CG1112 Engineering Principle and Practice
Semester 2 AY2020/2021

"Alex to the Rescue"
Final Report
Team: B01-5A

Name	Student #	Sub-Team	Role
Teoh Yi Zheng	1	Software	Design Software Implementation
Tay Jing Ying Kimberley	2	Software	Code Hardware Functions
Fan Shixi	3	Hardware and Software	Code and Oversee Hardware assembly
Christopher Nge Jing Qi	4	Hardware and Software	Code and Oversee Hardware assembly

Table of Contents

Section 1 Introduction **3**
1.1 Problem statement 3
1.2 Features of Alex 3

Section 2 Review of State of the Art **4**

Section 3 System Architecture **5**
3.1 Details of the system 5
3.2 Workflow Description 5

Section 4 Hardware Design **6**
4.1 Overall Alex design 6
4.2 Obstacle detection: HC-SR04 - Ultrasonic Distance Sensor 6
4.3 Path Straightening - FC-51 IR Sensor 7
4.4 Colour Detection - TCS3200 Sensor Module 7

Section 5 Firmware Design **8**
5.1 Higher-Order Steps of Overall Algorithm 8
5.2.1 LiDAR Scan 8
5.2.2 Motor Control 9
5.2.3 IR Sensor & Ultrasonic Sensor 9
5.2.4 Colour Sensor 10

Section 6 Software Design **12**
6.1 Teleoperation between Laptop and RPi 12
6.2 Graphical User Interface (GUI) 13
6.3 Power Saving 13

Section 7 Lessons Learnt - Conclusion **14**
7.1 Important Lessons 14
7.2 Greatest Mistakes 14

References **15**

Section 1 Introduction

1.1 Problem statement

Since the 1970s the number of cataclysmic events has increased by four times and over 100 million people are affected by natural disasters every year. With many unpredictable disasters happening around the world, search and rescue robots(SAR) are an important focus to extract survivors from the disaster. These robots provide benefits to search and rescue missions, being able to fit into places where people are not able to and operate in areas that in a typical disaster setting, where the terrain has many debris and obstacles that make it difficult for these robots to navigate and access. Furthermore, in an unknown environment, it is necessary for the robot to map out the surroundings for it to know where to travel and provide useful information for rescuers. Currently, there are numerous search and rescue robots being deployed for different terrain types such as the Boston Dynamic Robot Spot. This 4-legged robot is able to navigate in tumultuous environments that are otherwise too dangerous for humans to travel, this allows it to have the potential to be deployed in a Search and Rescue Mission in terrains that are otherwise too dangerous for rescuers. It was recently tested on an abandoned mine [1] where its ability to map and navigate the dangerous environment was truly outlined by the team controlling the robot. Our solution builds on the current technology to create a search and rescue robot, Alex, that is likewise, remotely controlled to survey a terrain.

1.2 Features of Alex

The robot should be **remotely and wirelessly operable** by a human operator. A graphical user interface (GUI), as mentioned in Section 6.2, is available to make the search and rescue operation more intuitive for the operator. The robot's movements will be **manually and remotely controlled** by the Raspberry Pi (RPi) connected to the Arduino Uno. The robot can create an **accurate scaled map** of the environment using the measurements from the 2D LiDAR. The robot will be able to **track and identify its real-time position** on the map. Diagram 2 illustrates the path taken by the robot as it moves about within the mapped 2D environment. The robot will have sensors to alert the operator when it is in close proximity to an obstacle for **obstacle detection**. The robot should be able to accurately **detect the colour** of objects that are scattered around the environment.

Section 2 Review of State of the Art

This section focuses on the evaluation of two available tele-operable search and rescue robots. This evaluation will help us to better gauge the required specifications and capabilities of the Alex robot we are developing.

