CG1112 Engineering Principles and Practice

Semester 2 2020/2021

"Alex to the Rescue"

Final Report

Team: B03-04B

Name	Student ID	Sub-Team	Role
Lee Yi Kai	A0218514J	Hardware	Design & Procurement Lead
Liang Yuzhao	A0217766U	Software	Networking Lead
Ng Cheng Yang, Titus	A0217442L	Software	ROS/SLAM Lead
Nguyen Van Binh	A0219795L	Hardware	Coding & Debug Lead

Table of Contents

Section 1 Introduction

1.1 Overview of Aims

Natural disasters kill approximately 60,000 people annually and are responsible for about 0.1% of global deaths [1]. In these situations, there is a need to search for and rescue the largest number of casualties in the shortest time while minimizing risk for the rescuers.

Our goal is to build a remotely-operated search-and-rescue (SAR) robot, Alex, which is capable of mapping out the terrain it is placed in. Using the map information, an operator would then issue commands to navigate Alex around the terrain.

Alex can detect obstacles in its environment, and identify the colours of objects within the terrain. Alex should also draw minimal power to preserve battery charge levels.

1.2 Methodology & Functionalities

To achieve the aims stated above, the following functionalities are implemented:

- 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.
- 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 traveled. The Arduino also sends signals, which are entered by the operator and relayed to the Arduino through the RPi, to both wheels to set each wheel to rest, turn forward or turn backward.
- When the colour-sensing command is issued, a colour sensor will detect the colour of an object in front of Alex and return the detected colour to the operator.

Section 2 Review of State of the Art

2.1 Variable Geometry Tracked Vehicle (VGTV)

The VGTV was initially built for industrial and air-conditioning duct inspections. However, it was ultimately made capable of functioning as a search-and-rescue robot, debuting its first SAR operation during the 9/11 attacks.

The original VGTV's chassis was capable of modifying its shape to fit the confined quarters it needed to operate in. Two sets of tracks that take the form of conventional crawlers provide stability and increased traction, while three sets of track wheels allow the bot to change into three different shapes, as demonstrated in Figure 2.1 [5].

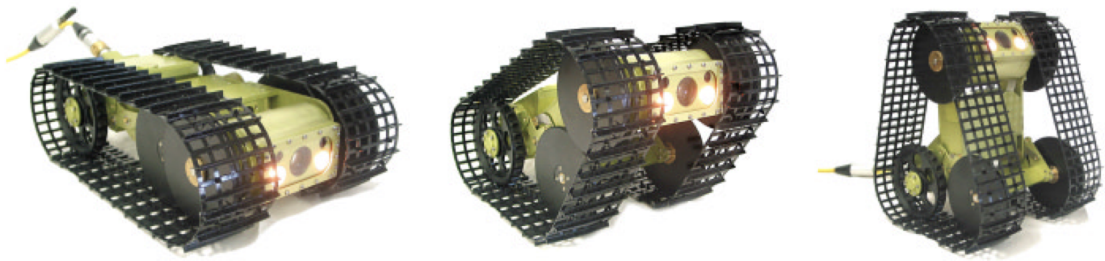

Figure 2.1 Illustration of the different shapes the VGTV can change into

The VGTV also featured a camera and bi-directional audio, allowing rescuers to see underneath the rubble and communicate with survivors [5]. All this information is wirelessly sent to a handheld console where the operator controls the bot and views the information relayed.

However, there were limitations with the track system of the VGTV. The fixed ground clearance meant that objects close to the clearance height would get trapped under the bot. The resulting accumulation of debris would lead to a weakening of the traction system. The adaptation of traditional timing track systems also contributed to the buildup of debris under the chassis of the VGTV, as there was no opportunity for debris that was trapped to be freed from the track pulleys and belt [5].

2.2 iRobot Packbot

The Packbot was initially designed as a military robot for the United States Army to detect and defuse Improvised Explosive Devices (IEDs).

