CG1112 Engineering Principle and Practice
Semester 2 2020/2021

"Alex to the Rescue"
Final Report
Team: B03-6A

Name	Student #	Sub-Team	Role
Jin Minyue	A0225642J	Software / Firmware	Software Subteam Lead
Muhammad Ashraf B Mohamad J	A0217396W	Software / Hardware	Nominal Team Lead
Peng Yanjia	A0217841E	Firmware / Hardware	Firmware Subteam Lead
Renzo Rivera Canare	A0217466Y	Firmware / Hardware	Hardware Subteam Lead

Table of Contents

Section 1: Introduction

On 13 February 2021, a 7.1 magnitude earthquake struck the coast of Tohoku, Japan, disrupting the lives of thousands of residents across 9 prefectures [1]. The earthquake left at least 100 people injured from the various prefectures [2]. The numbers could have been reduced if search-and-rescue (SAR) robots were deployed after the incident. Rescue teams often risk their lives trying to save victims as the rubble is unstable and may crumble over them if touched. However, with advancements in robotics, SAR robots can be used to analyse the environment and locate survivors in hazardous areas, thus reducing the risks faced by these rescuers. As such, we have designed and built a remotely operated SAR robot called Alex that can assist the rescue teams in gathering data of the environment and locate victims within the disaster-stricken area.

Section 2: Review of State of the Art

In this section, we will be discussing 2 state-of-the-art tele-operating SAR robotic platforms that are currently available in the market.

1. Spot by Boston Dynamics

Firstly, we have Spot by Boston Dynamics, as seen in Figure 1, which can analyse hazardous and otherwise inaccessible environments. Its functionalities include a 360° live feed, remote operation, object detection, obstacle avoidance and dynamic balance for unstable environments [3].

Spot has many strengths of which include: the ability to climb over low obstructions, which allows it to traverse through rubble and debris-filled environments; its variable height which enables it to avoid collisions and crawl through low and tight spaces; and its ability to right itself independently if it topples over [3]. However, its main weakness is its inability to rescue the victims, since it can only carry up to 14 kg of equipment [3].

2. Guardian S by Sarcos

Secondly, we have the Guardian S by Sarcos, depicted in Figure 2, which has similar functionalities to Spot. Its remote operation can reach up to a 4.8-km range. Additionally, it has 2-way audio functionalities and can also move vertically on ferromagnetic surfaces [4]. However, unlike Spot, the Guardian S has a snake-like build and movement which allows it to traverse over challenging terrains and through extremely tight spaces with a minimum of 7” diameter opening.

The Guardian S has many strengths similar to Spot, as mentioned above. Furthermore, its surveillance time of up to 12 hours allows for extended usage for multiple missions without having to be recharged, which is essential when trying to save victims within the ‘golden period’

of 72 hours. Its two-way audio allows rescuers to communicate with any trapped survivors [4]. However, like Spot, due to its small size, the Guardian S cannot rescue people, and can only gather information on the environment.

From our analysis of these two SAR robots, we learnt about the characteristics of a state-of-the-art SAR robot. Most importantly, the robot has to be small in order to fit through compact and tight spaces, which is necessary in disaster-stricken areas. Also, good software design would allow the SAR robot to be remotely operated, provide a live feed and detect objects or walls, to be of use to rescuers.

While we have been limited on what can be used and by our budget, we have tried to emulate some feasible characteristics that we deem possible. One similar characteristic between the 2 robots mentioned above and Alex is their size. Alex is small in size and this allows the operator to move it through tight spaces caused by debris. On top of that, Alex is able to be remotely operated and can provide an accurate live feed of its surroundings by scanning the area using the LIDAR.

Figure 1. Spot by Boston Dynamics [3]

Figure 2. Guardian S by Sarcos [4]

Section 3: System Architecture

Figure 3 illustrates the system architecture of Alex. The orange boxes represent the physical components while the pink boxes represent different software functions. The remote computer, controlled by the operator, transmits commands to and receives information from the Raspberry Pi (RPi). The remote computer has to be connected to the same Wi-Fi network that the RPi is connected to before it can access the RPi's terminal via Secure Shell Protocol (SSH). The RPi communicates with the Arduino through the Universal Synchronous/Asynchronous Receiver Transmitter (USART). This allows the RPi to receive information about the robot's surroundings and transmit command data to the Arduino.