	Platform 1 **Taurob Tracker [**2]	Platform 2 **Boston Dynamics Robot Spot** [1]
Description	A teleoperated robot that aids search and rescue operations in dangerous situations by exploring and mapping environments to detect humans and provide rescuers a clear picture of the safest way to gain access to people trapped.	Semi-autonomous teleoperated search and rescue robot, that is capable of working together with an operator as a team, solving problems such as exploring its surroundings and performing object detection.
Functions	• Path planning • Obstacle detection • Human detection • Area mapping	• Navigating complex terrain • Obstacle detection • Human detection
Hardware	• Track vehicle platform • robotic arm • Front and rear 3-D camera • GPS • front and rear CMOS camera • Front, rear-facing 2-D LiDARs • Elevated 3-D LiDAR • Intel i7 2.90GHz desktop computer • External wirelessly connected laptop • External controller board with joystick	• 4-legged robot • 360-degree camera • Intel i5 3.10GHz desktop computer • Tablet controlled
Software	• Robot Operating system on ubuntu	• Robot Operating system on ubuntu
Strengths	• Easily controlled due to joystick • High maneuverability through terrain due to tracks • Easy for the operator to identify surroundings due to camera array • Accurate environment mapping due to 3-D Lidar	• Easily controlled with a proprietary controller • High maneuverability through terrain due to legs • Easy for the operator to identify surroundings due to camera • Remotely controlled
Weaknesses	• Complex to develop • Short operation period due to desktop-grade computer and a significant number of cameras and sensors • Expensive due to the number of components	• Complex control due to robot balance and leg movement • Limited battery life due to use of legs instead of wheels for movement resolution 3-d camera • Expensive due to incorporation of complex components

Table 2.1: The two most commonly used SAR Robots

Observing the implementation of the current technology we have decided on dependence on laptop running ubuntu to utilise a more powerful processor to process complex computations such as hector_slam algorithm. Similar to the Taurob Tracker, Alex will utilise a LiDAR for mapping the surrounding area. Instead of utilising a 3-D LiDAR or 360 degree camera used by the two robots, Alex uses an ultrasonic and an array of infrared sensors for obstacle identification. Thus, allowing it to be effective and significantly cheaper than the Taurob Tracker and Boston Dynamic Robot Spot.

Section 3 System Architecture

This section focuses on the System Architecture of Alex. Diagram below illustrates the different hardware and software components in the project.

3.1 Details of the system

1. The two big blue and red boxes are the two main micro-controllers that will be used to communicate with the other hardware, which are the smaller yellow boxes.
2. The big green box is the master control program (MCP) that controls the commands sent by the Raspberry Pi (RPi).
3. The small rectangles within each micro-controller are the software instructions loaded within the two respectively micro-controllers.
4. Arduino interacts with the RGB Sensor, IR Sensors and wheel encoders through digital input and LiDAR interfaces directly with the RPi via serial communication. The Arduino will be interfaced with the RPi, which is connected to the Ubuntu laptop via a SSH secure channel.

3.2 Workflow Description

To begin, RPi will send a trigger to the LiDAR to scan the environment around the robot. After scanning, it will send back the serial data of the surroundings to the laptop which creates a 2D map of the environment and generates the GUI.

After receiving, manual instructions to move Alex would be sent from the MCP through RPi. Once received, the Arduino begins to send the software instructions, such as sending a PWM signal to the motor to move the robot. It will also trigger the IR Sensors for obstacle detection. If it detects a nearby wall, it will send back data to the Arduino where it will process it and notify us whether a collision is imminent. When the robot reaches a checkpoint for colour detection, a trigger will be sent from the Arduino to the RGB Sensor to begin scanning. Afterward, the RGB sensor will return the analog RGB Data back to the Arduino where it would then process the data and identify the colour of the surface detected and send it back to the RPi.

