CG2111A Engineering Principles and practice II

Semester 2 2021/2022

"Alex to the rescue"

Final report

Team: B01-6A

Name	Student No.	Sub-team	Role
Bui Duc Thanh	A0243318J	Firmware	Coding Arduino & debug
Chien Jing Wei	A0233540R	Hardware	Assembling, wiring & debug
Brandon Owen Sjarif	A0245892R	Software	ROS/SLAM
Chiew Yi Xiang	A0233756Y	Software	Networking - TLS

Section 1 Introduction

1.1 Aim of Project

Natural disasters kill on average 45,000 people per year, globally. They are also responsible for 0.1% of global deaths over the past decade. The first 72 hours after a disaster are the most critical, as the actions and response of the search and rescue team during this period can determine whether the survivors live or die.

Therefore our goal is to design a search and rescue robotic vehicle ‘Alex’ to map out the location it is placed in. It will be remotely operated, with the operator issuing Alex commands to navigate the terrain based on the map information it provides in real time.

1.2 Alex Functionalities

- A Lidar (Light Detection and Ranging) sensor is mounted on top of Alex to map out its surroundings. The Lidar sensor spins at a predetermined frequency to provide a 360-degree view of the bot’s surroundings. These signals are fed back to the Raspberry Pi (RPi), which is then displayed to the operator for him to deduce the mapping of the area around the robot and navigate the robot through the course.

- A pair of Hall Effect Sensors, one on each wheel encoder, detects the turning motion of the wheels and sends these pulses to the Arduino to determine the distance travelled.

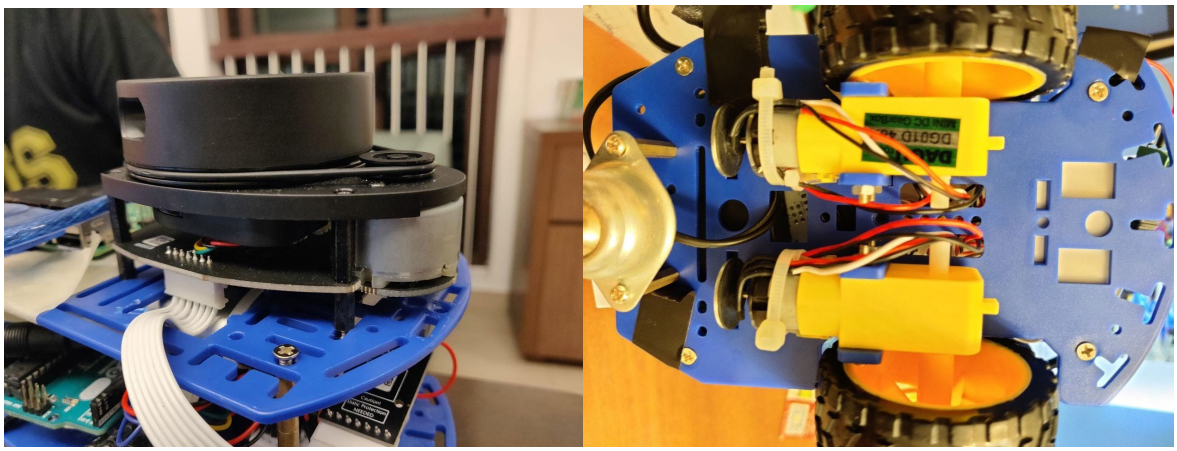

Figure 1.1 & 1.2: Picture of the Lidar and Hall Effect Sensors with the motors

- An Arduino Uno and Raspberry Pi are used to send information between each other to make the robot move according to the user’s inputs. User inputs the commands for direction of movement of the robot as well as the distance and speed (PWM of the motor) to the RPi, which will send a data packet to the Arduino. Once the commands are sent, the Arduino will send back an acknowledgement packet. Depending on whether the received packet has error, bad checksum or is valid, the Arduino will send a corresponding packet to

RPi to inform it. If the command sent is valid, the Arduino will execute the code and move the robot and send a sendOK() function to let the RPi know that it is done. When the Get() function is used, the Arduino will send back the relevant information of the robot, such as the number of left/right ticks, distance travelled etc.