When the robot is ready to start the operation, the operator will input commands into the RPi terminal directly via the remote computer. The RPi will serialise the data to an agreed format between the Arduino and itself and transmits it to the Arduino.

The LIDAR measures the distances between Alex and the walls or objects in the environment and sends them to the remote computer via the RPi acting as a node. The remote computer converts the data into a visual map using Simultaneous Localization and Mapping (SLAM). This user-readable map of Alex's surroundings will then be used to navigate Alex through the environment.

The operator can then input movement commands into the RPi, which transmits them to the Arduino. Subsequently, the Arduino will process this data and use the motor control code to control the motors through Pulse Width Modulation (PWM), causing them to move accordingly. The Wheel Encoder captures the rotation of the motor and communicates this information to the Arduino, which converts the rotations into the distance moved by Alex and sends the distance data to the RPi.

Similarly, the operator can input commands to the RPi to activate the RGB sensor, IR temperature sensor, gas sensor, sound sensor and flame sensor via the Arduino, which will send the respective data collected to the RPi.

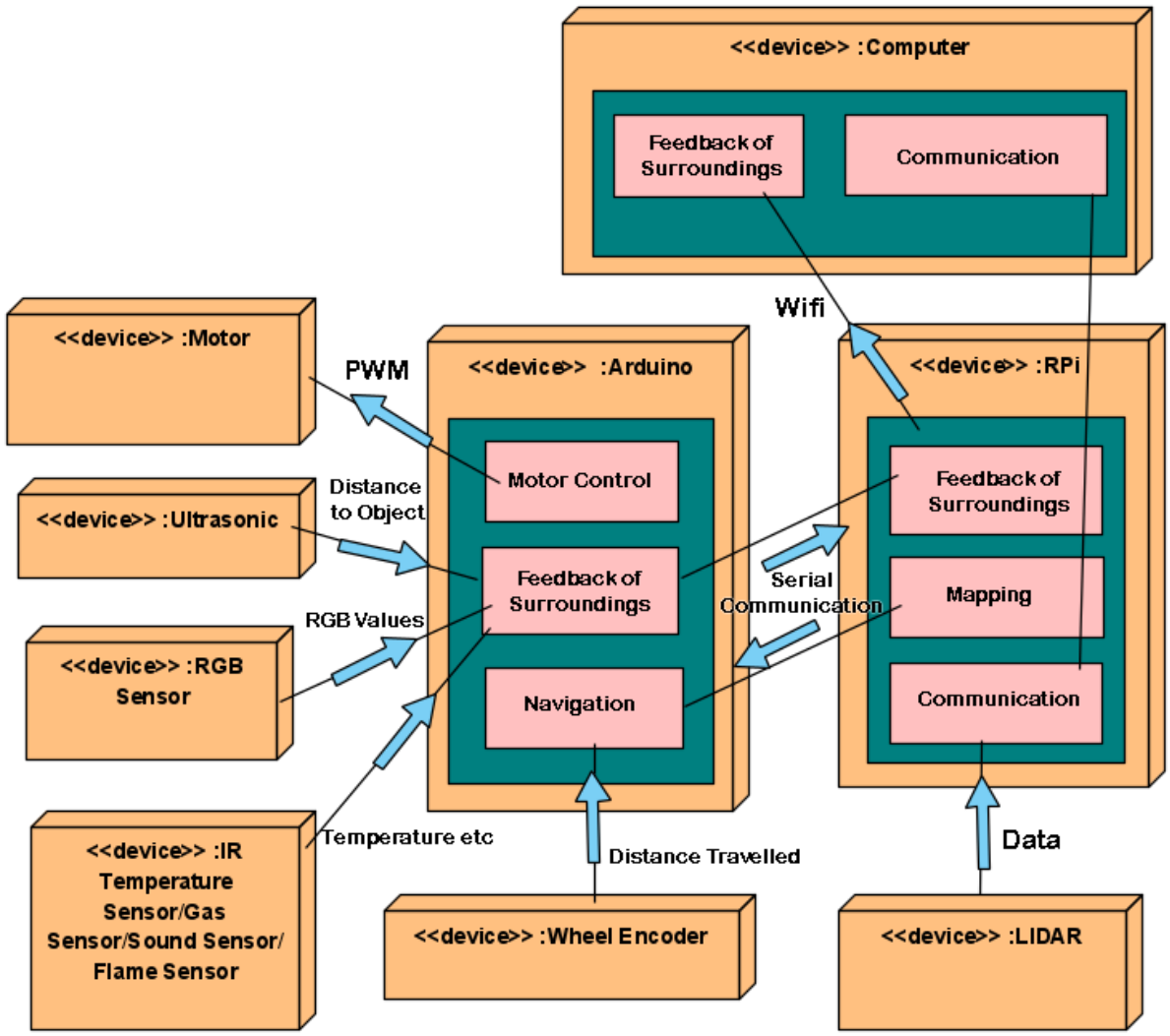

Figure 3. UML Deployment Diagram of Alex's System Architecture

Section 4: Hardware Design

This section focuses and elaborates on the hardware design for Alex. The primary purpose of the components in Alex is to allow it to analyse the environment. The algorithm is represented below, including a further breakdown of each step.