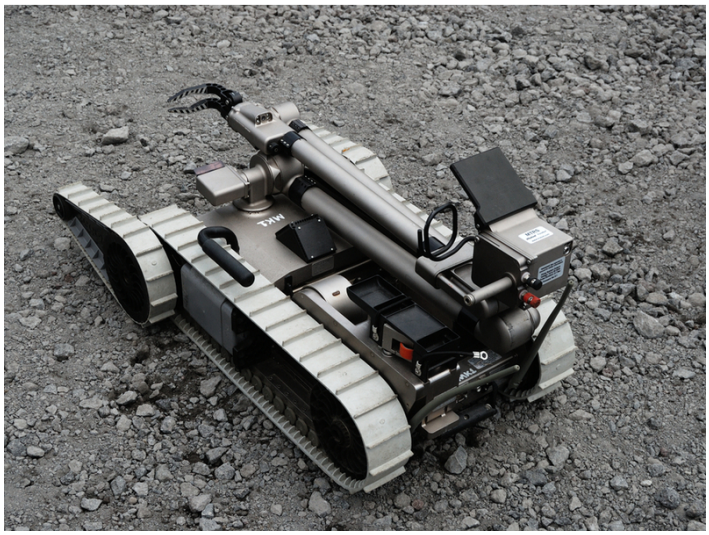

Figure 2.2 Photo of an iRobot Packbot

The Packbot, like the VGTV, runs on tracked wheels to provide greater maneuverability around obstructions [4]. It features flippers, which help it better navigate obstacles and allows the bot to restore its upright position if it inadvertently lands upside down [2]. The bot is installed with

four cameras mounted on an arm, which allows for a wider range of pan-and-tilt functionalities instead of the VGTV's fixed mounted camera. Like the VGTV, the Packbot features two-way audio for communication. A PlayStation controller is used to control the robot's movement, providing an easy-to-use interface to operate the Packbot. A mobile control unit features a 15-inch display, providing two simultaneous camera feeds to the operator. Hot-swappable batteries meant that should the robot run out of battery, a new one could be swapped in immediately, and the bot could then be sent back into the field, reducing operational downtime [3].

One main drawback frequently noted about the Packbot is its lack of speed - the maximum operating speed is 9.3km/h, which could be an issue given the urgency needed for SAR operations [6].

Section 3 System Architecture

This section aims to illustrate the system architecture of Alex. A diagram of the intended structural design of the Alex SAR system is provided below in Figure 3.1.