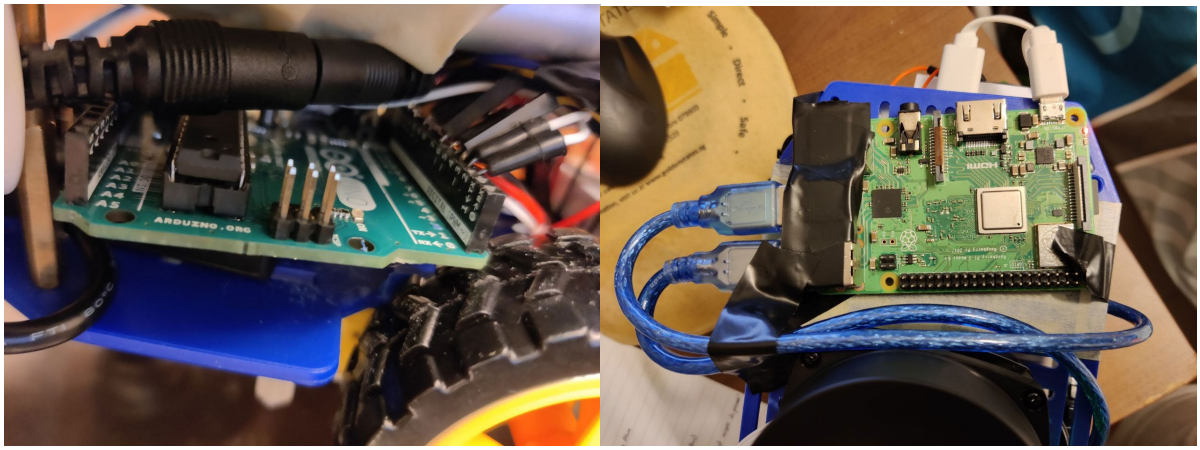

Figure 1.3 &1.4 Picture of the Arduino and Raspberry Pi

Section 2 Review of state of the art

2.1 SmokeBot

SmokeBot is an EU-funded research project. The focus is on civil robots supporting fire brigades in search and rescue missions, e.g. in post-disaster management operations in response to tunnel fires.

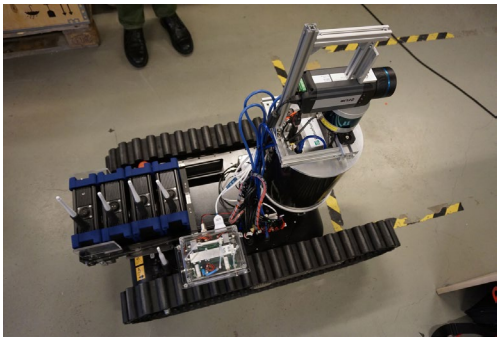

Figure 2.1 SmokeBot

In addition to traditional sensors such as LIDAR and cameras, which are affected by smoke or dust, the sensing unit also includes a novel 3D radar camera, a stereo thermal camera, and high-bandwidth gas sensors. This allows the operators to navigate the smoke bot through smoke which is often found at disaster sites.

The main drawback of the SmokeBot is its size. This will limit its movement and the areas it will be able to access during emergencies.

2.2 Delta Extreme

The Delta Extreme is a remotely operated robotic system designed for inspection and reconnaissance missions inside of confined spaces and hazardous environments. The system includes bi-directional audio, variable intensity lights, and a colour zoom camera. Options include a variety of sensors, a battery pack, Digital Video Recorder (DVR) and laser lines for on-screen sizing.

Figure 2.2 Delta Extreme bot

The Delta Extreme Robot uses a track system specifically designed for operation in sandy or muddy conditions where loose debris or silt may be present. The vehicle is capable of operating over a cable as long as 90 metres and can operate submerged to a depth of 10 metres.

