CG1112 Engineering Principle and Practice

Semester 2 2019/2020

"Alex to the Rescue"

Design Report

Team: 3-3-1

Table 1: Role Delegation

Name	Student #	Sub-Team	Role
Samuel Paul Christopher	A0200667J	Software/Hardware	Hardware Engineer
Neo Yao Jie Joel	A0199695Y	Hardware	Hardware Engineer
R Ramana	A0197788X	Software	Software Engineer
Matthew Gani	A0204882A	Software	Software Engineer
Gunit Mittal	A0206438E	Software	Software Engineer

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

Figures

Tables

Section 1: Introduction

An average person can go 8-10 days without food, and only a short 3-5 days without water [1]. This essentially renders all rescue efforts a race against time. There are many complications like the perilous environment and dilemmas rescuers face such as how far they are willing to put their own lives at risk as they seek to rescue potential survivors. Tele-operated robots provide a platform to enhance efficiency of these rescue missions, as well as support and protect the rescuers in their efforts to save lives.

Alex, a search and rescue robot has the potential to carry out the following:

1. Map the environment
2. Transmit relevant data of its current position to the user
3. Avoid obstacles and quickly traverse in and out of rooms
4. Be energy efficient in order to last a longer duration

Alex is designed to be able to locate and rescue survivors at high speeds while being conservative in its power consumption, in order to be out in the field for a long time. It is small and portable so that it can go into small and tight areas in which humans cannot. Being tele-operated ensures that an operator can control Alex from a distance and instruct it to carry out specific tasks. The Light Detection and Ranging (LiDAR) system on Alex creates accurate map representations of the environment and can even show the outer wall dimensions to guide the operator.

In the subsequent sections of this report, the review of the state of the art and system architecture, as well as the hardware, firmware and software design of Alex will be elaborated on.

A video overview of Alex is available **here**.

Section 2: Review of State of the Art

While the previous section established the need for a teleoperated robot, Alex, this section will discuss the strengths and weaknesses of two commercial teleoperated robots that have been factored in during the conception of Alex.

Section 2.1: TALON

TALON is an unmanned military robot (refer to Figure A.1 in Appendix A), produced by Foster-Miller [2], and is used in situations such as first response, heavy lifting and rescue missions. Being built with both a high payload-to-weight ratio and a modular design, they are highly customizable with various sensors and components to suit the mission objectives. TALON robot can be controlled by a two-way radio or fibre-optic link, or even via a laptop control unit or a tactical robotic controller [2].

Strengths

TALON is simple to use, fast and sturdy [2]. It is customizable and hence suited for multi-purpose use in various environments. Sensors can be used to detect gas, chemical, radiation and so on, while a variety of weapons can also be attached [2]. The heavy-duty rotating shoulder is used for heavy-lifting [2]. TALON is cost-effective as it costs less ($230, 000) than what is normally spent on a soldier ($850, 000) per year [3].

Weaknesses

TALON relies heavily on constant and stable latency. Any delay due to unstable internet connection could possibly cause it to do wrong things at the wrong time, such as driving into wrong terrain [4]. TALON also uses two lead-acid rechargeable batteries which tend to have a lower capacity and a lower depth of discharge when compared to lithium-ion batteries [5]. These batteries also require frequent maintenance and tend to have lower life-spans, possibly resulting in higher costs in the long run [6].

Section 2.2: SmokeBot

SmokeBot seeks to address limitations the existing robots and possibly rescuers have in environments where sensor technologies and vision can be affected due to dust and smoke [7]. It is built specifically to counter low visibility situations using a range of sensors (thermal, radar, gas) to help firefighters in search and rescue missions (refer to Figure A.2 in Appendix A) [7]. This makes it thrive in situations where humans are not able to do so. It can also detect harmful gases that pose a risk to the lives of firefighters [7].

Strengths

One useful function of SmokeBot is that it can compare its surroundings to pre-existing maps of the area so that it is easy for the operator to traverse in areas that were mapped before [7]. SmokeBot is able to detect the possibility of a gas explosion by using data collected from its sensors [8]. In the case where SmokeBot loses internet connection, it has the ability to automatically traverse back to the last place where it had a connection [8].

