

**CG1112 Engineering Principle and Practice**

Semester 2 2020/2021

**“Alex to the Rescue”**

**Design Report**

**Team: B03-2B**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Student # | Sub-Team | Role |
| Ng Andre | 1 | Hardware | Hardware Lead |
| Lim Chang Quan Thaddeus | 2 | Software | Software Lead |
| Nigel Ng | 3 | Software | Software Member |
| Lim Shyun Yin | 4 | Hardware | Hardware Member |

**Table Of Contents**

[**Section 1 System Functionalities**](#_heading=h.poogggxaqdh8) **3**

[Overall aims of the project](#_heading=h.vr8xr8hc7u6g) 3

[Description for actions to realise aims](#_heading=h.bup37uoo3e6d) 3

[System functionalities](#_heading=h.smhfbmhonqm2) 3

[**Section 2 Review of State of the Art**](#_heading=h.tp4ucfn4sqwa) **4**

[1. ATLAS (Search and Rescue Bot)](#_heading=h.nmk4w6tf4cx) 4

[2. INACHUS (Search and Rescue Robot)](#_heading=h.j5spaadwl1lc) 4

[Learning from previous projects](#_heading=h.qwwwo7dh5qve) 4

[**Section 3 System Architecture**](#_heading=h.xtmi77cu17vr) **5**

[Hardware and Software Components](#_heading=h.jaejhr64a2vz) 5

[Master Control Program (MCP)](#_heading=h.o3v7mltfmdpm) 6

[**Section 4 Component Design**](#_heading=h.qm1zd0m2rxfo) **7**

[Hardware Implementation of Alex](#_heading=h.ff76u5ir3g6l) 7

[High-level Steps](#_heading=h.xdu4ejvgb24u) 8

[Further Breakdown of Steps](#_heading=h.owdot6ofgcpa) 8

Algorithm Pseudocode 9

[**Section 5: Firmware Design**](#_heading=h.za5mcnh8hl2p) **10**

[Communication Protocols between RPi and Arduino Uno](#_heading=h.4vyukbh8e8rx) 10

[Packet Types](#_heading=h.5dee8r6ly44q) 10

RPi and Arduino Communication 11

[**Section 6: Software Design**](#_heading=h.qre8i6694fg6) **13**

[Movement and User Input](#_heading=h.u0lzluyfdwyy) 13

[**Section 7 Lessons Learnt**](#_heading=h.astmvwunbteo) **14**

**References 15**

# 

# Section 1 System Functionalities

## Overall aims of the project

Due to unpredictable occurrences of disasters all around the world, it is important for rescue workers to race against the clock to find and save survivors amidst the rubble. To reduce further injuries incurred during rescue operations, search and rescue robots are widely used. Thus, there is a need to research and improve on current search and rescue robots to increase the number of lives saved in a disaster.

## Description for actions to realise aims

This report aims to document the functions of Alex, a tele-operated robot with search and rescue functionalities. Alex will be able to traverse unfamiliar terrain and map back its surroundings to its operator at the same time through the Light Detection and Ranging (LiDAR) function. It will also be able to detect different coloured objects using a colour sensor. All this will be done remotely, with traversal handled by the operator. Alex will be assembled, tested and calibrated under simulated conditions on its ability to carry out its mission successfully.

## System functionalities

Alex can perform basic functions such as travelling forward and backwards, or turning left and right. The surrounding environment is also mapped out by LiDAR, thus generating a map that aids Alex’s movement. Alex will be able to detect obstacles using its Infrared Sensor (HW-201 [1]) and come to a stop. It also has a colour sensor (TCS230 or TCS3200 RGB [2]) to read the colour of objects and return the colour to the main system.

# Section 2 Review of State of the Art

This section highlights two state of the art search and rescue robots, namely ATLAS and INACHUS, and also describes their system functionalities, its hardware and software components, as well as its strengths and weaknesses.

## 1. ATLAS (Search and Rescue Bot)

Pioneered by Boston Dynamics, ATLAS was created to eventually participate in search and rescue operations.