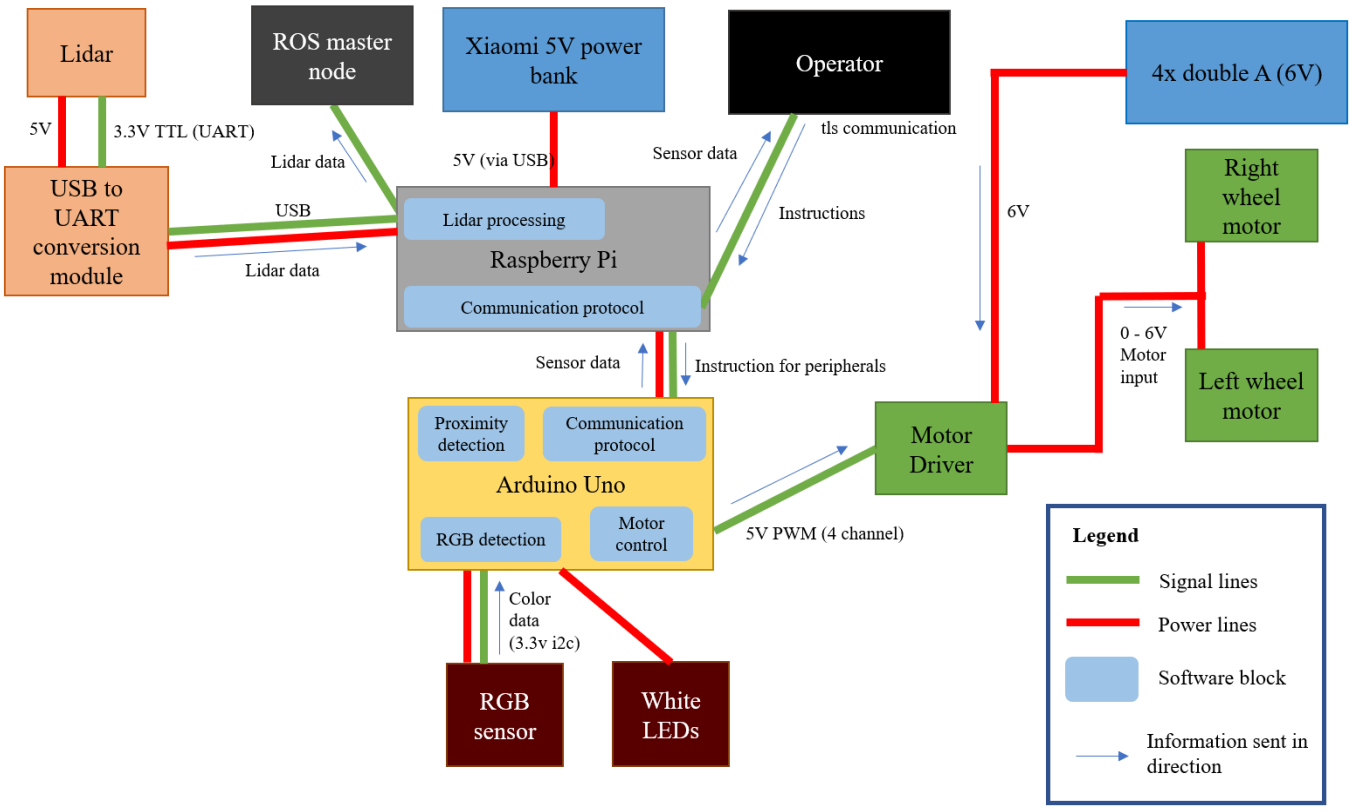

Figure 3.1: System Architecture of the Alex SAR system

The RPi, powered by the 5V power bank, controls the Arduino and the Lidar through Universal Asynchronous Receiver/Transmitter (UART) communication. The Arduino is connected to the Red/Green/Blue (RGB) colour sensor, white LEDs, and motors.

The RPi and the operator device are connected to the same wireless network. Using Transport Layer Security (TLS), the operator sends movement and information retrieval instructions to the RPi, and the RPi sends back sensor data after executing movements or when the operator requests for sensor data.

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 node, which is on the operator's laptop. This data is processed and displayed on the ROS Visualization (RVIZ), a 3D visualization tool, for the operator to see.

The Arduino Uno has two main functions, color sensing and movement. When color sensing is required, the Arduino Uno will power on the white LEDs and the color sensor (APDS-9960 RGB) and begin obtaining the color data required. The color sensor uses the Inter-Integrated Circuit (I2C) protocol to send data back to the Arduino Uno, which will then be processed in the Arduino Uno to determine the color before sending the color code back to the operator via the RPi.

For movement, the operator inputs the direction, power, and distance, which is then translated through the Arduino into the respective values required, and subsequently relayed to the motor driver. The motor driver, which is powered by 4 AA batteries, drives the wheels to spin in specific directions, and data, such as how many revolutions the wheel has spun, is captured by the motor encoder.

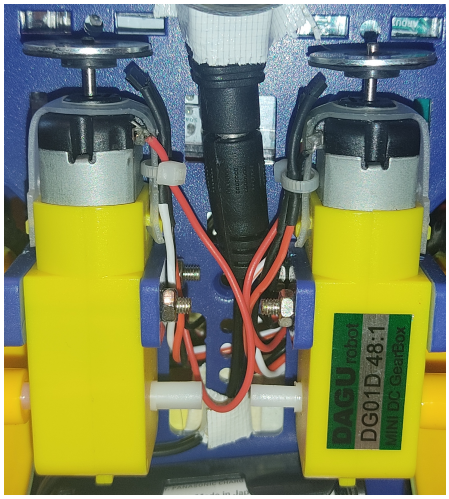

Figure 3.2 Picture of the Motor Encoder

This data is relayed back from the Arduino, and through the RPi to the operator using TLS.

Section 4 Hardware Design

When conceptualizing the design for Alex, we considered the following factors: form factor and center of gravity (CG).

For the form factor, we wanted to achieve a small footprint as it reduces the possibility of overhanging parts colliding into walls while navigating the maze. To do that, we constrained most of Alex's parts to the platforms provided in the base kit. Most of the jumper wires are constrained to the middle of Alex and firmly taped down, which prevents them from extending out of the plates or from contacting the wheels, which could cause greater friction to the wheels or, in the worst case, cause the internal wiring to come loose (see Figure 4.1).

The power bank was strapped to the underside of the upper platform, and the AA battery pack was strapped to the underside of the bottom plate. The Arduino and the Lidar cables posed a challenge to organize as they were long and inflexible. To prevent these cables from inadvertently touching the walls of the maze, we opted to wrap our cables around the standoffs to reduce the total width of Alex (see Figure 4.2).