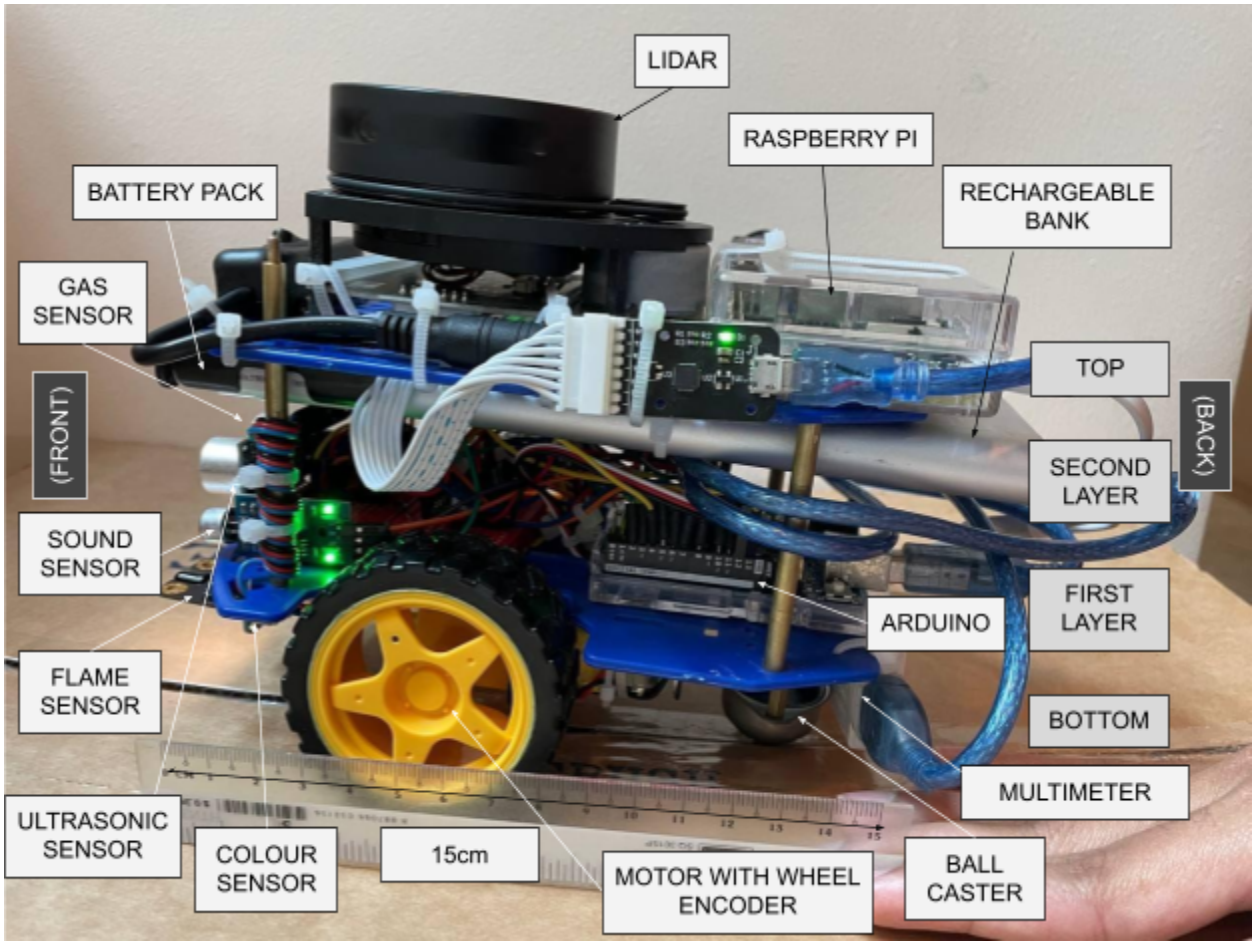

Figure 4. Assembly layout of Alex with its important hardware components labelled.

The Alex robot has a compact form factor that includes many different sensors mounted on the body, which houses the main components of the robot, the Arduino and Raspberry Pi. Alex consists of four distinct layers, separated by two waffle plates that are situated one above the other. We will now discuss each layer in further detail, starting with the bottom layer:

(1) Bottom layer (Underneath lower waffle plate): Below the lower waffle plate, there are two motors. These motors are used to drive Alex along with a wheel encoder attached to the secondary shaft of each motor. The wheel encoder tracks the rotation of the motor. Additionally, one ball caster serves as the third point of contact with the ground and

allows Alex to balance and turn about the centerpoint of the main wheels. Behind the ball caster is the multimeter to measure the current and voltage supplied to the Raspberry Pi from the battery bank. Lastly, a small breadboard which connects the colour sensor is located at the nose of the robot, facing towards the front along with a flame sensor beside it.

(2) First Layer (Above the lower waffle plate): The Arduino and the main breadboard is located in the middle of the plate, which serves as the hub for connection for most of the electronic components of Alex. An ultrasonic sensor, gas sensor and a sound sensor are also placed facing forward at the front of the lower waffle plate.

(3) Second Layer (Below the upper waffle plate): The rechargeable battery bank along with the battery pack are centered above the first layer, both of which act to balance Alex and position its center of gravity in the middle, above the wheels.

(4) Top Layer (Above the upper waffle plate): The Raspberry Pi, the computer of the robot is located here. It serves as the centerpoint of communications and processing of the firmware and software of Alex and also controls the Arduino, which manages the rest of the components of Alex. Lastly, the LIDAR sits at the very top to scan a 360 degree view of its surroundings.

Section 5: Firmware Design