Weaknesses

SmokeBot cannot be used in immediate rescue efforts as it takes about 15-30 minutes to collect information [8]. Although it has the ability to automatically traverse back to the last place it had a connection, the approach in the algorithms used can be inefficient and can cause the SmokeBot to skip closer and safe areas with stable internet connection [9].

Section 3: System Architecture

In this section, Figure 1 is used as an overview to show how all of the components like the hardware, firmware and software work together to make sure Alex works properly.

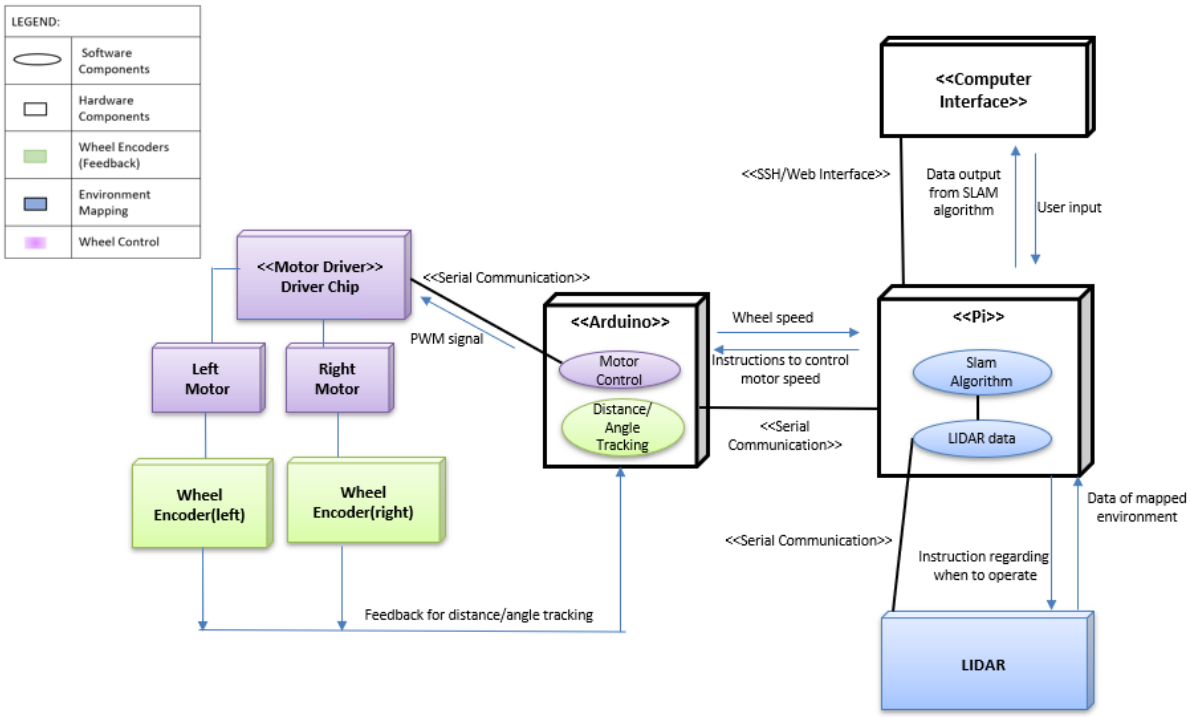

Figure 1: System Architecture

Referring to Figure 1 and the legend above, the ovals represent the software components, and the rectangular boxes represent the hardware components. The arrows show the direction of the flow of data to and from the various hardware components, while the wires represent the relevant communication tools which will be used to transmit the data.

As can be seen from Figure 1, the wheel encoders and magnets are placed on the motors to compute the revolutions and give us the number of ticks (explained in Section 5). The motor is connected to the Arduino which will communicate the motor data (values provided by wheel encoders) to the Pi. Through the Pi, the user can adjust the speed and direction of Alex and this information will then be relayed to the Arduino.

The LiDAR will be directly linked to the Pi and will utilize the Simultaneous Localization And Mapping (SLAM) algorithm to obtain data regarding the environment. This is done by setting up a topic and subscribing to that particular topic. Through this, the data of the surroundings on the Pi will be published to the topic and from the user's PC this data can be viewed.