However, there are certain limitations with the track system. The ground clearance is fixed and hence, objects close to the clearance height would get trapped under the bot, which might lead to a weakening of the traction system when debris gets accumulated.

Section 3 System architecture

The diagram below (Fig 3.1) illustrates the system architecture of the Alex rescue vehicle:

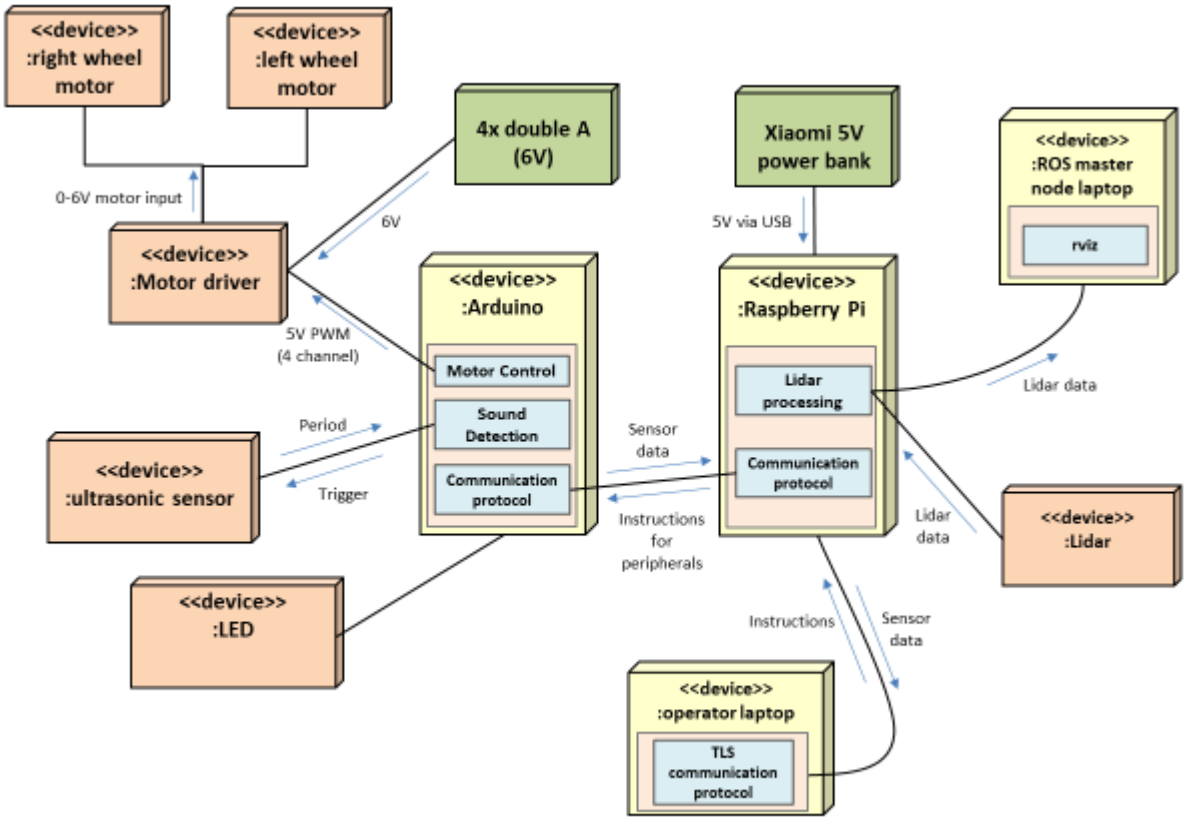

Figure 3.1 System architecture

Hardware components:

1) Motors: Responsible for the actual movement of the robot which are powered by the 4 double A batteries

2) Arduino and LiDar which are both connected to the RPi and communicate through UART communication. The RPi is powered by a 5V power bank.

3) LEDs which are powered by the Arduino. Green LED switches on when Alex is moving and red LED switches on when Alex stops.