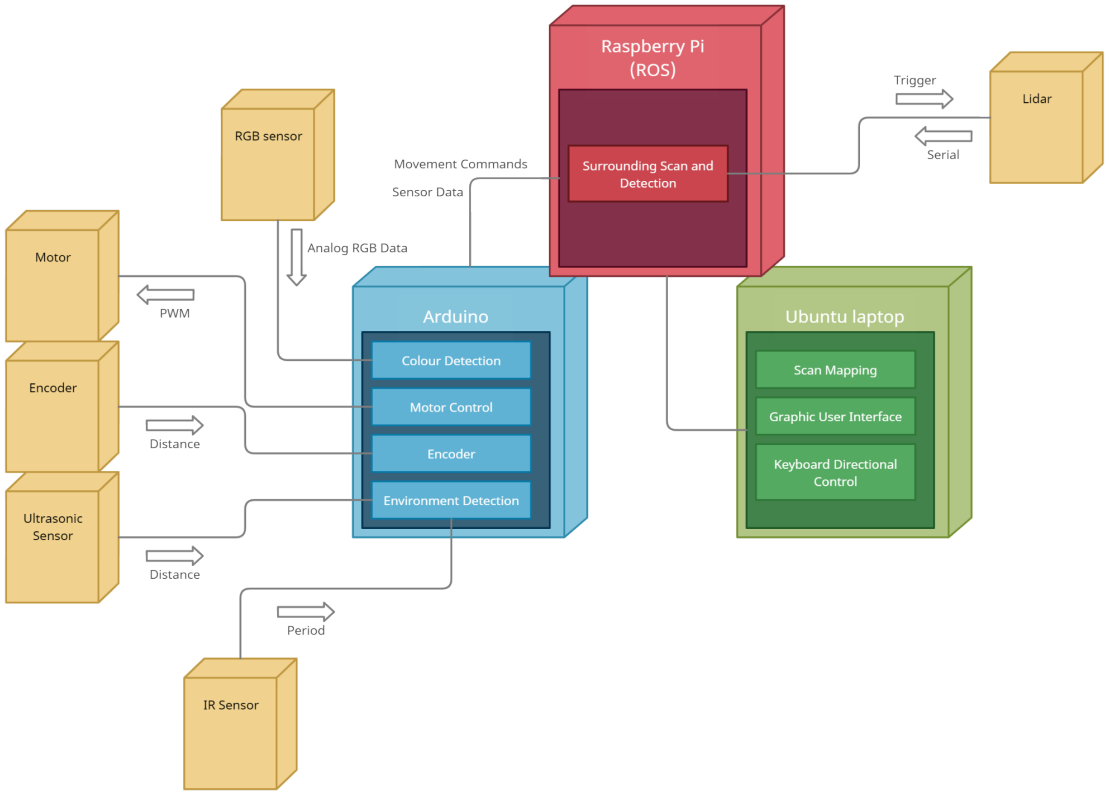

Diagram 1: UML Diagram for the System Architecture of Alex

Section 4 Hardware Design

4.1 Overall Alex design

The design of Alex consists of various subsystems and components that work together to implement the features necessary for the search and rescue mission (Fig. 4.1). These systems will be explained in detail in the sections below.