Figure 4.1 (Left): Jumper wires are constrained to the middle of Alex

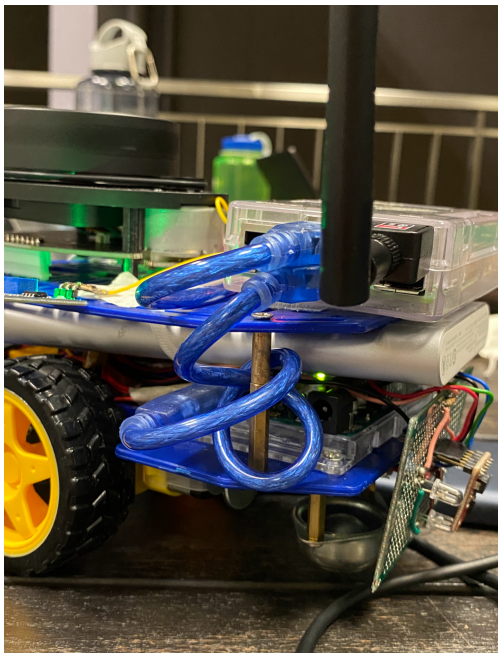

Figure 4.2 (Right): Arduino and Lidar cables (Blue) wrapped around the standoffs (gold)

To implement color sensing, we provisioned additional LEDs to brighten up the object in the event of poor lighting. Initially, we used a breadboard to house and wire the components. However, the wires would frequently drop out of the breadboard connectors, due to the loose connections. Hence, we decided to solder the color sensor onto a protoboard, with LEDs flanking both sides of the color sensor.

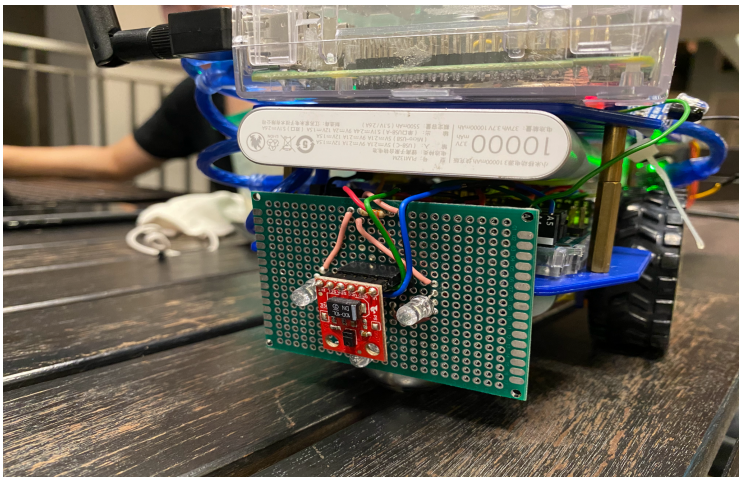

Figure 4.3: Protoboard with LEDs and colour sensor (red) soldered on.

Due to the inclusion of a hump in the maze, we needed to factor in the center of gravity of Alex to prevent it from tipping over while mounting and dismounting the hump. With the AA battery pack placed at the rear end of Alex, the likelihood of Alex tipping when the ball caster strikes the edge of the hump is very high. Hence, we have shifted the power bank slightly forward, placing more weight on the ball caster to prevent potential tipping.

Section 5 Firmware Design

5.1 High-level algorithm on the Arduino Uno

The high-level algorithm on the Arduino Uno in Alex is illustrated in Figure 5.1 below.