It is an electrically powered humanoid robot that can operate both indoors and outdoors [3]. It has self-balancing capability [3] and can sense obstacles through the use of sensors, stereo cameras and LiDAR technology [4]. It can be operated via tele-operation as well as traverse difficult terrain and complete simple tasks like opening a door or climbing up a ladder [3]. This is achieved through the use of hydraulic joints [3].

ATLAS’s strengths lay in the fact that it is humanoid and thus can potentially perform more complex tasks than a non-humanoid robot. It is also able to recover to some extent should it fall or meet an obstacle [4]. However, its large size makes it cumbersome and thus not suited to small spaces where victims may be trapped.

## 2. INACHUS (Search and Rescue Robot)

Developed by SINTEF, INACHUS’s primary purpose is to assist Urban Search and Rescue (USaR) teams in finding victims in disaster zones [5].

It is shaped like a train or snake and is remotely controlled from a rugged tablet PC. INACHUS has two video cameras with lights attached for rescuers to get a clearer view of rubble that they cannot reach [5]. It also has an infrared camera to detect survivors in the rubble and an electronic chemical nose to help detect survivors through the composition of gases in the air [5]. Two-way communication is also enabled so that survivors can communicate with rescuers [5]

The strength of INACHUS is that it has many functionalities that enable rescuers many options in finding victims and providing immediate assistance to them. However, the slightly large size may mean it cannot reach very tight spaces. Furthermore, the need for power cables limits the range of INACHUS’s operation.

## Learning from previous projects

In both examples, the large size of the bot posed resulted in difficulty in reaching into small spaces. To overcome this, Alex is designed to be a compact robot with its various components attached to 2 thin chassis boards and stacked upwards, reducing width space and increasing Alex’s ability to reach into smaller spaces. Furthermore, to reduce the reliance on cables to power the robot as in INACHUS, the RPi is used to allow for remote control. A WiFi-Antenna is connected to the RPi, allowing Alex to be controlled remotely at great distances away from the operator.

# Section 3 System Architecture

## Hardware and Software Components

Figure 1 illustrates the interworking of Alex, as well as the functionalities of the respective hardware and software components.

The yellow boxes represent hardware, while the blue boxes represent software. Additionally, the connections between boxes in the diagram represent association of data between the subsystems of Alex.