Section 4: Hardware Design

In this section, the specifics of the hardware design for Alex will be explained. Figure 2 below shows the placement of Alex's components which provided the best possible LiDAR output and even weight distribution.

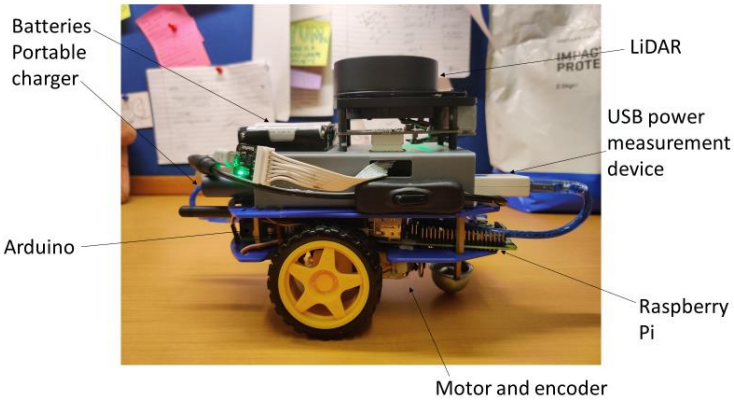

Figure 2: Alex

LiDAR

The LiDAR unit calculates the distance between itself and obstacles/walls by emitting light pulses. These light pulses reflect off its surroundings and back to its detection unit. The LiDAR rotates 360 degrees at a relatively high speed and returns the coordinates of the obstacles it observes relative to its position. This data is then sent to the Pi and a visualization node is used so a map of the surroundings can be generated.

Hardware Design Considerations

The LiDAR has to be in a high enough position so that none of the other components will block its field of view. To overcome this, a 3D printed board is used as an extra layer with holes drilled in it to secure the LiDAR so that Alex would be more stable. This way the LiDAR would be upright and stable ensuring the data recorded is accurate.

The breadboard was inserted at the bottom level of Alex alongside the motors so the wires would be less messy and could go directly to other components like the motors and the Arduino. The Arduino and Pi were placed on the middle level above it and some wires were made to go through the holes in the platform so they were easily accessible to the top and bottom layer.

The top layer of Alex contains the 3D printed board and all sources of energy (portable charger and batteries) so that it would be easier to remove and insert batteries or charge the portable charger. There are also connecting wires going from the top level to the middle level to power the Arduino and the Pi.

Section 5: Firmware design

The Arduino and the Pi are essential components of Alex akin to the brain. The Arduino controls the functions that Alex has, such as the actual movement and turning since it is directly connected to the motors. While the Pi, transmits and receives data. This section explains the thought process behind Alex, outlining how Alex communicates using the Pi and the Arduino in synchrony and explains how the Arduino calculates distances travelled.

Receival of Command Packets

The Pi sends commands over to the Arduino in the form of packets via serial communication. The necessary steps to initialize serial communication are detailed below.

1. Test communication between Pi, Arduino and Laptop
    1. Ensure to set the PORT_NAME macro to the port where the Arduino is connected.
    2. Proceed to create a serial connection at 9600 bits per second (bps).
    3. Pi creates a packet with the message “Ready?”.
    4. Pi compiles the message and serializes the Protocol Packet as a structure into a stream of bytes to Arduino.
    5. Arduino responds with an ACK package to Pi.
    6. The Arduino will then send back a packet to inform the Pi that it is done.
    7. The Pi implements a Checksum to check that the data received is correct. The Pi will compare the Checksum from the sender side against an attached Checksum. If the data received matches correctly, the Pi will reply to the Arduino with an ACK packet. Otherwise, a NAK packet is sent instead.
    8. The Pi will deserialize the stream of bytes received by the Arduino back to structures, which is readable by the user.
    9. The user will receive an output message from Arduino titled “Yes”.
2. Test communication between Pi and Laptop (for ROS)
    1. Ensure the Pi and the laptop are connected to the same network
    2. Start a test program on the Pi that outputs messages to a topic
    3. Ensure that the right messages are received on the Laptop’s end that is listening for the output messages at the topic.

