

**CG1112 Engineering Principle and Practice**

Semester 2 2019/2020

**“Alex to the Rescue”**

**Final Report**

**Team: 04-02-01**

|  |  |  |
| --- | --- | --- |
| Name | Student # | Main Role |
| Wang Zihao | A0204706M | Member |
| Vo Quang Hung | A0200697A | Leader |
| Thng Yu Xuan | A0200065Y | Member |
| Xu Ziyi | A0204675A | Member |
| Tra Quang Minh Thong | A0200659H | Member |
|  |  |  |

**Link of Video Demonstration of Project Alex**

[**https://www.youtube.com/watch?v=hj4xA06EfTM**](https://www.youtube.com/watch?v=hj4xA06EfTM)

**Purpose of Report**

This report describes the functionalities of Alex, analyzes two remote search and rescue robotic platforms, explains the hardware, firmware and software designs of Alex, and finally suggests lessons learnt from this project.

**Section 1 Introduction**

In this project, Alex is designed to reproduce the layout of an unknown maze in order to conduct search and rescue. It will be remotely controlled from a laptop to produce a map of the environment via a series of operations. Specifically, the layout of the maze and the dimensions of the wall and obstacles will be recorded and noted down. Then, Alex will be navigated manually by the operator with the guidance of the environment map.

There are two basic functionalities implemented on Alex, which are going straight and turning left or right, and they would be used by the operator to control the movement of Alex. In the going straight function, Alex’s speed as well as its distance travelled could be controlled by the operator. In the turning function, the angle of turning could be defined by the operator. Moreover, there is one additional power-saving feature implemented on Alex in order to reduce its power consumption by temporarily turning off the operation systems that are not in use.

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

This section provides a brief overview of two other existing search-and-rescue robots which are designed by other engineers in other parts of the world. It describes the basic functions of the robots, as well as their strengths and weaknesses. The first robot is the RAPOSA, and the second robot is the SpotMini.

RAPOSA is a search-and-rescue robot that is designed by the Institute for Systems and Robotics, Lisbon [1]. The robot is tele-operated and controlled by a human operator through a Graphical User Interface (GUI), which allows the user to drive the robot, configure the sensors, and monitor low-level microcontroller data tables [2]. The robot is controlled using a gamepad [1]. RAPOSA uses gas, infrared, temperature, and humidity sensors [1]. It also has 3 conventional cameras and 1 thermal camera to provide a better field of view for the operator [1]. Low consumption LEDs are installed near the cameras for artificial illumination in dark areas [2].

The specialty of this robot is its tethering function. It can be tethered or untethered at any point in time when necessary. This allows the robot to be able to be pulled out and establish stable power and communication [2].

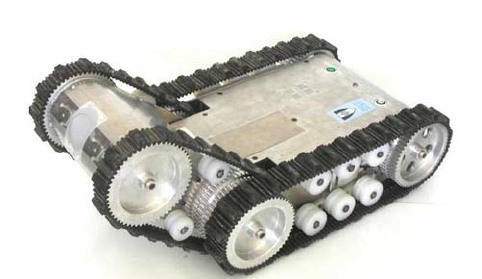


Fig. 1 An external view of the RAPOSA [2]

When the RAPOSA was tested in scenarios by the Lisbon Firefighters School, the robot was able to traverse across rubble successfully, and climb up and down stairs [1]. It was also able to detect a victim trapped inside a pipe via its infrared sensor [2]. It was also proven that the robot reduced inspection time by 25% during these scenarios, compared to the time taken by experienced firefighters [2].

However, a problem it faced was with the wireless communication, which was due to the location of the antennas on the robot, and also interference with other existing wireless networks [2].

Next, the robot SpotMini will be introduced. Made by Boston Dynamics USA, SpotMini is a versatile robot dog that can be used for many kinds of jobs, such as construction, search-and-rescue, and public safety [4]. It has sensors that include stereo cameras, depth cameras, Inertial Measurement Unit (IMU), and position sensors, which aid in navigation and manipulation [3]. It is remotely controlled and uses 3D vision with SLAM and obstacle avoidance [3].