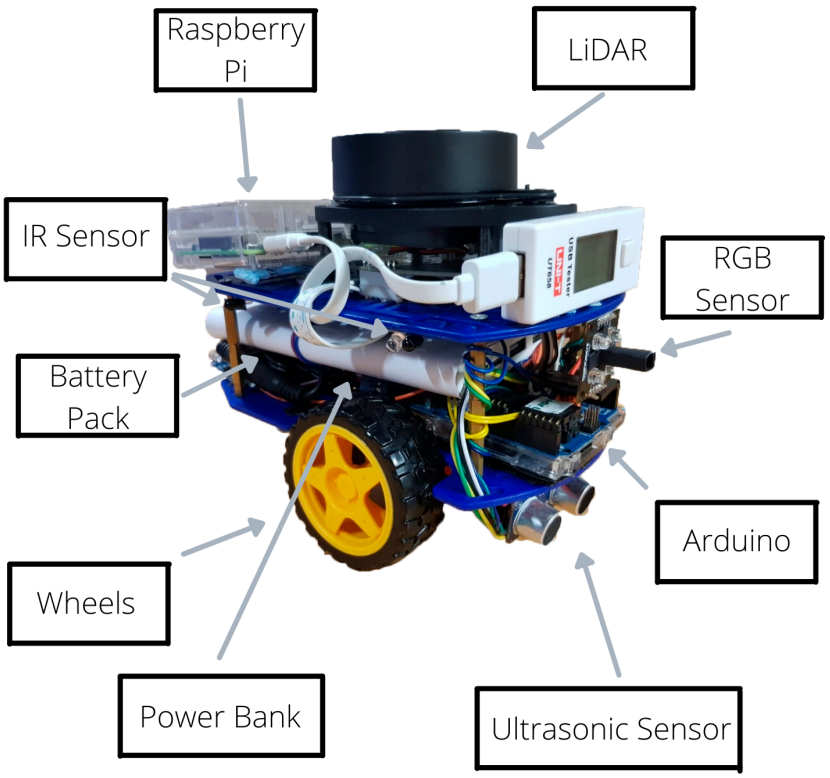

Fig. 4.1: Final form of Alex with all hardware components labelled

4.2 Obstacle detection: HC-SR04 - Ultrasonic Distance Sensor

To ensure that Alex does not collide with the surrounding, the team decided to install an ultrasonic sensor at the front of Alex (From Fig. 4.2). The ultrasonic sensor is programmed to return the distance between Alex and the closest obstacle in front of it, this allows Alex to be notified when an obstacle is too close.

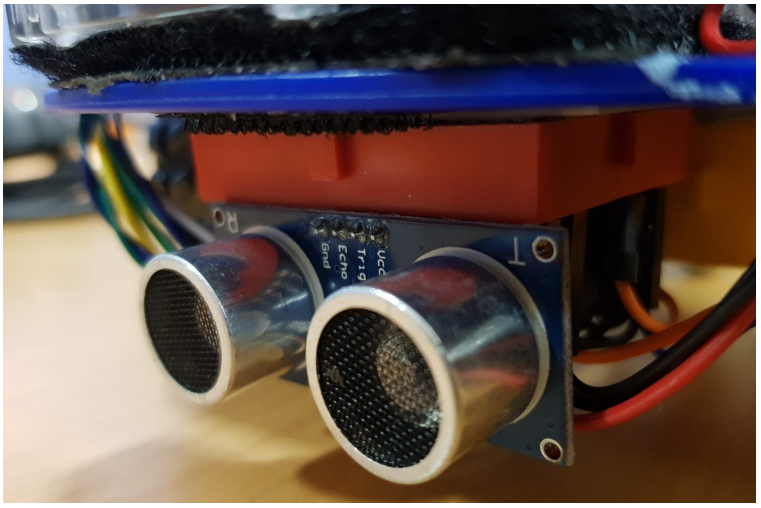

Fig. 4.2: Ultrasonic Sensor facing the front on the Alex

4.3 Path Straightening - FC-51 IR Sensor

To account for the non-uniform running of the motors provided and possibly small spaces Alex has to travel through, the team chose to position 5 infrared sensors at the front left, front right, back left, back right, and rear. The infrared sensors (shown in Fig. 4.3) send back a period signal to the arduino if the distance between Alex and the surrounding walls is less than a certain value, which the user will receive. The user can then decide to move Alex away from the wall to prevent collision.

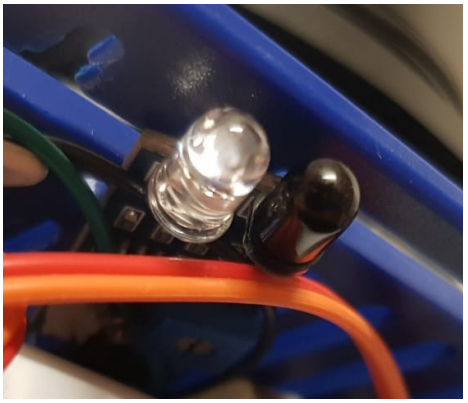

Fig. 4.3: IR Sensor located at the side of Alex that detects if surrounding wall is too close to Alex

4.4 Colour Detection - TCS3200 Sensor Module

During the course of the Search and Rescue mission, Alex will have to identify two coloured surfaces(red and green). To detect the colour of the surface, the team chose to use a colour sensor, mounted in front of Alex, and tapped with a black box to reduce influence from ambient light allowing for more accurate colour reading (shown in Fig 4.4). When Alex has been successfully positioned, the user will send instructions to start colour detection. Data received from the colour sensor will allow the user to know the colour of the surface.