On the Arduino, functions like readPacket and readSerial are used to deserialize and understand what has been input into the Pi. The sendResponse function is used to take a packet, serialize it and send it out over the serial port back to the Pi.

Functions in the Arduino code such as sendBadChecksum, sendBadCommand and sendBadResponse are used so as to debug and check for errors and if the commands sent to the Arduino from the Pi are being read correctly.

When the packet is received and can be read properly, the sendOK function is called to tell the user, the message is read successfully. The sendStatus function tells the user the number of ticks in each direction, in the respective motors, and the forward and backward distance that is being travelled.

Movement of Alex Facilitated by Arduino

As mentioned in Section 1, in order for Alex to know that it has travelled the distance that was inputted by the user, it continuously calculates the distance it travelled and compares it to the value inputted. When the values are equal, Alex stops moving. A step-by-step explanation of how distance travelled is calculated is outlined below.

1. Magnets are attached to the motors. When the wheel completes one revolution, the encoders detect the falling edge at the end of the cycle, and causes an interrupt through the ISR.
2. For each revolution of the wheel, the interrupt records the respective direction (left/right) tick to increase by a certain amount.
3. Distance covered by one revolution of the wheel is recorded as WHEEL_CIRC cm angular distance.
4. When the forward function is called, distance travelled can be calculated using the following formula

$$\frac{(WHEEL\_CIRC \times total\ number\ of\ ticks)}{number\ of\ ticks\ for\ one\ wheel\ revolution}$$

In the case of turning left or right, a function called computeDeltaTicks is called, which computes an estimate of the number of wheel encoder ticks needed to turn Alex by the angle (in degrees) that was inputted by the user. When Alex is moving in a straight line, the distance covered is WHEEL_CIRC cm forward (or backward) in one-wheel revolution. To turn by x degrees, the number of wheel turns is given by the following formula, where AlexCirc is the circumference of the circle made by Alex turning on a dime.

$$\frac{ang}{360.0} \times \frac{AlexCIRC}{WHEEL\_CIRC}$$

The number of ticks is given by the following formula, where COUNTS_PER_REV refers to the number of ticks required for 1 full revolution of the wheel.

$$\frac{ang}{360.0} \times \frac{AlexCIRC}{WHEEL\_CIRC} \times COUNTS\_PER\_REV$$

Travelling Speed of Alex

To be able to select a suitable (and the fastest) speed at which Alex should travel, the presence of latency is taken into consideration. Latency is the delay before the transfer of data begins after an instruction has been sent by the sender. This would mean that there might be several possibilities that obstacles might be sensed by Alex, but the data not being able to reach the receiver in certain short periods of time due to latency issues. Collisions might occur if Alex is travelling at a speed faster than the latency period. Therefore, to obtain a good balance between the accuracy of the sensors and the speed travelled by Alex, test runs have been implemented so that to learn how to best control Alex.

Transport Layer Security (TLS)

A secure channel for communication between Alex's Pi and Arduino was created using TLS, as such the user now can control Alex without using VNC Viewer. In order to do this, a certificate authority was created to allow Pi and Arduino to authenticate messages that have been transmitted. Lastly, these transmitted messages are encrypted using symmetric cipher after the handshake protocol has taken place between the Pi and Arduino.

Bare Metal

Motors

Using Timer 0 and Timer 1, four PWM signals (in Phase Correct PWM mode) are generated where each signal is supplied to one of the terminals of the wheels. By setting the OCRxx value, the duty cycle of the PWM and hence the speed of Alex is altered. The formula for calculating the duty cycle of PWM signal is shown below.

$$Duty\ Cycle = \left(\frac{OCRxx}{255}\right) * 100\%$$

Serial Communication

For this portion, a frame format of 8 bits with no parity checking and 1 stop bit was used with a baud rate of 9600 bps. The team adopted interrupts version of the implementation for UART, as such USART_RX_vect is triggered when data is received and USART_UDRE_vect is triggered when the data register is empty.

