
  - Our colour sensor works accurately at distances of 8-12 cm.
  - If the robot gets too close to an object (<10 cm), it will trigger the buzzer, indicating to us to get further away from the object, then scan for its colour.

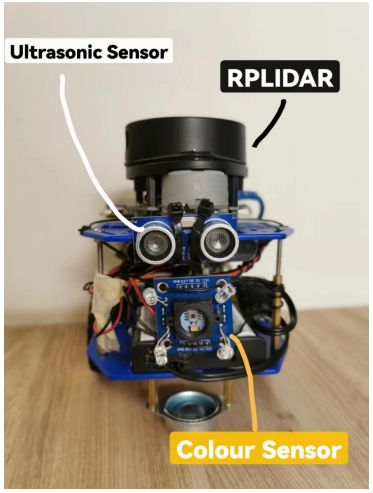


Fig 4.1: Front View of Alex

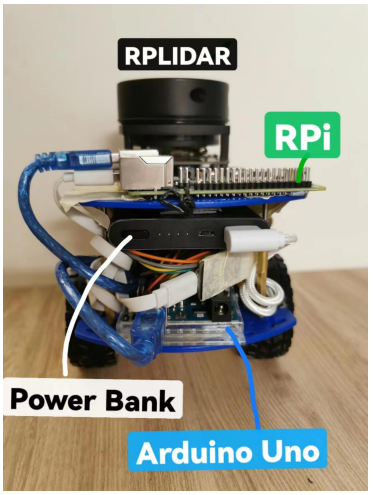


Fig 4.2: Back View of Alex

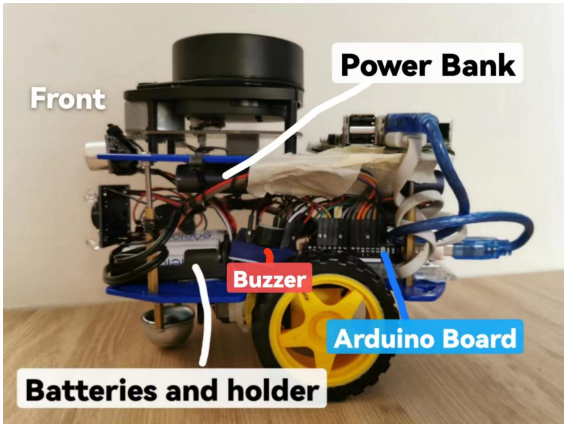


Fig 4.3: Left View of Alex

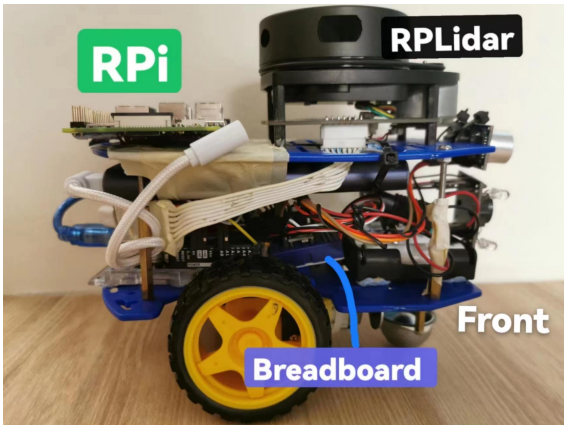


Fig 4.4: Right View of Alex

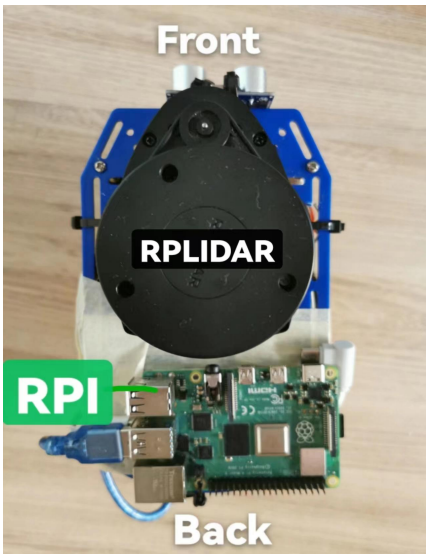


Fig 4.5: Top View of Alex

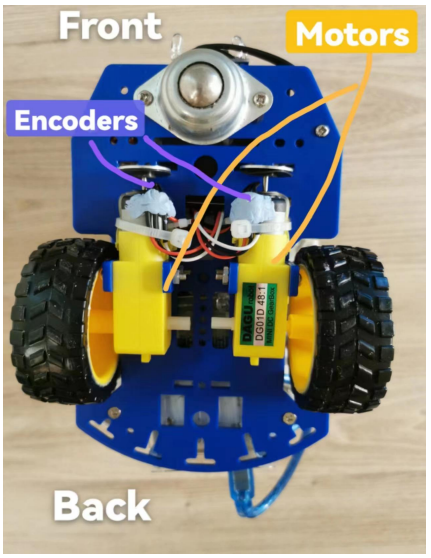


Fig 4.6: Bottom View of Alex

Section 5 Firmware Design