Fig. 2 SpotMini [3]

More importantly, SpotMini is very powerful and able to carry payloads more than normal drones. For instance, a few SpotMinis together are able to pull a large truck [5]. Moreover, SpotMini is nimble as it is not a wheeled robot. Hence it is able to climb up and down obstacles, which makes it more capable to traverse through rough terrain [3]. However, SpotMini is expensive and costs “as much as a luxury car” [6].

Overall, it is incontrovertible that from the case studies shown in this section, Alex has similar functions despite having many differences. Alex is able to be remotely controlled and able to map out its surroundings as it maneuvers around a foreign environment. With a larger budget and greater backing, Alex has the potential to be implemented with more advanced features, such as more sensors like the ones on RAPOSA, or the strength of the SpotMini.

**Section 3 System Architecture**

The system for Alex contains six parts, which are remote desktop, Raspberry Pi (RPi), Arduino (Uno version), LiDAR, motor and wheel encoders. Their relationships are demonstrated as shown below.

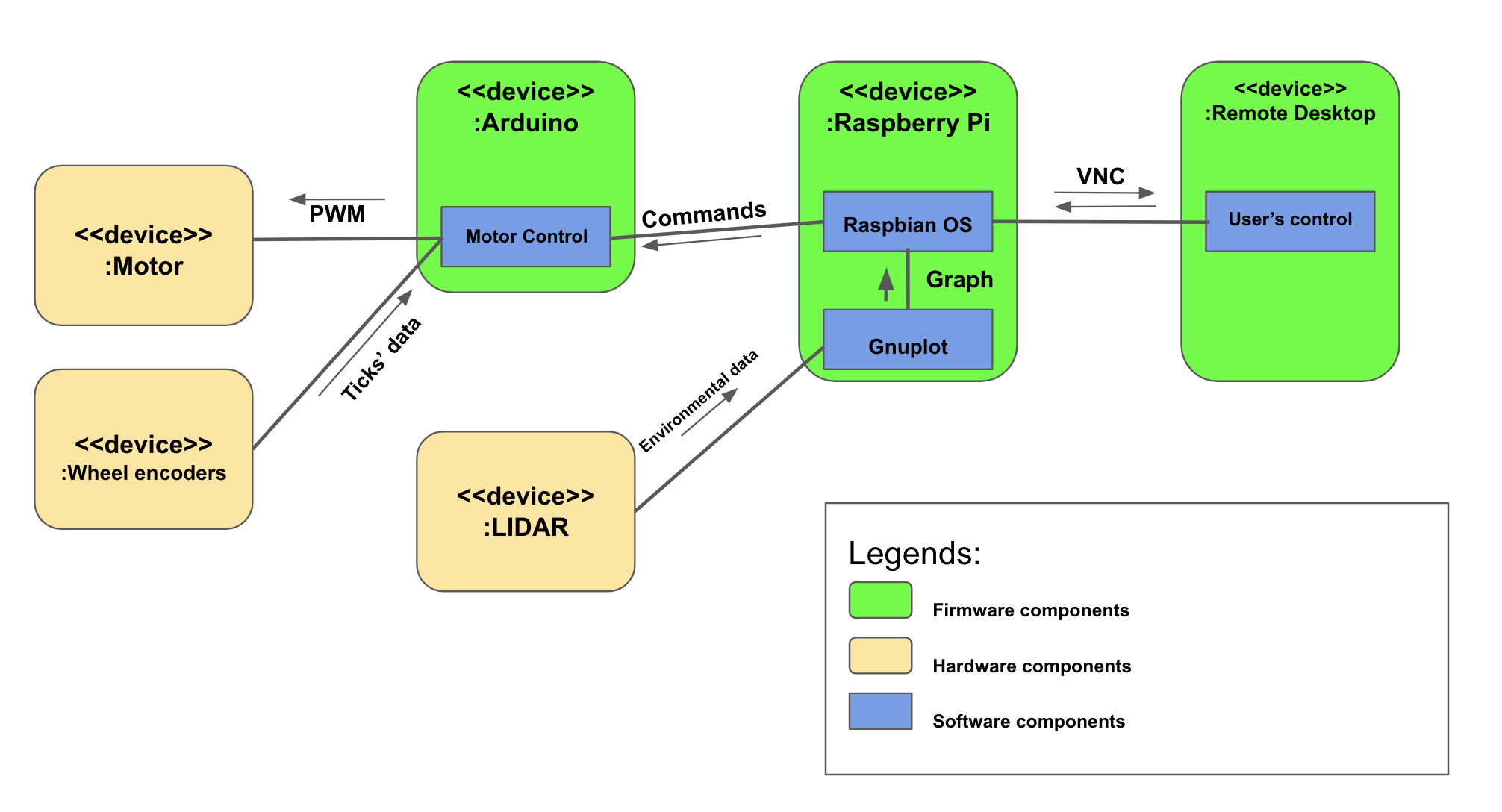


Fig. 3 System Architecture

As seen in Fig. 3, the system for Alex is the combination of hardware, firmware and software designs, in which, the hardware comprises 3 main components, which are LIDAR, DC Motors and Wheel encoders. Meanwhile, the Arduino Uno board, Raspberry Pi and user’s Remote Desktop are firmware that help transfer data between components and send commands from the users to Alex. Besides that, there are 4 main programs, which are the software components of Alex. These programs include algorithms and instructions that assist the telecommunication between the user and the robot.

**Section 4 Hardware Design**

The layout of Alex is in Figure 4 below.