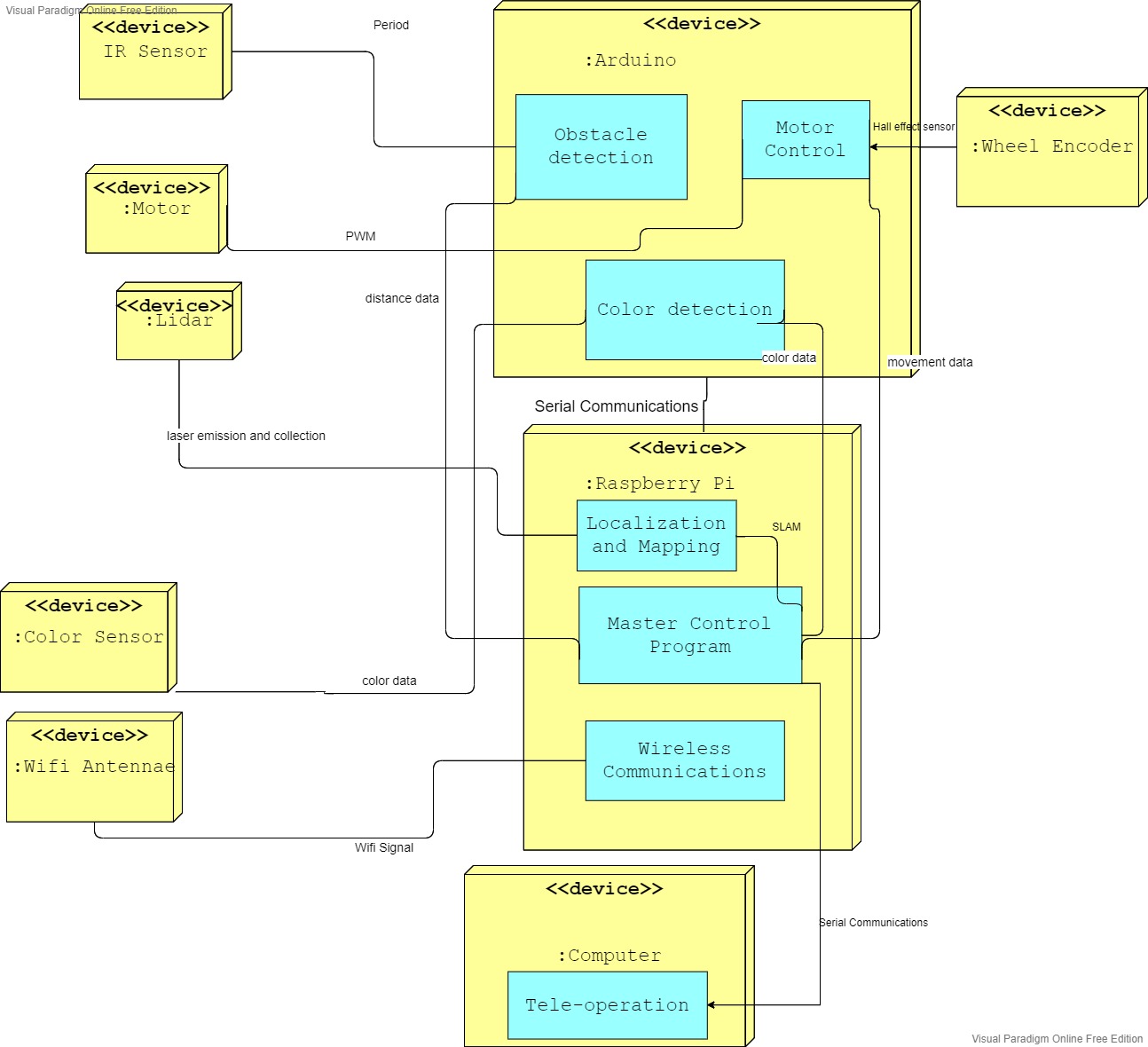
****

Fig 1. Interworking of Alex

## 

## Master Control Program (MCP)

The MCP on the RPi refers to the main code running on the RPi [6]. The RPi operating system allows the user to input commands via the terminal application [7]. By using the terminal on the RPi, the user can issue commands to the various subsystems on Alex. With the code ensuring duplex communication [8] between Alex and the user, Alex runs smoothly throughout navigation. This translates to piloting Alex remotely, as well as simultaneous mapping of the environment during Alex’s movement around the maze.

The infrared sensor is used for detecting obstacles that may hinder the movement of Alex. This is done by sending infrared pulses between the sensor and a nearby object at a regular interval [9], and receiving the distance between the object and the sensor. This data is then passed into the obstacle detection software on the Arduino for the user to avoid collisions.

The power supply of Alex consists of the power bank and battery pack. Together, they supply a 5V DC input for the Arduino and RPi. To reduce the power consumption of Alex, any unused modules in the Arduino will be disabled via the registers. Also, HDMI output and Bluetooth will be disabled on the RPi since only WiFi is required to tele-operate Alex via Secure Shell (SSH) remote access. Lastly, the refresh rate of the LiDAR will also be reduced as much as possible without affecting system functionality, to reduce the processing power required and thus reduce power consumption.

The Motor drives Alex’s Wheels. The motor is controlled by motor control software in the Arduino, which sends Pulse Width Modulation (PWM) signals to the motor [10], which in turn drives Alex’s Wheels.

Similarly, the Wheel Encoder records the distance travelled by Alex per revolution. By measuring the rotation of the ring magnet around the hall effect sensors on the Wheel Encoder [11], the rotation is then converted into data to be processed by the Arduino, which maps the rotations to the distance travelled by each wheel on the Arduino. This is useful for calibrating the speed of Alex’s movement, to prevent sudden jerks by Alex upon user input.

The LiDAR maps out the area around Alex through the transmission and receiving of laser scans. By doing so, data is then sent to the RPi which works with the LiDAR for simultaneous localization and mapping of the area around Alex [12]. In other words, the surrounding area around Alex would be shown graphically as a map to the user. This data is simultaneously shared with the user via the MCP, where the user would then use the map to navigate the simulated environment manually, using the Hector SLAM algorithm.

The colour sensor reads off the colour of an object when called on by the MCP on the RPi. More specifically, the user would send an “identify object” command to the MCP when such obstructions are detected during navigation. The colour of the object is returned as data to the MCP, which then relays the data to the user for recording purposes.