Section 6: Software Design

The following section explains how the software aids in generating the map of the location, and how Alex is going to be controlled remotely via the user's laptop.

Processing User Input

On the Pi, there is a Graphical User Interface (GUI) for user to input actions (as shown in Figure 3).

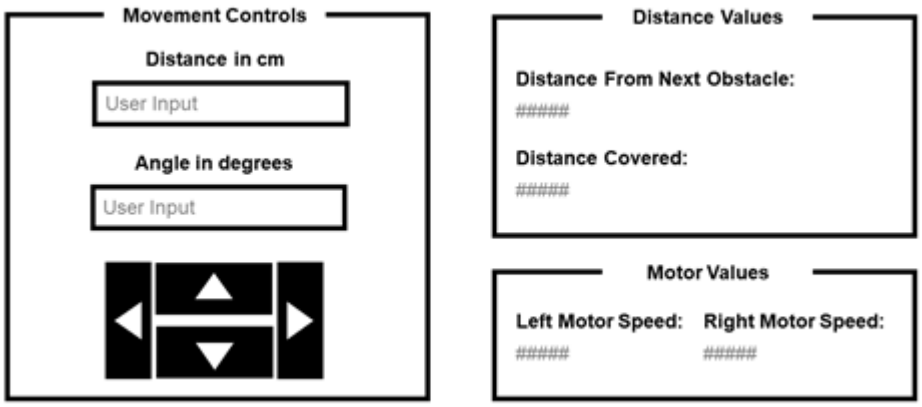

Figure 3: GUI Layout

The GUI allows easier and more intuitive usage for the user. The user will be able to control the robot's direction, distance travelled as well as the angle of movement. The mapping and processing work have also been offloaded to the user's CPU which will allow for a more efficient energy usage of the Pi's CPU.

NodeJS was used for the back-end of the GUI while the front-end was made using HTML5 and CSS3. Sockets programming in JavaScript was used to allow the GUI components to emit socket events when interacted with. Once an event is emitted by the front-end, the server will pick up on that event and send the appropriate command to Alex's server running on the Pi.

Furthermore, the user can control Alex using the arrow keys. The conio.h header file is used to map the keys pressed on the keyboard to different movement commands. Specifically, the four arrow keys are initialized to cause movement in the four directions, and the other keys (such as 'g' to get wheel encoder values and 's' to stop the motors) to cause the respective processes. For example, if the reverse key is pressed and held by the user, Alex continues to move backwards at a fixed speed. When users make invalid keypresses, "BAD COMMAND" is output to notify the user that an invalid keypress was made.

Offloading from Pi to Laptop

Figure 4 will be used to explain how processing of Hector SLAM is offloaded from the Pi to a laptop.

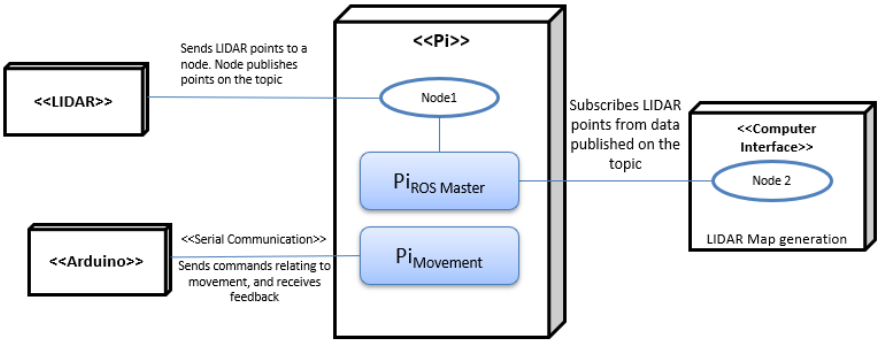

Figure 4: Map Generation Using ROS Master

As seen in Figure 4 above, the Ros Master is the Pi which runs ROSCORE and acts as the hub for the topics (the ROS_MASTER_URI on the computer and the Pi are set to the IP address of the Pi). There are two nodes, Node 1 which runs on the Pi and serves to publishes the LiDAR points to a topic and Node 2 which runs on the computer to run Hector SLAM and the visualization of the Map by using the points published by Node 1.