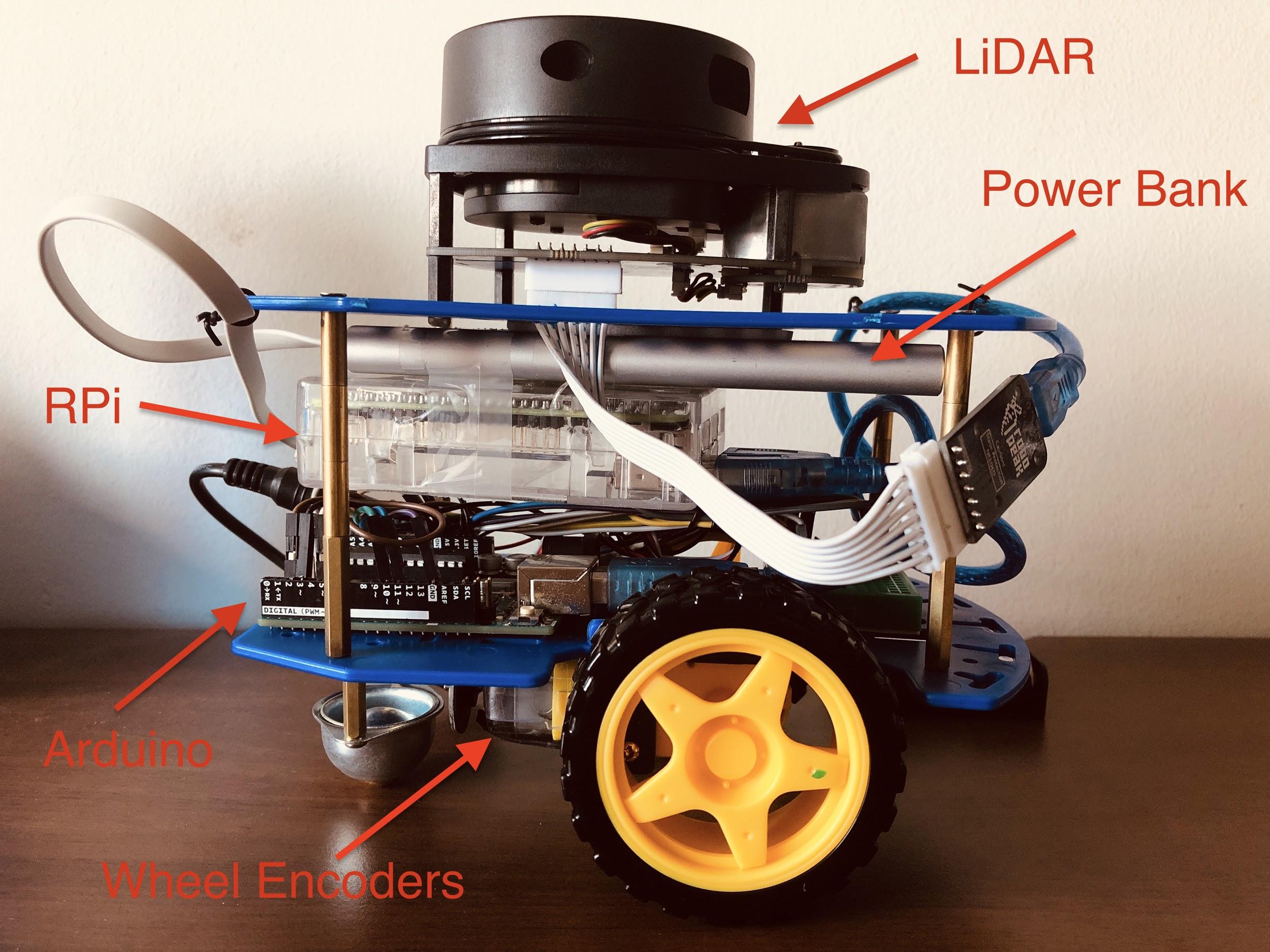


Fig. 4 Hardware Design

As seen in Fig. 4, the hardware of Alex contains six parts, including wheels, wheel encoders, LiDAR, RPi, Arduino and a power bank.

**Section 5 Firmware Design**

In the beginning, when the user turns on Alex and starts the program, the communication protocol between the Arduino and the RPi is set up. Through the communication protocol, the RPi and Arduino can send and receive serialized data from each other. When the Arduino receives data, it processes to deserialize it and assign a suitable PWM value to the motors. The motors then start to spin in the desired direction and this process repeats multiple times until the user decides to end the program.

Particularly, when the program starts, the RPi will send a hello packet, namely PACKET\_TYPE\_HELLO, to the Arduino. Subsequently, the Arduino will deserialize the packet and acknowledge it. If the packet is good, it will send an OK message back to the RPi; otherwise, it will notify the RPi and the RPi will print out the error to the user’s screen.

When the setup is done, the user starts to send commands to control the motors’ movement, which includes moving forward or backward, and turning left or right. For each command, the user will enter the direction together with the requirement for that command. For example, if the command is ‘f” or ‘b’, which stands for ‘forward’ and ‘backward’, the user will then enter the travel distance in centimeters and power use in percentage. If the command is ‘l’ or ‘r’, which stands for ‘left’ or ‘right’, the user will enter the angle of turning in degrees and power use in percentage. For each command, the Arduino will reply with a ‘good’ message if the command is good; otherwise, it will send back a bad message, which requires the user to re-enter another appropriate command.

**Section 6 Software Design**

This section will introduce the software designs of Alex, which are teleoperation on Alex and power management on Alex.

1. Teleoperation on Alex:

In the project, Alex can be remotely controlled by Virtual Network Computing (VNC) and Secure Shell (SSH) Communication. Since LiDAR takes in the data of the surroundings and the RPi uses GNUplot to display data on the screen, VNC can be used to visualize the screen of the RPi remotely. Therefore, the researcher can visualize the map displayed by the RPi. Besides that, SSH is used on the user's computer to send commands over to the RPi. Subsequently, these commands will be sent to Arduino to control Alex’s motors. Therefore, with the combination of VNC and SSH Communication, the users can do teleoperation on Alex and achieve the purpose of search and rescue.