The WiFi Antenna is designed for strengthening the WiFi signal received by the RPi. In doing so, the effective range of Alex is increased. This is especially important since the user would be tele-operating Alex from their laptop at a distance away from the obstacle course.

The user’s computer/laptop is necessary for duplex communications between the user and the MCP on the RPi. From the user’s end, input on the terminal of RPi would be translated by the MCP to move Alex. Concurrently, the MCP would grant the user a simulated environment, which includes walls as well as obstructions. This is represented through the environment map, which would visually indicate when Alex is about to collide with an object, or when the area around Alex is clear.

# Section 4 Component Design

This section will focus on the working principle and hardware of Alex and how it will attempt to complete its objective by having the operator follow an overall algorithm.

The hardware components of Alex are presented in Figure 2, and the overall algorithm is demonstrated in Figure 3.

## Hardware Implementation of Alex

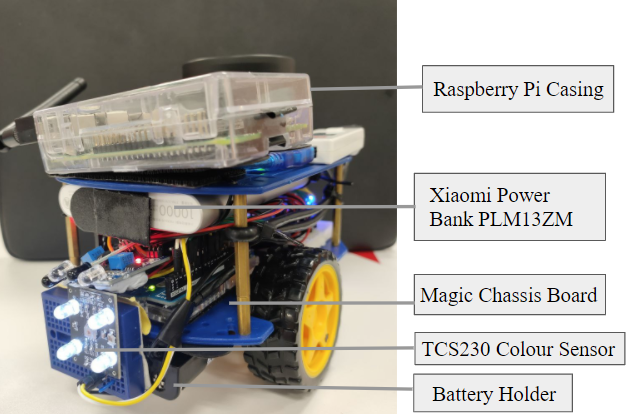
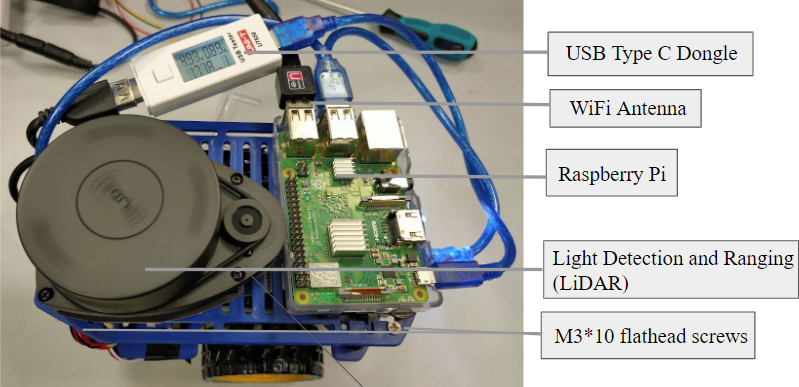
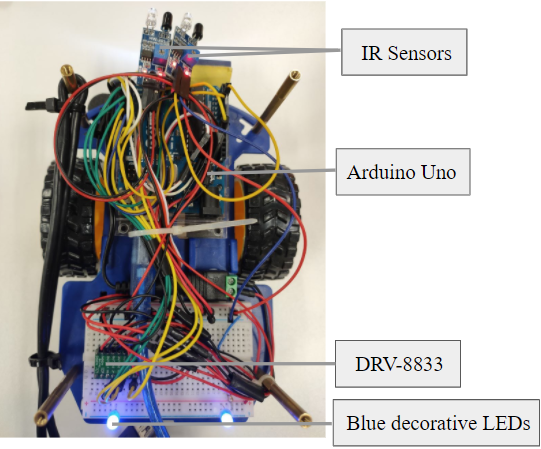
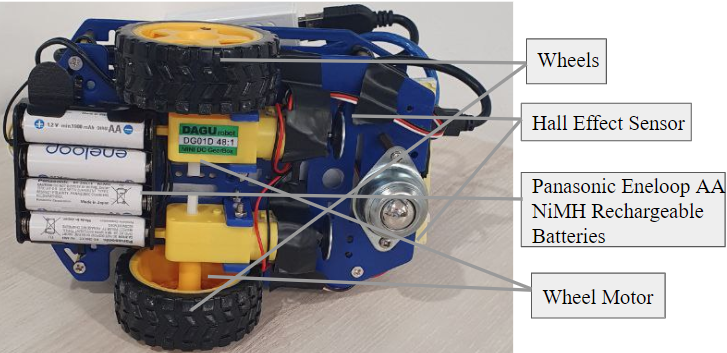
Fig 2.1. Alex’s front and side view Fig 2.2. Alex’s Top Layer