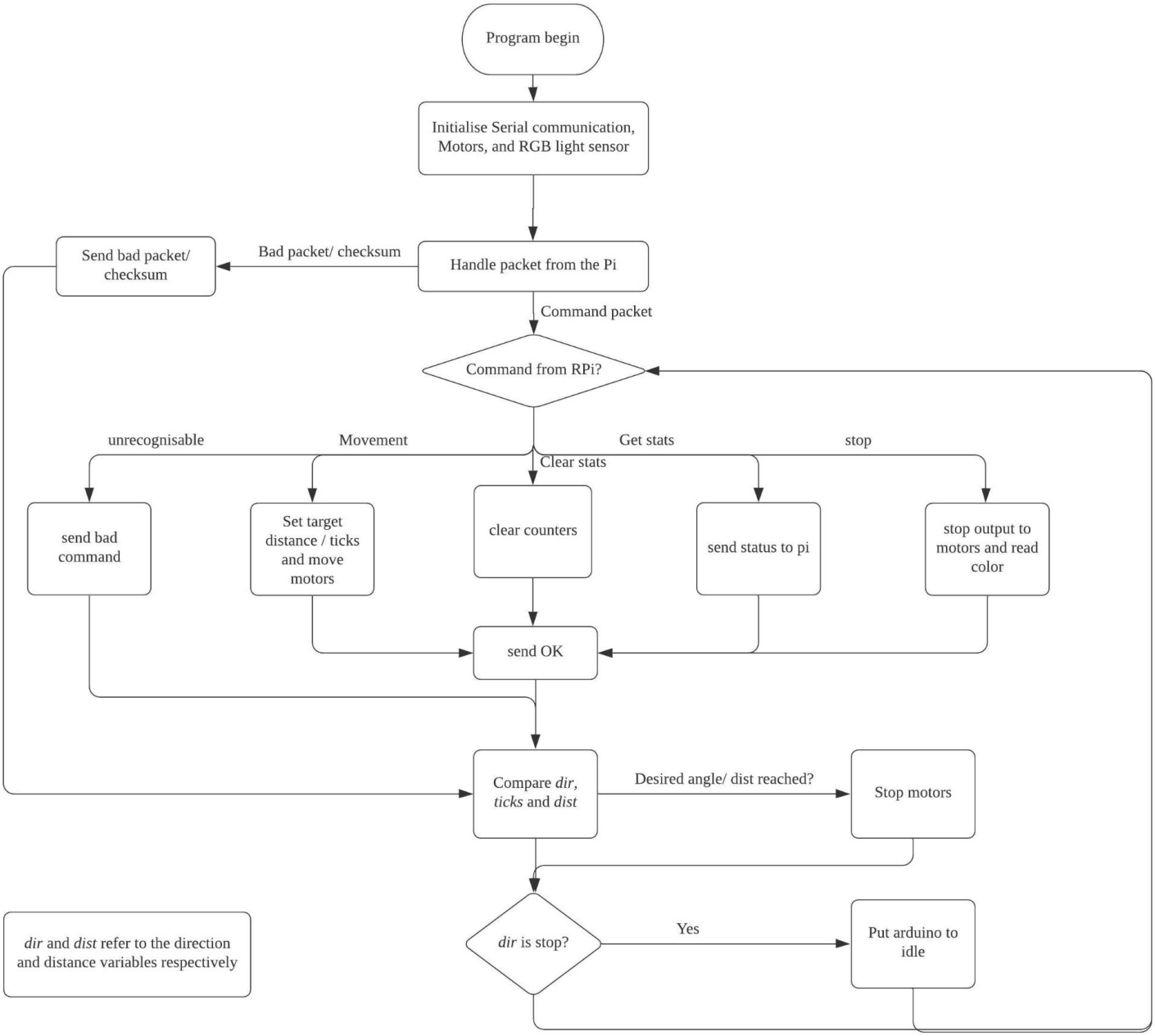

Fig 5.1: Firmware design flowchart of Alex

Once the program begins, the Arduino will first initialize its necessary sensors, outputs, and communication protocols.
Then, it will repeatedly check for and read any incoming packets through serial communication with the RPi. Upon receiving the command, the Arduino will handle the packet by returning a

bad packet/checksum response to the RPi or reading the command and executing it accordingly. A more detailed explanation of each command can be found below, in section 5.2.

5.2 Communication protocol (format of messages and responses).

Messages and responses are sent in the form of packets, which consists of:

1. packetType (char)
2. command (char)
3. dummy (array of 2 char)
4. data (array of 32 char)
5. params (array of 16 32-bit integers)

"packetType" describes the packet information, while "command" tells us what to do with the information in the packet. The packet "dummy" is required as the pi reads four char characters at a time. Finally, "data" and "params" contain the information to be sent, in char and integer data types, respectively.

When sending data from the RPi to the Arduino, the "command" and "params" chars describe what movement or sensing function Alex should execute based on the operator input. If the packet is received as expected by the Arduino (with the correct magic number and checksum), the Arduino will return an "OK" packet to the RPi to indicate that the packet was well received. Otherwise, it will return an error packet indicating the error to the RPi.

The "command" char contains characters that correspond to different functions of Alex, which are:

1. The forward (f), backward (b), turn left (l), or turn right (r) commands instruct the Arduino to activate certain motor pins to move Alex in the desired direction.

    a. For the forward (f) and reverse (b) commands, the operator will be required to input 2 additional numbers corresponding to distance (in cm) and power, which will be saved into "params".
    b. For the left (l) and right (r) command, Alex will turn by approximately 10 degrees in the specified direction at 100% power. This information is hardcoded into "params".