5.1 High-level Algorithm on the Arduino Uno

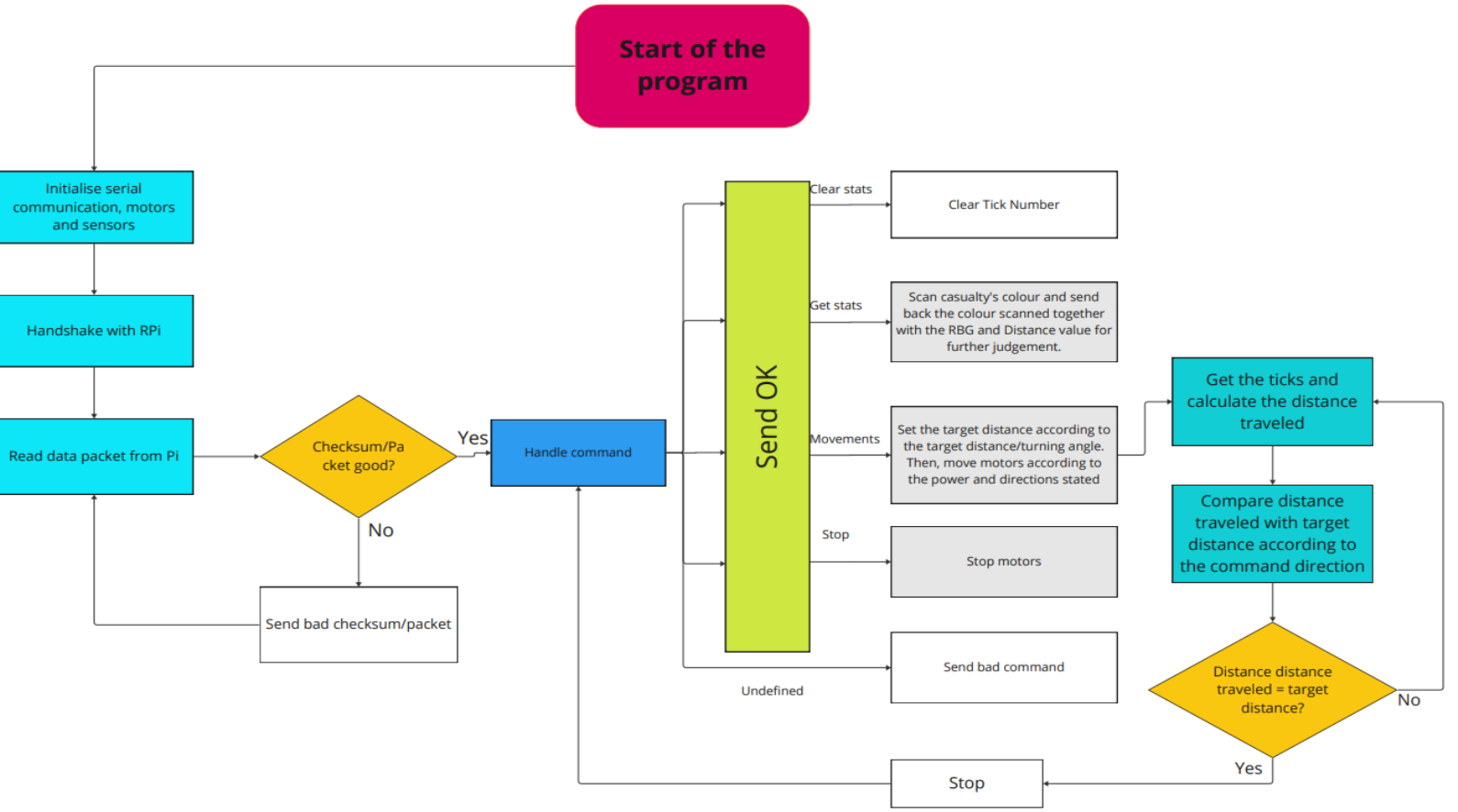


Fig 5.1: Arduino Uno algorithm design flow chart

Algorithm Outline

1. Initialise:
   When RPi is booted and powers the Arduino, Arduino would initialise its serial communication and the components control.
2. Receive and execution of user commands:
   Commands about movements, data detection, clear status or stop alex.
3. Repeat step 2 until evacuation finished

5.2 Communication Protocol

Firstly, the physical connection between Arduino Uno and RPi is through USB cable. We are using 9600 baud rate and 8N1 which stands for eight bits data length, no parity bit and one stop bit. These can be referred from the Alex.ino file.