Fig 2.3. Alex’s Middle Layer Fig 2.4. Alex’s Bottom Layer

## 

## High-level Steps:

1. Initialisation and Calibration

2. Identify Object Colour

3. Receive and Execute Movement Command

4. Repeat until the mapping is complete

## 

## Further Breakdown of Steps

1. Initialisation and Calibration

During initialisation, Alex is turned on, and the LiDAR starts to map out its surrounding environment. This information is then relayed back to the operator. The operator can then begin controlling Alex by issuing commands to move forward, backward, turn left or turn right.

2. Identify Object Colour

If an object is detected, Alex will identify the object’s colour using its colour sensor, at the step Identifying Object Colour from Figure 3. The colour sensor module on Alex works by emitting a bright white light from its LEDs. The module will then read in the red, green and blue values of the object, then determine its colour - which is then relayed back to the operator.

3. Receive and Execute Movement Command

Next, with the environmental data received, the operator can then make a judgement call on which direction to move. The operator will then issue a command that will be transmitted serially to the RPi on Alex. This information will then be passed on to the Arduino, which subsequently sends the relevant information to the motor.

4. Repeat until the mapping is complete

As depicted in Figure 3, Alex will attempt to map out its surrounding environment using LiDAR, while the entire maze is not mapped out. Concurrently, Alex will continue to receive movement commands and move accordingly.

On the operator’s end, they would need to:

(1) Look out for objects that need to be identified and move towards them;

(2) Avoid collisions with obstacles;

(3) Move Alex around the area;

The process is repeated until each object has been successfully identified and the entire disaster area has been identified.

Additionally, given the project requirements the operator would then have to park Alex after identifying the casualties and mapping out the area.

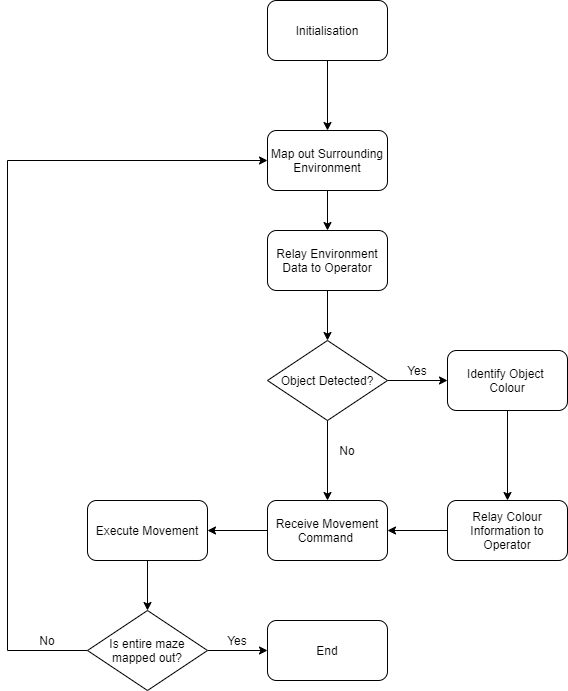


Fig 3. Algorithm pseudocode of Alex

# Section 5 Firmware Design

## Communication Protocols between RPi and Arduino Uno

The RPi and the Arduino Uno are connected through a physical connection. The communication’s bit-level protocol is set to 9600 8N1, which has a baud rate of 9600, a data length of 8 bytes, no parity bits and 1 stop bit.