    After the desired turning angle or movement distance is reached (i.e. the change in wheel ticks is greater than the input distance/angle), an automatic stop command is sent to stop the motors.

2. The stop command (s) instructs the Arduino to stop all motors and sense the object's colour. The Arduino is then put to idle mode.

3. The “get stats” command (g) will retrieve the movement and colour-sensing information stored from the Arduino, which includes distance moved, wheel encoder ticks, and the colour of the last object sensed.
4. The “clear stats” (c) command will clear all stored information on movement and colour sensing on the Arduino.

Once the command received has been fully executed, the Arduino will send a packet to the RPi indicating that the command has been successfully executed. Any other command entered will return a “BAD COMMAND” error packet to the operator.

When data is sent from the Arduino to the RPi, the “params” bits will contain total forward and reverse distance moved, number of wheel ticks for each wheel in both directions, and the colour of the last object sensed.

Section 6 Software Design

High-level Algorithm

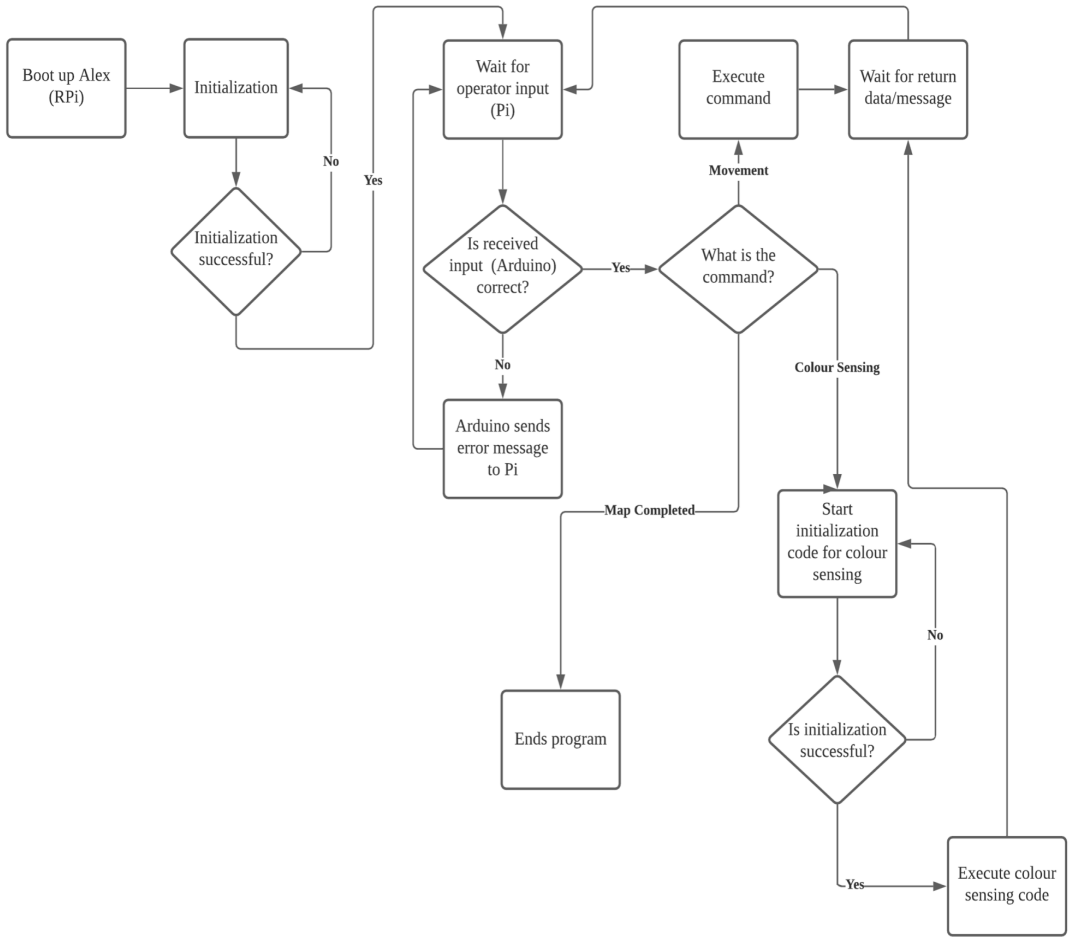

Fig 6.1 High level Algorithm on the Pi

Breakdown of Algorithm

Initialization

The RPi and Arduino will perform an initialization handshake. The RPi will first send over a packet containing a predetermined word (for example, “hello”) to the Arduino. The Arduino, which already has the same word (“hello”) hard-coded in, would compare the packet received from the RPi with its own hardcoded 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 (for example, “world”). The RPi would receive this packet from the Arduino and compare it with its hardcoded word (“world”). If the two words are the same, the RPi will indicate to the operator that the initialization handshake was successfully completed. Otherwise, the handshake process will be restarted.

The Lidar will be started, 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 will be used to create a map of the starting position of Alex. As Alex waits for operator input, the Lidar will continue to scan the surroundings and update the map accordingly.

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 the integrity of the transmitted packet. 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, which can be viewed using the "g" command.

b. Colour sensing

   An RGB sensor is connected to the Arduino. The command to execute colour-sensing is integrated with the "s" command. Once the integrity checks mentioned above are passed, the values returned from the RGB sensor will be compared against one another. Since the maze only includes red and green objects, the red and green values returned will be compared, and whichever value is greater is saved as the detected colour. Then, the detected colour is returned to the Pi as an integer and can be viewed by the operator using the 'g" command.

Mapping

The RPi repeatedly receives Lidar data and publishes it as a node using ROS. On the operator side, a laptop acting as the ROS master subscribes to the Lidar data and maps out the area using Simultaneous Localisation And Mapping (SLAM) algorithm. This allows us to offload the computationally taxing algorithm to a laptop, reducing the power draw on the Pi.