Software components:

1) RPi and operator laptop are connected through the same wireless network and using Transport Layer Security (TLS), the operator can send commands to the RPi to move the robot or retrieve data.

2) On a separate laptop, through dual booting or using a virtual machine to run Linux OS, the operator can run the RViz application to see the map of the robot's surroundings more efficiently as compared to VNC from the RPi.

There are two main components connected to the RPi, the Lidar and the Arduino Uno. The Lidar is powered by the RPi and receives data from the surrounding environment and sends this data back to the RPi through the USB to UART conversion module, which is then sent back to the Robot Operating Software (ROS) master set to the operator's laptop IP address through the local network. This data is then processed by the Hector SLAM algorithm on the laptop and depicted in the ROS Visualisation (RVIZ), a 3D visualisation tool, for the operator to see.

For the movement of the robot, the operator inputs the direction followed by the distance and power, which is then transmitted through the RPi to the Arduino and translated into the corresponding values to be subsequently relayed to the motor driver. The motor driver, which is powered by 4 AA batteries, drives the wheels to spin in specific directions. The movement inputs also trigger the LEDs, green when Alex is moving and red when Alex stops. Data such as the distance travelled can be calculated from the number of revolutions the wheel has spun, which is captured by the motor encoder. The data can be seen by using the 'g' command.

Section 4 Hardware design

When designing Alex, we had to consider two main factors: **(1) a small form factor** and **(2) the centre of gravity**

4.1 Form factor

Firstly, we had to minimise the footprint of Alex to reduce the possibility of Alex colliding into walls. To do so, we kept Alex as compact as possible by limiting the use of Alex's parts to the items provided to us initially. The jumper wires are also constrained to the middle of Alex, cable-tied and taped down in order to minimise its movement while Alex is moving. This prevents the wires from coming loose and also prevents them from interfering with other parts of the robot, namely the wheels which will cause greater friction to the wheels, causing them to move slower. We fitted all wires within the blue frames of the Alex as much as possible to minimise the amount of loose parts sticking out of the robot's body.

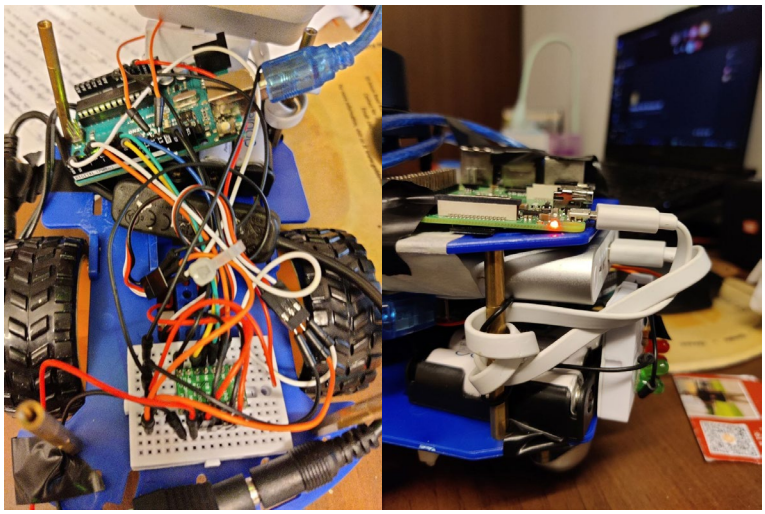

Figure 4.1 & 4.2 Cable management of our Alex

4.2 Centre of gravity

The placement of the parts of Alex were also carefully picked after considering the form factor and centre of gravity of Alex. The power bank was strapped under the upper platform since it is one of the heaviest parts of Alex. This helps to lower the centre of gravity of the robot to make it more stable. The AA battery pack and Lidar are placed at opposite ends of Alex so as to ensure the robot's centre of gravity is close to the centre of the robot, to prevent Alex from tipping over when it goes over the hump.

Initially, the power bank was attached towards the back to counter the weight of the Lidar which was attached at the front of Alex. However, since the AA battery pack was also attached towards the front, Alex was too front-heavy causing it to tip forward after it dismounted the hump. Therefore, we made a decision to shift the battery pack to the back, to counter the weight of the Lidar.