## Packet Types

Data is sent in the form of packets and each packet’s structure is listed in Figure 4.1 below:

|  |  |
| --- | --- |
| **Packet Variable** | **Description** |
| 1. **packetType** | * A char data type that describes what type of information the packet contains * see Figure 4.2 on different packet types |
| 2. **command** | * A char data type that contains the specific command |
| 3. **dummy[2]** | * A char array data type of size 2 acts as padding so that the packet achieves natural word alignment |
| 4. **data[MAX\_STR\_LEN]** | * A char array data type of size MAX\_STR\_LEN. MAX\_STR\_LEN describes the maximum length of the string, which is defined to be 32. This char array contains the string data to be sent |
| 5. **params[16]** | * A size 16 array of type unsigned 32bit integers. It contains the parameters distance to travel/ angle to turn and the speed at which to do so. |

Figure 4.1. Description for various types of packet variable

The Packets are differentiated by their type and command given. Each type and command is assigned a value such as the one in the table shown below. This allows the Arduino and RPi to know what action it should execute.

|  |  |
| --- | --- |
| Packet Type | Assigned value |
| PACKET\_TYPE\_COMMAND | 0 |
| PACKET\_TYPE\_RESPONSE | 1 |
| PACKET\_TYPE\_ERROR | 2 |
| PACKET\_TYPE\_MESSAGE | 3 |
| PACKET\_TYPE\_HELLO | 4 |

Figure 4.2. Table about the different Packet Types

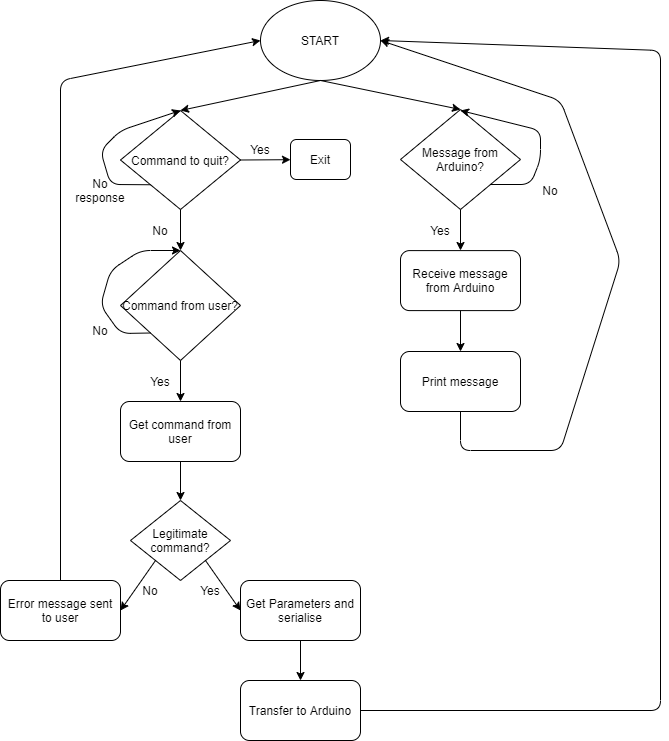


Figure 5. RPi communication flowchart

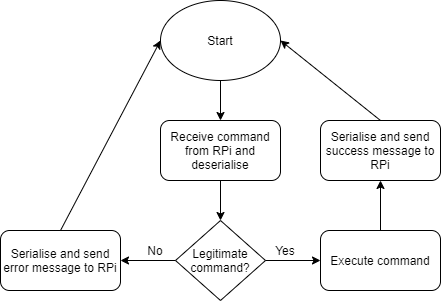


Figure 6. Arduino communication flowchart

Once on, both the RPi and the Arduino will start their serial communication, in accordance with its predetermined bit-level protocol. The RPi will continuously wait to get commands from the user until it is told to stop. At that point, the RPi will close its connection to the Arduino.

Once the RPi gets a command from the user, the packet type is declared to be PACKET\_TYPE\_COMMAND. If the command is illegitimate, a message will be sent to the user to inform them of the error. If the command is legitimate, parameters(distance/angle and speed) are requested and obtained from the user. The packet’s command type is then specified according to the user’s input. The packet is then serialised and sent to the Arduino.

Once the Arduino receives the serialised data, it deserialises it. If the command is legitimate, it will execute the instructions within it, else it will send an error message back to the RPi. If the command is executed successfully, Arduino will also send a message to RPi to print out for the user to see.

When the RPi receives a response from the Arduino, depending on the packet type, it will print out different information, such as statistics or error messages.