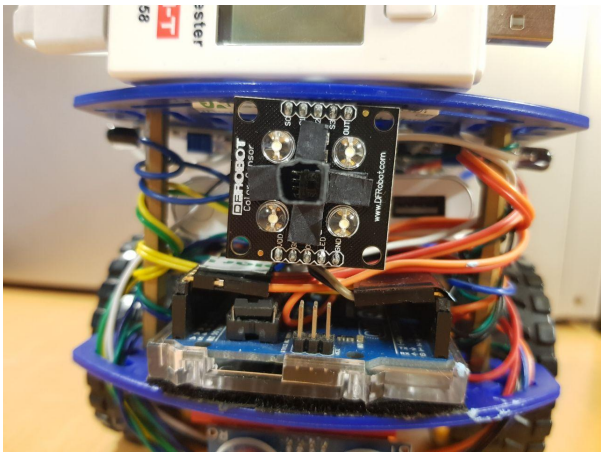

Fig. 4.4: Colour Sensor placed at the front of Alex that scans the surface and return the RGB value of it

Section 5 Firmware Design

5.1 Higher-Order Steps of Overall Algorithm

1. Setup
2. Sending, Receiving and Execution of user command
3. Repeat step 2 till Navigation is over

Further Breakdown:

Step 1: Setup:

1. The Robot Operating System (ROS) hosted by Raspberry Pi (RPi) will establish a connection with the Arduino via wired connection at a baud rate of 9600.
2. The RPi will send a 'hello' packet to the Arduino to verify that connection is established properly. If the packet was sent and received successfully to the Arduino, the Arduino will send back to the RPi an 'ok' packet, but if the packet was not sent successfully, Arduino will send back to the RPi a 'bad magic number' or 'bad checksum' error packet depending on the problem. The team can determine when connection has been established when the RPi terminal shows COMMAND_OK. This will help the team to decide when to re-establish the connection between the RPi and the Arduino.

Step 2: Sending, Receiving and Execution of user command

1. Throughout the mission, the Arduino will poll for data from the RPi.
2. When data is sent from the RPi to the Arduino, the Arduino will read and process the data through the handlePacket function. The Arduino executes the command using the handleCommand function.
3. The handleCommand function will then determine how Alex behaves. Each command will then send back an 'ok' packet to the RPi to allow users to ascertain that command has been received. However, if the command does not match any of the preset cases then the Arduino will send back a 'bad' packet to notify users that the command is invalid.

Step 3: Repeat Step 2 until Search and Rescue Mission is over

5.2 Communication Protocol

Before the other features below can be carried out, the three steps below apply to secure a connection between the ROS and RPi.

1. A node(function) is created on the RPi to receive the instructions from ROS.
   Another node(function) is created on the RPi to receive data from the Arduino, which is directly connected to the RPi. The function then publishes the readings from the ultrasonic, infrared and colour sensors connected to the Arduino, to the Ubuntu laptop, through the RPi.
2. Whenever the keys on the laptop are activated, two float linear and float lateral variables are generated on the laptop and sent to the node on the RPi, which then sends the respective directional command to the Arduino.
3. A subroutine takes in the two float variables: the linear movement instruction 'linX' and the lateral movement instruction 'angZ'. The variables are then processed by the RPi to produce a packet which is then sent directly to the Arduino.

5.2.1 LiDAR Scan

1. The Arduino initiates a scan of the environment using the LiDAR
2. The Arduino sends the data from the scan back to RPi.
3. The RPi relays data to Ubuntu Laptop.
4. Ubuntu laptop creates a map of the environment.

5.2.2 Motor Control

1. The Arduino processes the command type of the packet and writes the corresponding analog PWM signal through the motor driver to each of the two wheels, one on the left and the other on the right side of Alex, to drive Alex in a certain direction.
2. Therefore, the command type of the packet sent from the RPi to the Arduino determines the movement command sent from the Arduino to Alex.
3. The table in Fig 5.2.2 below shows the respective linear and lateral variable values mapped to the command type of the packet sent from the RPi to the Arduino. The corresponding movement expected of Alex is also shown.