4.3 Additional hardware

An additional hardware feature we added to Alex was LED lights which are located at Alex's rear. Red LEDs will light up when Alex is stopping and green LEDs will light up when Alex is moving, which simulates an actual car moving on the road. We had to use multiple LEDs to ensure that it is bright enough to be seen in actual real world usage.

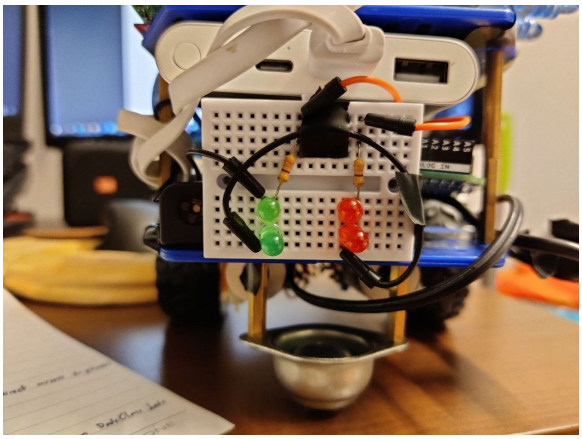

Figure 4.3 LED circuit

We also tried to add an ultrasonic sensor on Alex to measure the distance between Alex and the wall when the Lidar is too close to detect the distance. This way we will be able to tell the distance between Alex and the walls around it and adjust accordingly when Alex moves too close to the wall. However, we had an issue with the software implementation of the ultrasonic sensor and hence decided to remove it from Alex.