# Section 6 Software Design

Displaying environment data through Robot Operating System(ROS)

## 

This section focuses on how Alex accepts user input for movement and colour sensing. In particular, the ROS architecture of the LiDAR would be discussed in detail. Additionally, we would explore how Alex displays the environment data gathered by the LiDAR. For Alex to be tele-operated and display environment data simultaneously, two computers are needed - we would term these computers (1) Movement and (2) Display. The Movement computer would be used to send input to trigger movements of Alex as well as to instantiate the RPlidarDriver instance on the Pi, while the Display computer runs roscore and displays environment data to the user.

Next, the user would instantiate an RPlidarDriver instance via the software development kit (SDK) feature . Effectively, the RPlidarDriver would retrieve environment data in the form of scan data output regularly once instantiated. This is done through the (1) Movement computer after the roscore command is activated on the (2) Display computer, to allow for data transfer from the rplidarDriver node on (1) Movement computer to the roscore server running on (2) Display computer. The data transfer between nodes is elaborated on later, and can be referred to as the /rosout topic.

The following step requires the user to run hector SLAM on the (2) Display computer. Hector SLAM contains ROS packages related to performing simultaneous localization and mapping in unstructured environments, especially for Urban Search and Rescue operations. [13]

One of the approaches to SLAM is the hector mapping, which does not require the use of odometry data and instead relies on the high update rate of LiDAR. Once hector SLAM is instantiated, a map representing the terrain around Alex would be displayed on a graphical user interface (GUI). Similar to the mapping and lidar nodes, a GUI node is activated for this functionality.

Additionally, the GUI node would take in the scan data from the RPlidarDriver over the roscore service, and translate the data into a visual map to the user, where objects that may restrict Alex’s movement such as walls and obstacles are displayed to the user. Moreover, additional functionalities such as displaying the current scan-data of the LiDAR as well as highlighting Alex’s current position are granted to the operator via this node.

Simultaneously, Alex’s path of movement and the map are updated and shown through the GUI node.

Movement commands for Alex would be sent through the (1) Movement computer, with direction of movement, distance/angle and power entered through user input. The parameters (angle/distance) would differ if Alex is moving forward/ backwards, or making a turn (left/ right). The movement of Alex would be reflected visually on the (2) Display computer for the operator’s reference.

Overall, the ROS nodes used by Alex which generate the map, track the trajectory of Alex as well as display the map graphically are tracked via the roscore service, and viewed on the

(2) Display computer. This reduces the need for processing power on Alex, as displaying the map generated by SLAM consumes a lot of power and is thus more efficient when run on the display computer. Importantly, all the outputs of the nodes are connected to a singular /rosout topic, mentioned previously. The /rosout topic is also a package from the LiDAR SDK, and is a system-wide logging mechanism for all the outputs of the nodes [14].

# 

# Section 7 Lessons Learnt

Through our experience building Alex, we learnt many things.

First, we learnt to be more meticulous in obtaining parts. For example, we thought any type of colour sensor would do but we later realised that the colour sensor we bought had a very limited accurate range. This led to us having to come up with ways to better improve the accuracy of the colour sensor at larger distances. If we had been more meticulous, we could have just bought a colour sensor with greater range and we would not have faced the aforementioned problem.

Secondly, we learnt that designing is an iterative process. When building Alex, we faced Murphy's law on a practical level - for instance, we wired the power bank to the first level when arranging Alex initially. However, we realized that with additional weight added onto Alex (LiDAR, breadboards etc.), the wheels would be in constant contact with the sides of the power bank. Thus, we use cable-ties to secure the power bank on the bottom of the top layer instead.

Lastly, we learnt that form follows function. An important principle in design - we had to continuously rewire the electrical connections of Alex as well as remove unnecessary components that would increase the power consumption of Alex and thus reduce the performance of Alex in it’s intended mission of mapping and obstacle detection. For example, a member of our team demonstrated a great level of aptitude in hardware design and suggested adding blue decorative LEDs to the rear of Alex, similar to reverse lights of an automobile. To the optimist in an actual casualty rescue scenario, this would represent the “light at the end of the tunnel”, as well as alert casualties to the presence of a search and rescue robot. However, after much consideration and advice from the teaching team we prioritised the functionality of Alex to improve its overall performance and removed the decorative lighting. After all, a fully-functional robot is a beauty itself.