Live Mapping

SLAM, is a collection of algorithms that enables the construction and mapping of an unknown environment, all while allowing the user to monitor the robot's position. SLAM is always used with several types of sensors. In this case, coupled with the infrared laser signals, the LiDAR emits laser signals in a circular manner around the surroundings. The laser bounces from the various obstacles and objects in the environment and is received back by the LiDAR to go through processing. The data collected goes through a Digital Signal Process (DSP), which produces a graphical representation of the simulated environment. This visualization process provides an accurate depiction of the surrounding environment, making it easier for the user to control Alex that is traversing the terrain while avoiding any collisions.

Acknowledging the perks of SLAM, the SLAM combined with the visualization process consumes approximately, a hefty 90% of the CPU load. This is clearly a major downside as the large energy consumption will cause the battery to drain quickly and Alex's energy consumption will be inefficient.

In order for the Map to be generated based on the LiDAR data, the SLAM algorithm from the Pi will be offloaded to a laptop as explained above. This reduces CPU usage on the Pi and allows for a higher refresh rate. This will also free up the Pi's CPU usage as data points rather than images are transferred from the Pi to the user's laptop so that it can process our movement commands quickly and properly.

A list of steps taken when mapping is started and updated is shown below.

For the first instance,

1. Pi triggers LiDAR to take the first set of readings.
2. The LiDAR takes readings and sends it to the Pi
3. LiDAR data is transmitted to the laptop using a ROS node
4. Laptop runs HECTOR SLAM algorithm to generate the map

For updating of the map,

1. Pi triggers LiDAR to take the readings
2. Pi publishes LiDAR readings to a Pi topic
3. Laptop is running a ROS visualization node that is subscribed to the same topic that the LiDAR data points are published to
4. Laptop runs HECTOR SLAM algorithm to generate the map for user

Further features of the map allow the user to measure the distances of walls so that it will be easier for the user to gauge how far the bot is from the wall instead of a mere estimation. Additionally, a zero-degree marker informs the user of the direction Alex is facing.

Two maps will be processed, one being the SLAM output and the other will be a graph of LiDAR points to see the environment better. The dots will be in red so that they stand out to ensure that the points are clear and to help the user understand Alex's surroundings better.

Power Savings

Power consumption is a chief area of consideration when designing Alex. Arduino has a built-in sleep function that allows the user to stop or turn off any unused modules in the Microcontroller, which in turn significantly reduces the power consumption. Together with interrupts, the function removes the need to continuously poll for instructions, enabling the Arduino to be more efficient in executing tasks.

Adding on to this, the number of cores activated on the Pi have been reduced to two active core, the maximum clock speed the Pi's CPU has been decreased as power consumption is proportional to the maximum clock speed, the minimum clock speed of the Pi's CPU has been decreased and functions like the Bluetooth, PWR LEDs and ACT LEDs have been disabled on the Pi. Excerpts of all commands for implementing these power reduction features are listed in Appendix B.

Table 2 below shows how the current drawn by Alex changes with some of the power reduction measures implemented.

Table 2: Change in Current Drawn

Power reduction measure	Current drawn before	Current drawn after	Change in current drawn
Bluetooth disabled	0.55 amps	0.53 amps	0.02 amps
Number of active cores changed from 4 to 2	1.14 amps	0.99 amps	0.15 amps

Section 7: Conclusions & Lessons Learnt

This project has allowed us to consolidate and make use of what we have learnt from the studio sessions to build a functioning search and rescue robot. However, it was not always smooth sailing, and this section explains some of the issues that were faced while building Alex and how these challenges were successfully overcame.

Challenge 1 - Weight Distribution

The effect of weight distribution on Alex was initially underestimated. Originally, the portable charger was placed towards the right side of the topmost platform, with the LiDAR unit resting on top of it. This meant that when Alex underwent test runs to move forward in a straight line, Alex would veer off to one side as more of its weight was resting on the right wheel. A large portion of the time was wasted trying to recalibrate the motors through the code and that was valuable time that could have been spent on further enhancing Alex's capabilities.