We arrange the commands and data in a data structure to send out different messages and responses.

Messages and Responses are sent in Data Packets which involves:

- Packet type
  This is used to define which kind of data e.g. message, response or error that a data packet contains. Each packet type has its own ID as shown in the table below.

Packet Type	ID number
PACKET_TYPE_COMMAND	0
PACKET_TYPE_RESPONSE	1
PACKET_TYPE_ERROR	2
PACKET_TYPE_MESSAGE	3
PACKET_TYPE_HELLO	4

- Command
  This is the name of the commands which tells what exact the data is. For example, if the type is response then the command tells what response it is. If the type is message then the command tells what Alex should do.
- Dummy (at Arduino side)
  This is used for padding at the Arduino side as the architecture design of the RPi causes it to interpret 4 bytes at a time.
- Data Parameters
  These are the parameters that store the quantitative data information such as what the sensor detected from the Arduino side or the movement target distance and power input from the RPi side.

A packet containing a message or response is serialised and then sent between the Arduino and RPi. When it is received, at the start, a magic number check would be conducted to ensure it is a valid packet. Then, the checksum of the packet would be checked against its original checksum to ensure the data is not corrupted. If the checksum is ok, it is deserialised by the processor. There would be another check on whether the data received is complete and assemble together all the information from one data packet but arriving at a different time, and then execute the command accordingly. If the magic number or checksum is not identified correctly, an error response of a bad magic number or bad checksum would be sent back.

Section 6 Software Design

6.1 Teleoperation

Teleoperating Alex is split into 3 main parts: initialisation, environment mapping, and movement. Their respective high-level algorithms are outlined below.

1. Initialisation via TLS

1. Boot up R-Pi on Alex and start server together with handshake between Pi and Arduino
    - R-Pi sends 'Hello' to Arduino
    - Arduino replies with acknowledge data packet for the 'Hello'
    - Pi replies with acknowledge data packet for Arduino's reply
2. Connect operator's laptop (client) to the server using Transport Layer Security (TLS) protocol

2. Environment mapping

1. To generate a map of the environment, we first start a RPLIDAR node under Robotic Operating System (ROS) that gets data of the environment from the spinning RPLIDAR
2. We then run a Hector SLAM node that uses RPLIDAR data to create a map
3. SLAM identifies surrounding landmarks and Alex's location relative to them
4. Lastly, we run a visualisation node, *rviz*, that uses this environment data and Alex's location data to generate a real-time map of the environment
5. The map updates itself as Alex navigates the environment

3. Movement of Alex

1. To navigate Alex, the operator sends movement commands from the laptop to R-Pi
2. The operator enters an interface to control Alex using the keyboard.
   Below is a list of our movement commands, mapped to keypresses on the laptop:
    - Move forwards/backwards by a short or long distance
    - Turn left/right on the spot by a small or large angle
    - Move forwards/backwards faster to clear the hump
    - Stop moving completely

```
// WASD movement
case 119:    // w - forwards
   printf("Moving forwards\n");
   params[0]=3;
   params[1]=60;
   buffer[1] = 'b';
   memcpy(&buffer[2], params, sizeof(params));
   sendData(conn, buffer, sizeof(buffer));
   break;
```

Fig 6.1: Example of movement command on the laptop (client side)

3. Once the laptop has read a valid keypress, it sends a packet containing information on the movement command to R-Pi via the secure connection
4. R-Pi would then send another packet containing information on the movement command to the Arduino. The Arduino would acknowledge if the correct command is receive, or send an error message back to the R-Pi if the packet is invalid
5. After receiving a valid packet, Arduino will execute the appropriate movement
6. In terms of movement instruction:
    - Straight movement: wheels rotate at same speed in the same direction
    - Turning: wheels rotate at the same speed in different directions

   Depending on the movement, the relevant ticks are counted as the wheels rotate to track whether the robot has reached the target distance/angle
7. According to the generated map, the operator will navigate Alex to map the entire environment, find and identify "casualties", then move to the designated parking spot
8. All the movements of Alex were pre-calibrated before hand such that the "small angle" would be set to approximately 5°, the "large angle" would be set to approximately 45°, the "short distance" would be set to roughly 4cm and the "long distance" would be set to roughly 10cm.