This section discusses the communication protocols between the Arduino and the RPi. The algorithm mentioned in this section elaborates on how the operator is able to send commands to the Arduino via the RPi and for the Arduino to acknowledge the command. The high level algorithm on the Arduino whenever a command is sent is as stated below:

(1) The connection between the RPi and Arduino is made via a Universal Serial Bus (USB) cable. This cable supplies the power for the Arduino to start up and is also used to upload the firmware program required to run the motors and various sensors connected to the Arduino ports. To start the communication between the RPi and the Arduino, the operator runs a program on the RPi dedicated to taking inputs from the operator and passing them to the Arduino. When this program is executed, the RPi sends a hello packet to the Arduino. Upon receiving the hello packet, the Arduino replies by sending an ‘ok’ packet back to the RPi. The RPi then informs the operator that the connection between the RPi and Arduino is successful. Thus, the operator can begin to input relevant commands to move Alex or retrieve information from the respective sensors.

(2) When the operator inputs any commands into the RPi terminal, the RPi creates a command packet containing the inputs made by the operator. This packet is then serialised and sent to the Arduino via the serial port.

(3) Upon receiving the serialised data from the RPi, the Arduino first checks whether the packet is complete. If the packet is incomplete, the Arduino returns a 'packet incomplete' result; if not, the packet is deserialised. During deserialisation, the Arduino checks the magic number in the packet, and if it does not correspond to one of the accepted numbers, it returns a 'bad packet' result. The Arduino also checks if the checksum it has derived from the data received matches the checksum originally from the received packet. If the checksums do not match, the Arduino returns a 'bad checksum packet' result. Any of the aforementioned errors causes the Arduino to send an error packet to the RPi to notify the operator that the input command is wrong or has been corrupted. If there are no errors, the Arduino creates a packet to store the received data in order to be handled. When the command is ready to be handled by the program, the Arduino sends an 'ok' packet back to the RPi in order to inform that it has received the command and will be executing it.