Linear Variable Value: linX	Lateral Variable Value: angZ	Command Type of Packet (commandPacket.command =)	Movement Command
0.5	0	COMMAND_FORWARD	Forward movement
- 0.5	0	COMMAND_REVERSE	Reverse movement
0.5	1	COMMAND_TURN_LEFT	Left movement
0.5	-1	COMMAND_TURN_RIGHT	Right movement
-0.5	-1	COMMAND_REV_LEFT	Reverse Left movement
-0.5	1	COMMAND_REV_RIGHT	Reverse Right movement
0	0	COMMAND_STOP	Stop movement
0	-1	COMMAND_FAST	Fast Forward movement

Fig. 5.2.2: Table for movement commands

5.2.3 IR Sensor & Ultrasonic Sensor

1. Whenever a movement command is sent from the RPi to the arduino, a function is called to obtain readings from both the ultrasonic sensor and IR sensor.
2. Once the Arduino has obtained the values, it creates a packet which is then sent back to the RPi.
3. The RPi receives the packet and processes the params array of size 16 of the packet. The first column in the params array is utilised for the integer Ultrasonic distance reading in ‘cm’. The second column in the params array is utilised for the IR value, where the IR value is an unsigned long 16-bit integer. Bits 9-15 of the IR value are used for the five IR sensors located at the rear, back right, front right, back left and back right of Alex respectively. Each bit is set to either ‘0’ or ‘1’, where ‘0’ corresponds to an obstacle detected and ‘1’ corresponds to no obstacles detected.
4. A 16-bit ‘count’ variable is created on the Arduino to receive the ultrasonic sensor distance. We limited the output from the arduino to a 6-bit variable as we only require a reading up to 64 cm since this is to measure distance to an obstacle in front of Alex. This reading is sent to the RPi which shifts the reading to bits 2-7 of the ‘count’ variable. The bits 8-12 are reserved for the IR value. Fig 5.5.3 shows the diagram representation of the 16-bit ‘count’ value that will be sent from the RPi to the ros node.
5. The ros node then receives and processes the 16-bit count value. It outputs the ultrasonic distance reading and sets the IR sensor graphics on the user interface on the laptop to red if there is an obstacle, and green if there is no obstacle for each IR sensor.

5.2.4 Colour Sensor

1. The Arduino then processes the command type of the packet and activates the colour sensor.
2. Therefore, the command type of the packet sent from the RPi to the Arduino also determines the colour command sent from the Arduino to Alex.
3. The table in Fig 5.2.4.1 below shows the respective linear and lateral variable values mapped to the command type of the packet sent from the RPi to the Arduino. The corresponding action expected of Alex is also shown.

Linear Variable Value: linX	Lateral Variable Value: angZ	Command Type of Packet (commandPacket.command =)	Colour Command
0	1	COMMAND_COLOUR	Obtain colour readings

Fig. 5.2.4.1: Table for Colour command

4. Once the Arduino has obtained the colour value of '1' for red and '2' for green, it creates a packet which is then sent back to the RPi.
5. The RPi receives the packet and processes the params array of size 16 of the packet. Since the command here is for finding the colour of the object detected, the Arduino does not obtain the Ultrasonic distance reading and IR sensor values. Thus, even though these values are also sent back together with the colour value, they are not updated. The third column in the params array is utilised for the Colour value, where the Colour value is an 16-bit integer. The Fig 5.2.4.2 and Fig 5.2.4.4 shows a diagram representation of the Colour value and params array respectively.

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Colour	Colour

Fig. 5.2.4.2: Table for Colour Value

6. A 16-bit count value is created on the RPi. The RPi then converts the decimal Colour value to binary form. The bits 0-1 are reserved for the Colour value. The Fig 5.2.4.3 shows the diagram representation of the 16-bit count value that will be sent from the RPi to the ros node.