Section 7 Lessons Learnt - Conclusion

During the design phase of Alex, we considered features that we wanted to implement but overlooked the items we wanted. For example, the color sensor we had initially procured, TCS3200, needed too many wiring connections to the Arduino, which we could not provide due to the lack of pins on our Arduino board. As a result, we had to source for a different color sensor, costing us both time (to procure the new sensor and downtime from being unable to program the sensor) and money. This happened again during our search for buzzers, as we assumed that buzzers were all the same, and so we went for the cheapest possible buzzer we could find. However, the buzzer we got was an alarm buzzer that was not to generate tones that Alex needed to play music. (Note: we did not implement the music into Alex in the end). From this project, we have learned to plan to ensure that parts procured could be integrated into Alex.

Another major hurdle we had to overcome was when installing new parts into Alex. Even though each additional module only took up a little bit of space and only required at most four more wires, we ended up frequently tearing Alex down completely and rebuilding it from scratch to accommodate the additional module that we wanted to install. This was likely a result of us not looking sufficiently far ahead to leave the required space for the additional modules in our hardware layout and coming up with other features on the fly after having completed our construction of Alex. Even though we had a rough idea of how we wanted to lay out the many different parts, our plans often did not turn out the way we expected. From this, we have understood the importance of planning far ahead for the additional features we want to implement and the space required to implement them.

References

[1] H. Ritchie and M. Roser, “Natural Disasters,” *Our World in Data*, 03-Jun-2014. [Online]. Available: https://ourworldindata.org/natural-disasters. [Accessed: 07-Mar-2021].

[2] E. Grabianowski, “How Military Robots Work,” *HowStuffWorks Science*, 08-Jul-2020. [Online]. Available: https://science.howstuffworks.com/military-robot.htm. [Accessed: 07-Mar-2021].

[3] “FLIR PackBot (Datasheet).” [Online]. Available: https://www.flir.asia/products/packbot/. [Accessed: 07-Mar-2021].

[4] J. D. Sutter, “How 9/11 inspired a new era of robotics,” *CNN*, 07-Sep-2011. [Online]. Available: http://edition.cnn.com/2011/TECH/innovation/09/07/911.robots.disaster.response/index.html. [Accessed: 07-Mar-2021].

[5] M. J. Micire, “Evolution and field performance of a rescue robot,” *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 17–30, 2007.

[6] “Small Platform Tactical Robots Assessment Report.” United States Department of Homeland Security, Science and Technology Directorate, Oct-2014.

[7] “iRobot 510 PackBot Multi-Mission Robot,” Army Technology, 15-Apr-2021. [Online]. Available: https://www.army-technology.com/projects/irobot-510-packbot-multi-mission-robot/. [Accessed: 17-Apr-2021].