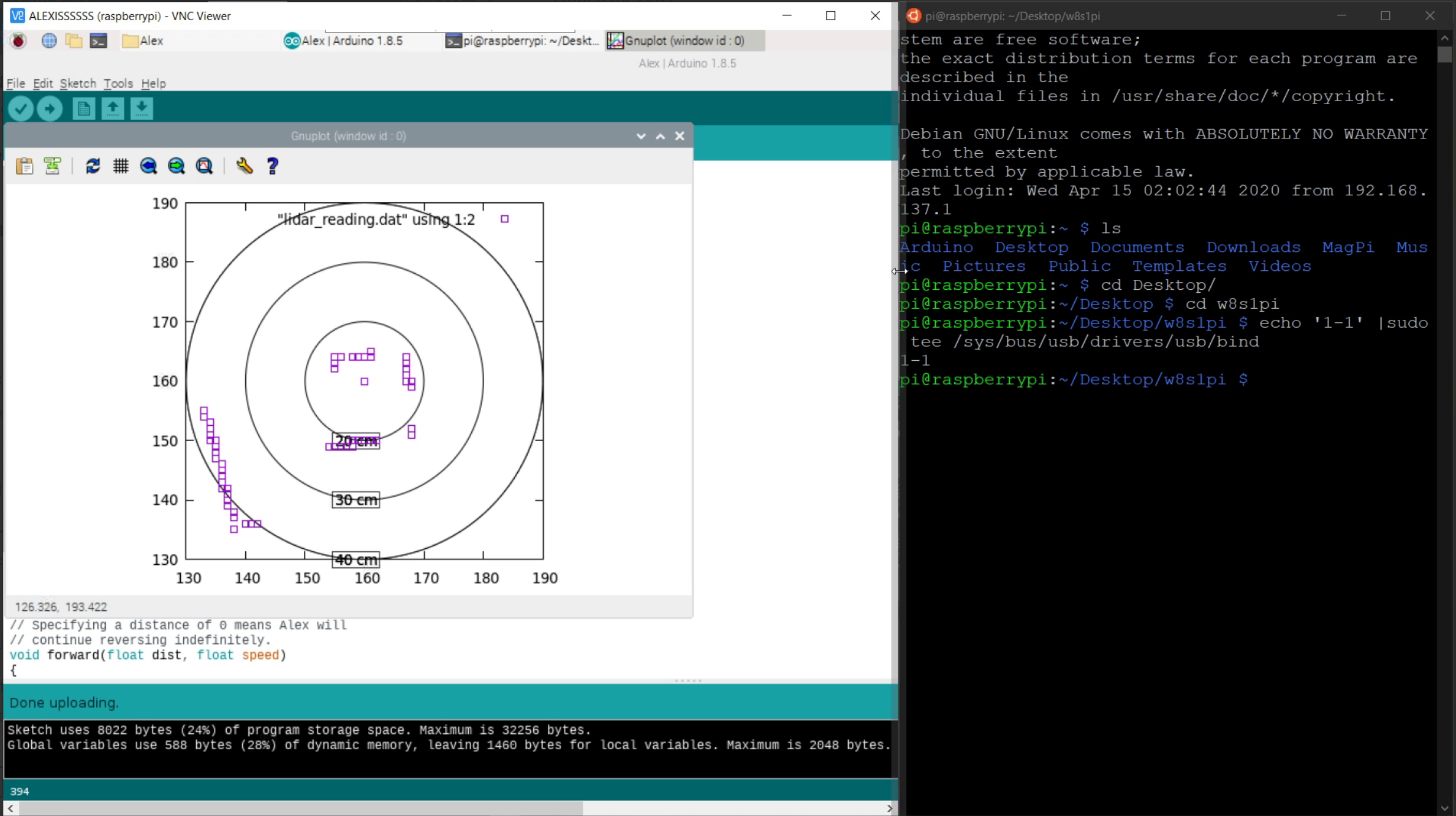


Fig. 5 VNC interface with GNUplot display (on the left) and SSH communication (on the right)

1. Power Management on Alex:

In order for Alex to function properly, power saving must be taken into consideration. Therefore, unused subsystems in the Raspberry Pi, Arduino and Lidar need to be shut down temporarily to reduce power consumption. The power management is implemented on three different components of Alex: the RPi, the Arduino and the LiDAR.