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				IR Front Right	IR Back Right	IR Front Right	IR Back Left	IR Front Right	Ultrasonic distance reading (binary)						Colour	Colour

Fig. 5.2.4.3: Table for Count Value

7. The ros node then receives and processes the 16-bit count value. It outputs the bits corresponding to the state, as shown below in Fig 5.2.4.5.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Ultrasonic distance reading in ‘cm’ (decimal)	IR value (in binary)	Colour value	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 5.2.4.4: Table for Params Array

Colour Bits	STATE
00	**NONE(WHITE)**
01	**RED**
10	**GREEN**
11	**OFF**

Fig. 5.2.4.5: Table for Params Array

Section 6 Software Design

6.1 Teleoperation between Laptop and RPi

Initialization

A dedicated network is used to provide a wireless connection between RPi and a laptop. Data is transferred between RPi and a laptop running Ubuntu using a function in the Robot Operating System (ROS). RPi is selected as ROS Master and a laptop as the Slave. This is to ensure that if the laptop is disconnected, Alex will still run. A URI(Uniform Resource Identifier) is then created and the shell is configured using the shell script .bashrc. The published IP address of the RPi and Laptop is specified when connected to the same network. When the Master initiates the ROS operating system, the slave will depend on the master for the ROS operating process.

Sending data from laptop to RPi

A node(function) will be created on the laptop to read keypresses on the laptop's keyboard. This function will publish a data string containing two variables: the first variable determines the linear movement instruction while the second variable determines the lateral movement instruction.

Sending data from RPi to laptop

A node(function) was created on the RPi to receive data from the Arduino, which is directly connected to the RPi. The function then publishes the readings. This node is to transmit readings from the ultrasonic, infrared and colour sensors. A function is created on the laptop to receive the published data for the sensors and represents the data in a Graphical User Interface(GUI). A function on the RPi which is provided takes in readings from the LiDAR and publishes the readings. Another function on the laptop that is also provided will read the data and transfer the readings to RVIZ, a function that prints a map based on readings received from the LiDAR.

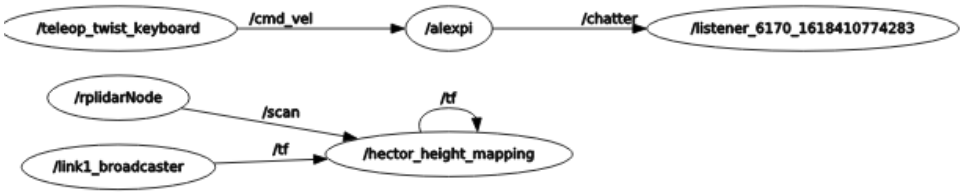

Fig.6.1.1: Node and topic table

For telecommunication, we utilised ROS System on Two Machines instead of Transport Layer Security(TLS). Table 2 presents the reasons we used this function instead.

Category	Benefit
Security	The intended connected device is specified so only the specified device can send and receive data from each ROS machine. Even if a third machine finds out the password and IP address and connects to the same network, they still cannot infiltrate the communication between the RPi and the laptop.
Control	Can have a big picture view of all the nodes (functions) running on both ROS devices the type of message they are sending one another.
	Transmission of information can be deactivated from the master by cutting ROSCORE which contains the nodes and allows the nodes to communicate.
Communication	Once the RPi and the laptop are connected via ROS System on Two Machines, bi-directional real-time data can be transmitted.

Multifunctional	The ROS System on Two Machines is optimised to transmit large data in real-time which is required for sending LiDAR readings. Using this function to transmit readings from other sensors and control the Alex bot's movements reduces the number of methods connections that enable the transfer of data between the RPi and the Laptop. This reduces CPU(Central Processing Unit) utilisation, network(wireless connection) utilisation while improving connection stability between the two devices.

Fig.6.1.2: Benefits compared to TLS.

Overall, these functions enable the operator to have a visual representation of the surroundings of the Alex bot and to identify survivors and well control the Alex bot's movement wirelessly.