(4) Once the 'ok' packet has been sent from the Arduino to the RPi, the Arduino executes the commands. For Alex, various keyboard keys have been assigned to the different movement and sensor functions. For standard movement, the operator can utilise the 'WASD' keyboard combination to move Alex up, left, down and right respectively for a predetermined distance (5 cm) and turning angle (5 degrees). With these keys, the operator does not have to input anything further in order to get Alex to move. If the operator would like to move or turn Alex outside the predetermined values, the keys 'f' for forward, 'b' for reverse, 'l' for left turn and 'r' for right turn can be used. These keys have to be accompanied by the amount of distance or turning angle followed by the amount of power to be used. Additionally, the 'x' key can be used to stop Alex on the spot, the 'g' key can be used to obtain data from the wheel encoders which will inform the operator how far Alex has travelled, and the 'c' key will clear all the wheel encoder data to 0. As for the additional sensors, the 'z' key can be used to display the RGB values of the object in front of Alex, the object's distance to Alex and the presence of sound, flame and combustible gases in the environment. Once the movement command has been carried out, the Arduino does not send any further information to the RPi. On the other hand, if the command is to get the wheel encoder data or the secondary sensor data, the Arduino will create a packet to hold the data which will be sent to the RPi.

(5) To send the collected data to the RPi, the Arduino serialises the packet before it is sent to the RPi via the serial port. Upon receiving the serialised data, the RPi performs similar steps as the Arduino does in Step (3) where it confirms the accuracy of the data by deriving the checksum before deserialising the data into a proper packet that contains the

data structure holding the data received from the Arduino. This packet is then printed on the terminal of the RPi for the operator to take note.

(6) The operator can then repeat steps 2 to 5 to input any commands required for the task at hand. The process continues as a loop until the exit flag has been triggered, which closes the connection between the RPi and the Arduino.

Section 6: Software Design

This section provides a detailed description of the different functionalities used within the RPi namely the teleoperation, LIDAR and Colour Detection.

Teleoperation

In order to connect to Alex remotely, the operator will have to be on the same WiFi network that the RPi on Alex is connected to. Once connected to the same WiFi network, the operator can then use the Secure Shell (SSH) Protocol to connect to the RPi terminal and remotely control Alex from a remote machine. On the remote machine, the operator can access and execute the codes required for Alex's functionalities such as its movement controls and to start up the LIDAR and Robotic Operating System (ROS) packages available on the RPi.

LIDAR, ROS and Hector SLAM

The LIDAR is one of the most important features on Alex as it scans and captures the surrounding environment. Data captured from the scans is relayed back to the remote computer. The data is then used to form a map of the surroundings. The transfer of information between Alex and the operator works mainly via ROS which uses a message broadcast and subscription model for communication between machines. This model works by using nodes, which are ROS packages that are able to generate or take in messages shared with other nodes. In order to visualise Alex's surroundings, the operator should follow the steps listed below:

(1) Launch 'roscore' from the master machine which is the operator's remote machine that has established a connection to RPi via SSH. This initialises a series of nodes that are required in order for the communication of messages between further individual node packages [5].

(2) Launch 'rplidar.launch' from the RPi. This creates a node on the RPi that is able to send out the data captured from the LIDAR in the form of messages which other nodes can subscribe to.

(3) Launch ‘view_slam.launch’ from the master machine. This launches an RViz node, which is a visualisation package using Hector SLAM, that subscribes to the rplidar node on the RPi. Hector SLAM is a package that is able to draw out a map from the scans made by the LIDAR, thus allowing us to accurately map out the area Alex is in as it is travelling through the area [6].

Colour Detection

To detect colour, we have attached an RGB sensor with IR filter (Adafruit TCS34725) to the Arduino. The sensor uses the Inter-Integrated Circuit (I^2C) protocol via ports A4 and A5, with the Arduino being the master and the sensor being the slave. The sensor uses colour-sensing photodiodes that convert the light detected into analogue readings [7], and the getRGB function from the Adafruit library will convert the analogue readings for each of the red, green and blue components such that they are within the range from 0 to 255.