Section 5 Firmware Design

5.1 High level algorithm on the Arduino Uno

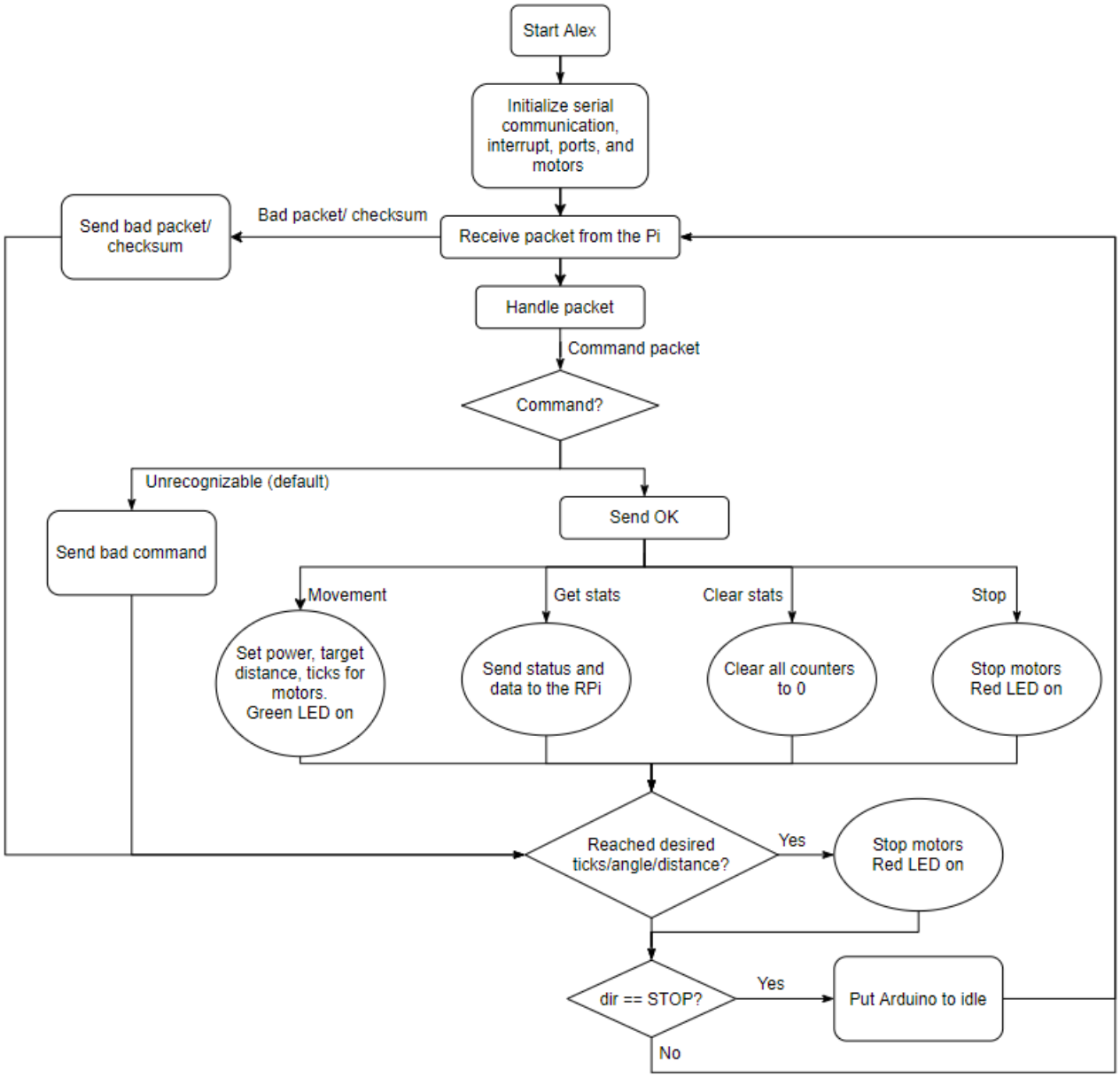

Fig 5.1 High level algorithm on Arduino Uno

5.2 Communication protocol

Messages and answers are transmitted in packets of 100 bytes each:

1) packetType (char - 1 byte): describes the type of the packet
2) Command (char - 1 byte): guides Alex how to handle the information in the packet

3) Dummy (array of 2 char - 2 bytes): pads to the nearest divisible by 4, as the RPi reads 4 bytes at a time
4) Data (array of 32 char - 32 bytes): Information to be sent in char type
5) Params (array of 16 32-bits integers - 64 bytes): Information to be sent in integer type

When sending data from the RPi to the Arduino, the operator input will be written on the "command" char and "params" integers which indicate the movement or activity Alex should execute. After receiving a packet from the RPi, Arduino will compare checksum and magic number to ensure that the packet is received properly. If either checksum or magic number is incorrect then Arduino will send a bad checksum/ magic number packet back to RPi.

1. There are 4 movement commands: forward (f), reverse (b), turn left (l), or turn right (r), which tell the Arduino to activate certain motor pins to move Alex in the desired direction.
    a. Forward (f) and Reverse (b): There are 2 values to be considered, which indicate the distance (in cm) and power, which will be saved into the params array.
    b. Left (l) and Right (r): There are 2 values to be considered, which indicate the angle (in degrees) and power, which will be saved into the params array.
    c. All 4 of the commands once executed will set the green LED pins to 1 for them to light up, and clear the red LED pins to 0 for them to turn off.

After the desired ticks are reached, motors will stop by calling stop() function. The stop command once executed will set the red LED pins to 1 for them to light up, and clear the green LED pins to 0 for them to turn off.

2. The stop command (s) instructs the Arduino to stop all motors.
3. The "get" command (g) will obtain Alex's status from the Arduino (numbers of ticks for each wheel, distance,…)
4. The "clear stats" (c) command will set the number of ticks and distance moved to 0.