**References**

[1] “Infrared Sensor Module,” *Cytron Technologies Singapore*. [Online]. Available: https://sg.cytron.io/p-infrared-sensor-module?r=1&gclid=CjwKCAjw07qDBhBxEiwA6pPbHqchUpF7\_Uar5eW6IngvqoyAEPVv70uInYuUKfwH8\_n3TbohE2HsnhoCaHAQAvD\_BwE. [Accessed: 08-Apr-2021].

[2] “Colour Sensor TCS230/TCS3200,” *Home – Smart Engineering Solutions*. [Online]. Available: https://www.continental.sg/colour-sensor-tcs230-tcs3200.html. [Accessed: 08-Apr-2021].

[3] *wevolver.com*. [Online]. Available: https://www.wevolver.com/wevolver.staff/atlas.robot#:~:text=Atlas%20is%20designed%20to%20operate,electrically%20powered%20and%20hydraulically%20actuated. [Accessed: 02-Mar-2021].

[4] E. Guizzo, “How Boston Dynamics Is Redefining Robot Agility” *IEEE Spectrum: Technology, Engineering, and Science News*, 27-Nov-2019. [Online]. Available: https://spectrum.ieee.org/robotics/humanoids/how-boston-dynamics-is-redefining-robot-agility. [Accessed: 02-Mar-2021].

[5] G. Marafioti, page, and Name, “Snake-like robots assisting Urban Search and Rescue teams (INACHUS project),” *#SINTEFblog*, 11-Oct-2018. [Online]. Available: https://blog.sintef.com/digital-en/inachus-project-robot-search-rescue/. [Accessed: 02-Mar-2021].

[6] M. Long, *What is Raspberry Pi OS?*, 18-Jun-2020. [Online]. Available: https://www.electromaker.io/blog/article/what-is-raspberry-pi-os#:~:text=Raspberry%20Pi%20OS%20is%20a,also%20run%20Raspberry%20Pi%20OS. [Accessed: 07-Mar-2021].

[7] “projects.raspberrypi.org,” *Using your Raspberry Pi*. [Online]. Available: https://projects.raspberrypi.org/en/projects/raspberry-pi-using/8#:~:text=The%20terminal%20is%20a%20really,of%20clicking%20on%20menu%20options. [Accessed: 07-Mar-2021].

[8] “Transmission Modes in Computer Networks (Simplex, Half-Duplex and Full-Duplex),” *GeeksforGeeks*, 19-Feb-2021. [Online]. Available: https://www.geeksforgeeks.org/transmission-modes-computer-networks/. [Accessed: 07-Mar-2021].

[9] L. Reese, S. Says, Sravya, A. K. says, A. Kalnoskas, H. says, Hari, E. says, and Eny, “The working principle, applications and limitations of ultrasonic sensors,” Microcontroller Tips. [Online]. Available: https://www.microcontrollertips.com/principle-applications-limitations-ultrasonic-sensors-faq/. [Accessed: 07-Mar-2021].

[10] J. Heath, “PWM: Pulse Width Modulation: What is it and how does it work?,” *Analog IC Tips*, 06-Nov-2018. [Online]. Available: https://www.analogictips.com/pulse-width-modulation-pwm/. [Accessed: 07-Mar-2021].

[11] “Home - Last Minute Engineers,” *How Rotary Encoder Works and Interface It with Arduino*. [Online]. Available: https://lastminuteengineers.com/rotary-encoder-arduino-tutorial/. [Accessed: 07-Mar-2021].

[12] “What Is SLAM (Simultaneous Localization and Mapping) – MATLAB & Simulink,” *What Is SLAM (Simultaneous Localization and Mapping) – MATLAB & Simulink - MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/discovery/slam.html. [Accessed: 07-Mar-2021].

[13] C. Rose and M. Oehler, “Hector SLAM,” *Team Hector*. [Online]. Available: https://www.teamhector.de/resources/32-hector-slam2. [Accessed: 14-Apr-2021].

[14] J. Faust, *rosout*. [Online]. Available: http://library.isr.ist.utl.pt/docs/roswiki/rosout.html. [Accessed: 14-Apr-2021].