Solution

Realizing that it was the weight distribution that was causing Alex to not move straight, and veer off, it was decided to re-arrange the hardware set-up on Alex and ensure that the weight was more evenly distributed by moving the LIDAR, portable charger and batteries to the middle. This resulted in Alex being able to move straight ensuring the operator could control Alex better without having to alter the code.

Challenge 2 - Role Delegation and Communication

In the beginning of the project, tasks were set verbally based on what had to be accomplished by the week and every member just focused on the same parts of Alex. It did not help that our tasks were vaguely written (for example working on bare metal or calibration of Alex). This turned out to be inefficient as a lack of communication meant that many of the tasks would overlap.

Solution

Eventually, it was decided upon to distribute work, assigning specific tasks and Trello was used to ensure accountability and kept to the tasks on hand and met then deadlines. As the project was affected by the COVID-19 pandemic, social distancing had to be maintained. Communication skills and working online became more salient, with online team meetings every few days to discuss our priorities, update on our tasks and ensuring we were on time to fulfilling the tasks.

References

[1] J. Castro, "How Long Can You Survive Under Earthquake Debris?," NBCnews.com, 26 October 2011. [Online]. Available: http://www.nbcnews.com/id/45053108/ns/technology_and_science-science/t/how-long-can-you-survive-under-earthquake-debris/. [Accessed 10 April 2020].

[2] Army Technology, "Army Technology," [Online]. Available: https://www.army-technology.com/projects/talon-tracked-military-robot/. [Accessed 6 March 2020].

[3] A. Etzioni and O. Etzioni, "Army University Press," May 2017. [Online]. Available: https://www.armyupress.army.mil/Journals/Military-Review/English-Edition-Archives/May-June-2017/Pros-and-Cons-of-Autonomous-Weapons-Systems/. [Accessed 6 March 2020].

[4] D. Quick, "News Atlas," 3 August 2011. [Online]. Available: https://newatlas.com/long-distance-tele-operation-of-ugvs/19423/. [Accessed 6 March 2020].

[5] J. Marsh, "EnergySage," 10 January 2019. [Online]. Available: https://news.energysage.com/lithium-ion-vs-lead-acid-batteries/. [Accessed 6 March 2020].

[6] P. Taylor-Parker, "Wholesale Solar," 23 October 2018. [Online]. Available: https://www.wholesalesolar.com/blog/lead-acid-vs-lithium-batteries. [Accessed 6 March 2020].

[7] CORDIS EU Research Results, "CORDIS EU Research Results," 5 December 2018. [Online]. Available: https://cordis.europa.eu/article/id/241251-stateoftheart-robot-to-better-assist-search-and-rescue-operations-in-lowvisibility-conditions. [Accessed 6 March 2020].

[8] Örebro Universitet, "Phys.org," 19 June 2018. [Online]. Available: https://phys.org/news/2018-06-smokebot-robot.html. [Accessed 6 March 2020].

[9] P. Fritsche, B. Zeise and P. Hemme, December 2017. [Online]. Available: https://www.smokebot.eu/deliverables/SmokeBot_D5.3.pdf. [Accessed 6 March 2020].

Appendix A

Figure A.1: TALON
Source: Adapted from [2]

Figure A.2: SmokeBot
Source: Adapted from [7]

Appendix B

// To limit Pi to 2 cores, add in /boot/cmdline.txt
maxcpus=2 to the end of the line in

// To change the minimum clock speed, add in /boot/config.txt
arm_freq_min=100 // (min freq in MHz)

// To change the maximum clock speed, add in /boot/config.txt
arm_freq = 900 // (max freq in MHz)

// To disable Bluetooth add in the /boot/config.txt
dtoverlay=pi3-disable-bt

sudo systemct1 disable hciuart.service
sudo systemct1 disable bluealsa.service
sudo systemct1 disable bluetooth.service //disable related services

sudo reboot // to apply changes

// disable ACT LEDs
dtparam=act_led_trigger=none
dtparam=act_led_activelow=off

// disable PWR LEDs
dtparam=pwr_led_trigger=none
dtparam=pwr_led_activelow=off