6.2 Colour detection

1. The colour sensor and its white LEDs are turned on once Alex is turned on
2. Alex navigates to the optimal distance of 10-12 cm from the object for colour detection
3. The colour sensor scans the object and obtains its Red, Green, Blue (RGB) values
4. Our Arduino code determines the colour according to our predetermined thresholds
5. The colour and RGB values are sent in a packet from the Arduino to R-Pi, then R-Pi sends another packet to the laptop over the secure connection
6. The colour and RGB values are then displayed on the laptop, allowing us to discern the colour of the object to determine whether it is a "casualty"

```
// display colour according to threshold set in Arduino's code
if (data[10] == 0)
{
   printf("Colour: RED\n");
}
else if (data[10] == 1)
{
   printf("Colour: GREEN\n");
}
else
{
   printf("NOT VICTIM\n");
}

// display RGB values, in case colour scanning is not accurate so we can determine manually
printf("RGB: %d - %d - %d\n", data[12], data[14], data[13]);
```

Fig 6.2: Output printed on laptop to identify colour

6.3 Additional functionalities

We connected a PS4 controller by Bluetooth to our laptop. To teleoperate Alex, the controller emulates keypresses, which are read by the laptop and the corresponding commands are sent to R-Pi. This feature provided a more intuitive method of controlling the movements of Alex which, in theory, would allow us to pilot Alex more effectively.

1. Initiate controller using 3rd-party application (Steam) that emulates keypresses
2. Movement is carried out according to the algorithm as outlined above

Section 7 Lessons Learnt - Conclusion

7.1 Lessons learnt

7.1.1 Implementation of Debouncing on the R-Pi

One major problem we encountered nearing the final run was a “bad magic number” whenever buttons on the PS4(or keys on the keyboard) were pressed in quick succession. This issue would cause time to be lost as a restart of all the programmes had to be made to fix this issue. Furthermore, this problem made controlling Alex a balancing act between controlling Alex fast enough to get a good timing but not too fast that we trigger the magic number error. We believe that this error is caused by the packets overlapping and overwriting over each other, thus producing a bad packet and hence bad magic number. Looking back, one way to fix this issue would have been to implement a debouncing feature on the R-Pi side such that after an input is being sent, a set delay prevents any more inputs from being sent to prevent the UART ports to be “jammed”. This debouncing code would be similar to one of the lab lessons on debouncing just that the delay would be set to much longer to allow an input to be fully processed before another is sent.

7.1.2 Iterative Problem Solving

Over the course of our project, as we encountered errors many changes had to be made to our original plan for Alex. We learnt that many iterative steps had to be taken to optimise every aspect of Alex. And for every problem we solved, a new one would take its place. The difficulty of this project was being able to discern between bugs we could live with and bugs that would be detrimental to Alex. For example, when Alex could not clear the bump, we had to plan and rebuild a new iteration of Alex that had a more centralised centre of gravity. However, in doing this we made the wiring of the Rplidar more difficult to deal with. Thus, many trade offs were made in the process of creating a fully functional Alex.

7.2 Mistakes made

7.2.1 Alex's orientation

Due to the placement of the Lidar sensor on Alex, we decided to define the front of Alex as the side with the small ball bearing. As a result the hump was not easily cleared as Alex would shift from side to side while on the hump, leading to erratic behaviour when clearing the hump. At times, Alex would also tip over when attempting to clear the hump. These issues were overcome by implementing certain special controls for Alex to self right itself in the event it tips over. We also cleared the hump backwards to make it easier for Alex. This cost us extra time during our runs.

7.2.1 Compactness of Alex

After ensuring the efficacy and reliability of Alex's software, one thing we overlooked was the optimisation of Alex's hardware. More specifically, Alex's cable management. Alex had a USB cable sticking out of its back from out of the Arduino which, during the actual run, caused it to get stuck on a wall during a turn which lost us some time. This protruding cable was initially done intentionally to allow us to more easily plug and unplug Alex when we were working on him. While we were not able to fix this before the run, looking back what we could have done would either be to re-route the cable to shorten Alex.

References

Basali. (2022, June 15). *The How and Why of Using Drones for Search and Rescue Operations?* FlytNow. Retrieved March 25, 2023, from https://www.flytnow.com/blog/drones-for-search-rescue

RMUS F.I.D.O. RMUS. (n.d.). Retrieved March 24, 2023, from https://www.rmus.com/collections/police-fire-search-and-rescue-drones/products/rmus-fido