Ultrasonic Sensor

To find the distance between the front of Alex and the nearest object in front of it, we have attached an ultrasonic sensor to the Arduino. The sensor will fire an ultrasonic pulse for 20 microseconds and measure the time taken for the pulse to return. The distance would then be measured by calculating the time taken for a single trip multiplied by the speed of sound (330m/s). This value would then be relayed to the operator.

Gas, Flame and Sound Sensors

All three of the gas, flame and sound sensors return a digital value of 0 or 1 when the sensor detects the presence of its respective elements above a certain threshold. The value is then converted into a Y (Yes) or N (No) value and relayed to the operator. To adjust the threshold of the sensors, there is a physical potentiometer on each of them to modify its sensitivity.

Section 7: Lessons Learnt and Conclusion

While this project has been a great experience, we have had our fair share of difficulties in reaching our end goal.

Firstly, we had issues with the robot’s hardware and design. The robot had many different components integrated within its chassis with varying mass. Initially, we chose to place the components where it would be the most obvious or convenient on the robot, which caused its center of gravity to be too high and its weight distribution to be front heavy. This oversight has caused the robot to repeatedly tip over when moving at high speeds as it is too unstable at the front. To resolve this problem, we decided to shift the power bank, the heaviest component of our robot, to the back to act as a counterweight. This change has not only solved the problem but also allowed our robot to be more stable as its weight is balanced on both sides equally.

Secondly, we had issues with the location of the ball caster wheel. The initial location of the wheel at the front caused the robot to skew its straight path as the ball is leading the robot. This problem was much easier to solve as all that was needed to be done was to swap the direction of the bottom waffle and top waffle.

Additionally, we also had a few difficulties with regards to the software design. Originally, our colour sensor was unable to function as intended despite using the provided software libraries. After further testing, we realized that the colour sensor we bought was unable to interface with the libraries that we had set up. Thus, a new colour sensor had to be bought and tested, which fortunately for us worked correctly with the provided libraries.

Lastly, when testing SLAM for the first time on our laptops, we found that the SLAM map created was very inaccurate and slow. Through constant debugging, we realized that the issue may be due to an incomplete installation of SLAM on the laptop. Thus, after reinstalling SLAM onto our laptops and the robot, we were able to obtain much clearer and accurate readings on the SLAM map.

Nonetheless, these hurdles have taught us important lessons about system design and problem solving. One takeaway that we have is that we should always consider possible flaws within the design before committing on it. It should be thoroughly thought through so as to account for foreseeable problems that may occur which may lead to drastic modifications. Moreover, we also learnt that sometimes problems are unavoidable, and that the most efficient course of action may be just to start over from scratch to avoid wasting time on further debugging.

In conclusion, our team has overcome many challenges while doing this project, and now, knowing these important takeaways, we will be more vigilant and experienced system thinkers the next time we embark on similar projects.

References

[1] M. Rich and B. Dooley, “Powerful Quake Hits Japan, Evoking a Worrisome Memory,” *The New York Times*, 13-Feb-2021.

[2] J. Kyodo, “Powerful magnitude 7.3 earthquake jolts Tohoku area,” *The Japan Times*, 14-Feb-2021.

[3] “Spot®: Boston Dynamics,” *Spot® | Boston Dynamics*, 2021. [Online]. Available: https://www.bostondynamics.com/spot#id_first. [Accessed: 06-Mar-2021].

[4] “Guardian® S Remote Visual Inspection Robot,” Sarcos Robotics, 10-Feb-2021. [Online]. Available: https://www.sarcos.com/products/guardian-s/. [Accessed: 06-Mar-2021].

[5] L. Walter, “Wiki,” *ros.org*, 05-Sep-2019. [Online]. Available: http://wiki.ros.org/roscore#:~:text=roscore%20is%20a%20collection%20of,launched%20using%20the%20roscore%20command. [Accessed: 08-Apr-2021].

[6] S. Kohlbrecher, “Wiki,” *ros.org*, 17-Apr-2014. [Online]. Available: http://wiki.ros.org/hector_slam. [Accessed: 08-Apr-2021].

[7] “RGB Color Sensor with IR filter and White LED - TCS34725,” *adafruit industries blog RSS*. [Online]. Available: https://www.adafruit.com/product/1334#description. [Accessed: 08-Apr-2021].