The Arduino will send a packet OK to the RPi to acknowledge that the packet was received correctly. By default, Arduino will send a BadCommand error packet to the RPi if the received command is unrecognisable

Section 6 Software Design

6.1 High Level Algorithm

1. Initialization
2. Receive user input
3. Carry out command
4. Mapping
5. Repeat step 2 until navigation is over

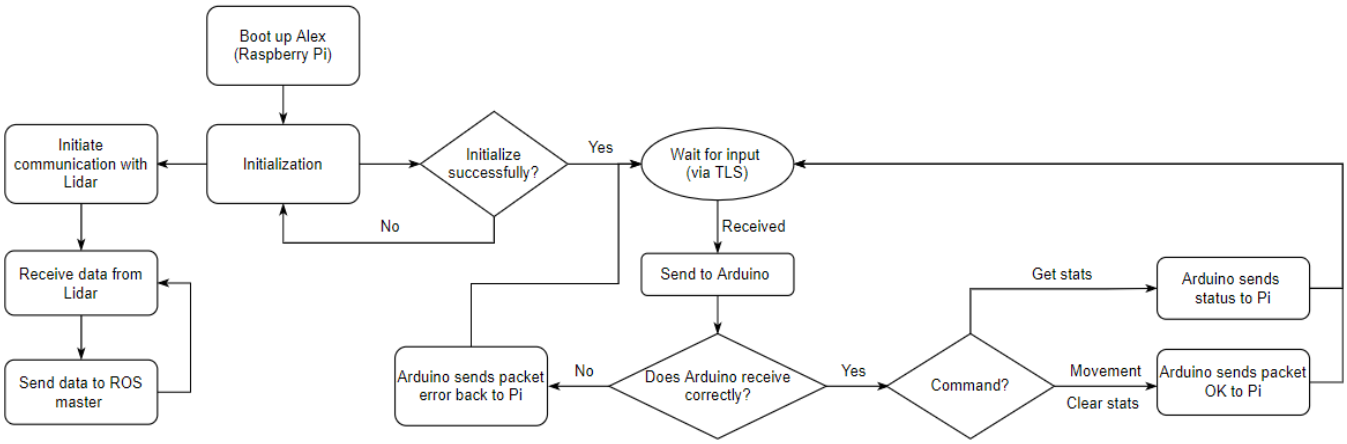

Figure 6.1 Flowchart of high level algorithm on Pi

- Initialization

An initialization handshake will be performed between the RPi and Arduino at the beginning. The RPi will first send over a packet containing a predetermined word to the Arduino. The Arduino, which already has the same word hard-coded in, would compare the packet received from the RPi with its own word and check if they are the same. If they are the same, the Arduino would reply with a packet containing its own hardcoded word. The RPi would receive this packet from the Arduino and compare it with its hardcoded word. If the two words are the same, the RPi will indicate to the operator that the initialization handshake has been successful. Otherwise, the handshake process will be restarted.
The Lidar will be started immediately when it is powered, and an initial scan of the Lidar will be sent to the Pi. The data obtained is stored as an array of 720 values. These values are used to create a map around the starting position of Alex. As Alex waits for operator input, the Lidar will continue to scan the surroundings and update the map periodically.

- User Input

Each transfer of data between the RPi and the Arduino will have two checks, a parity byte checksum and a magic number check. These checks allow us to verify if the transmitted packet is valid. If both checks are passed, the recipient device will then proceed to check if the command sent is valid. Only when all three checks are passed will the command be executed. Otherwise, an error message will be returned.

a. Movement

The RPi will receive the command from the operator and send a packet to the Arduino containing the command. After the integrity checks mentioned above are passed, the Arduino will check against its list of commands corresponding to movement control to verify if the command sent is valid. If the command is valid, the Arduino will execute the command and return the wheel encoder data to the RPi. The data can be viewed using the “g” command or reset to 0 using the ‘c’ command.

- Mapping

As mentioned previously, the Lidar, powered by the RPi, records data from the surrounding environment and sends this data back to the RPi through the USB to UART conversion module, which is then sent back to the ROS master. This data is processed and displayed on RVIZ and maps out the region using the Simultaneous Localisation And Mapping (SLAM) algorithm. Fortunately, by dual booting or using a virtual machine to run Linux OS, we can outsource the computationally intensive algorithm to our laptop and visualise the LiDar data on the RVIZ on it rather than straining the less powerful RPi. Unfortunately, we encountered some problems with our Lidar and had to switch from using the mapping option to using live point clouds to navigate and map out the area Alex was in. This was much more problematic as the perspective would change when Alex turns and we had to remember and keep track of the walls Alex had already seen.