6.2 Graphical User Interface (GUI)

A GUI is a program to represent data in a graphical form that is easy for the user to understand. This is especially important in high stress environments such as a search and rescue operation when the operator is required to analyse the environment while avoiding obstacles and identifying survivors. To achieve this, a python program was created to receive data as a topic from the Alex.cpp on the RPi. This topic contains the 16-bit integer consisting of the infrared, ultrasonic, and colour readings. This GUI program will decode the 16-bit integer and represent the data in an intuitive format as follows:

1. Infrared Sensors: Cones represent the proximity to an obstacle in 5 different directions. The cones will remain 'green' to signify "Clear" and turn 'red' to signify an obstacle is in close proximity.
2. Ultrasonic Sensors: The distance is displayed in decimal form in cm.
3. Colour Sensor: The state of the colour sensor as well as the colour is represented in word form and in background colour representation.

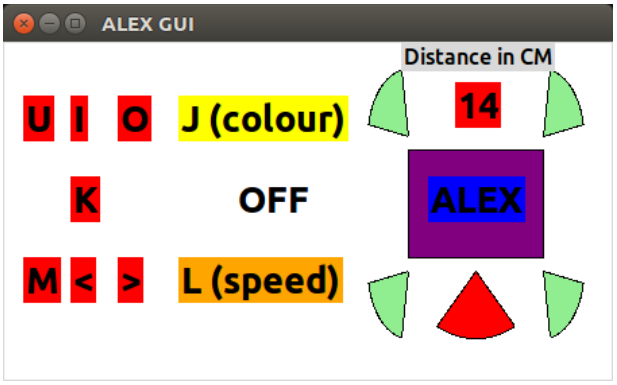

Fig.6.2.1: Alex's Graphic User Interface.

6.3 Power Saving

One additional area of concern the team had to take note of during the Search and Rescue Mission was the power consumption of Alex. In order to achieve lower power consumption, the team disabled the HDMI and ethernet port of the RPi since we are running headless RPi. In addition, the team set up a power saving-function on the Arduino Uno board by disabling the watchdog timer, the Two-Wire Interface (**TWI**), the Serial Peripheral Interface (**SPI**), the Analogue to Digital Converter (**ADC**), and setting the Arduino to **IDLE** mode, as well as disabling pin 13 of the Arduino to off the LED tied to the pin 13.

Section 7 Lessons Learnt - Conclusion

7.1 Important Lessons

1. Big O notation/ processing time affecting serial communication
   In a synchronous serial communication setup, in this case the RPi and Arduino communication, the time taken to process an instruction is important. If the process takes too long, there would be a delay in the control by the user and the feedback to the user which could cause accidents. This delay would also increase the risk of bad messages due to new instructions while arduino is processing the last instruction or sending out the response to the last instruction.
2. Security in data transmission
   During the testing phase for TLS teleoperation between Ubuntu and RPi, the team had difficulty sending data from our Laptop to RPi. We realised that we did not properly sign the Certificate Authority of RPi into our laptop before establishing connection between our Laptop and RPi at the start of our project. This is important as the operator could lose control of Alex if another device interferes with the communication protocol between the laptop (controlled by the operator) and Alex. This would limit the types of operation and the effectiveness of the operation conducted with the Alex bot. As such, our lesson learnt was to establish the certificate for any robot before establishing the connection between the 2 communication devices.

7.2 Greatest Mistakes

1. Skipping the design planning phase.
   This caused delays in the assembly of the Alex bot as our team could not agree on a placement for each device. This led to a delay in functional testing of the hardware such as the required speed of the motor.
2. Messy wiring
   Wiring despite being a simple task can cause many malfunctions as it can interfere with the proximity sensors and mechanical operations such as the wheels. We faced this problem with the wiring for the encoders causing resistance on the encoders which slows the motor speed.

References

[1] Altman, A. (2020, October 9). *Watch a Spot robot from Boston Dynamics explore an old mine*. CNET. https://www.cnet.com/news/watch-a-spot-robot-from-boston-dynamics-explore-an-old-mine/

[2] Frontz, M., (2019). *Wheel Encoders.* https://docs.idew.org/code-robotics/references/physical-inputs/wheel-encoders