6.2 Additional Features

1) To reduce the power consumption on the Arduino, we used bare metal code to modify the Power Reduction Register (PRR), Sleep Mode Control Register (SMCR) and MCUSR of the Arduino. We shut down the modules that we will not be using like the watchdog timer and the Serial Peripheral Interface. When Alex stops, we also put the Arduino to sleep for that duration.

2) Furthermore, to highlight Alex’s movements more clearly, especially in the dark, we set green LEDs to light up when Alex is moving, and red LEDs to light up when Alex is idle. This will make Alex more visible at night and allows the survivors to know the location of Alex and the rescue team.

Section 7 Lessons learnt - Conclusion

One of the most important lessons that we learned throughout the duration of this project was the value of teamwork. Whilst building Alex, both firmware and software, we encountered a countless number of problems. Even though we technically had our different roles to play during the construction of Alex, we each contributed to each area in one form or another. When one of us would be frustrated over any errors, the entire team would put our heads together to try to find a solution. Some problems could oftentimes be solved by having another perspective or trying to describe the problem.

Another lesson that we learned was learning to search for the answers to our problems online. Most of the problems that we encountered could not unfortunately be solved by searching through the curriculum. At first, we found it difficult to try to google the problem as it could not be too general that we would not be able to find any effective solution nor could it be too specific that no results could be found to help the problem. As a result, we had to do our own research to figure out which sites and sources we would be able to find reliable information and implement them to solve our problems.

One of the greatest mistakes was that we did not plan out the proper placement of the batteries. Initially we placed the batteries at the very front of Alex, which proved to be a problem as Alex would be too front-heavy. In the end, we decided to move the batteries into the inner chassis of Alex. However, as we placed it underneath the power bank and close to the inner wirings of Alex, removing the batteries to charge them took a great ordeal of effort as we had to remove the top of Alex and had to be careful when removing them so as to not accidentally unplug some of the wires.

Another mistake that we made was that we wasted a lot of time trying to run hector slam through the use of a virtual machine. Due to how virtual machines would change the ports and the IP addresses of the laptop, we spent hours trying to troubleshoot and figure out which combination of ports and IP addresses would be the correct one. We initially opted for using a virtual machine because we thought that the ROS distribution of the laptop had to match that of the Pi and we were unable to dual boot to Ubuntu 16.04. Fortunately, we were able to realise that we need not match the ROS distribution of the laptop with the Pi and were able to use Ubuntu 20.04.

References

Facts + statistics: Man-made disasters. III. (n.d.). Retrieved April 17, 2022, from https://www.iii.org/fact-statistic/facts-statistics-man-made-disasters

Achim J. Lilienthal, M. L. (n.d.). Smokebot Summary. SmokeBot. Retrieved April 17, 2022, from http://www.smokebot.eu/project.html

IRobot 510 PackBot multi-mission robot. Army Technology. Retrieved April 17, 2022, from https://www.army-technology.com/projects/irobot-510-packbot-multi-mission-robot/

State-of-the-art robot to better assist search and rescue operations in ... - cordis. (n.d.). Retrieved April 17, 2022, from https://cordis.europa.eu/article/id/241251-stateoftheart-robot-to-better-assist-search-and-rescue-operations-in-lowvisibility-conditions

Inuktun Services Ltd.. delta extreme inspection crawlers. Canadian Underground Infrastructure. (n.d.). Retrieved April 18, 2022, from https://www.canadianundergroundinfrastructure.com/product/3273/delta-extreme

Silveira, J. D. (n.d.). *Delta extreme Crawler: Rent, lease or buy*. Nexxis USA. Retrieved April 18, 2022, from https://nexxis.com/product/delta-extreme-crawler-2/