1. Power management on RPi:

Initially, when the RPi is not connected to the Arduino and the LiDAR, the current drawn from the battery to operate the RPi is 0.43A. Since the High-Definition Multimedia Interface (HDMI) port on RPi is not needed in this project, it is advisable that this port be turned off. Hence, this command is used to turn off the HDMI port on booting RPi: “/usr/bin/tvservice -p”. After using this command, the current drawn to operate RPi is reduced to 0.41A.

Additionally, the researcher uses WIFI that is initially set up on RPi to make connection to this hardware component. Therefore, the ethernet port should also be disable on booting RPi. In the Desktop folder, an additional Linux module called ‘hub-ctrl’ is installed by the command: “sudo ./hub-ctrl -h 0 -P 1 -p 0”, which ‘-h’ stands for hub, ‘-P’ stands for port, and ‘-p’ stands for power. The current drawn to operate the RPi is then further reduced to 0.36A, which implies that the energy consumed by RPi is also reduced.

1. Power management on Arduino:

Besides RPi, Arduino also consumes a great amount of power. Therefore, the following features should be configured to achieve our purpose of reducing power consumption on Arduino.

Firstly, the Power Reduction Register (PRR) is an 8-bit binary sequence that controls several extensions. It does not play any role in our project, but still consumes power. Therefore, a more optimized power consumption can be achieved by turning off these unnecessary features. Secondly, some configurations should also be made on the Sleep Mode Control Register (SMCR), which is an 8-bit binary sequence that determines several power modes of the Micro Computer Unit (MCU). Finally, the ADCSRA (Analog-to-digital Conversion Status Register A) is an 8-bit binary sequence that controls Analog-to-digital Conversion protocol, which does not have any significant role in the project.

The following piece of code is prepared for further configuration of the above registers.

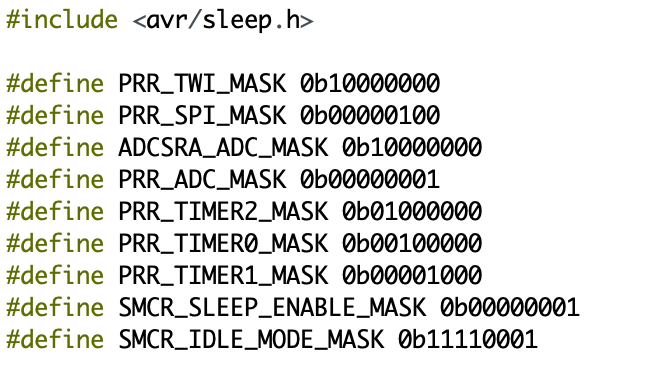


Fig. 6 Bit-mask definitions

Figure 6 shows the set of binary sequences constants that will be used later to set up the power saving features for the robot.

Legends and explanation of the use of above constant binary sequence:

* PRR\_TWI\_MASK: set the two wire interface (TWI) bit of PRR to 1 to disable it.
* PRR\_SPI\_MASK: set SPI bit of PRR to 1, hence disable the Timer/Counter2 module.
* ADCSRA\_ADC\_MASK: write 1 on bit 7 of ADCSRA, hence disable ADC.
* PRR\_ADC\_MASK: set the last bit of PRR to 1, hence shut down ADC.
* PRR\_TIMER2\_MASK: write 1 on bit 6 of PRR, hence disable the Timer/Counter2 module.
* PRR\_TIMER0\_MASK: write 1 on bit 5 of PRR, hence disable Timer/Counter0 module.
* PRR\_TIMER1\_MASK: write 1 on bit 4 of PRR, hence disable the Timer/Counter1 module.
* SMCR\_SLEEP\_ENABLE\_MASK: write 1 on the last bit of SMCR, hence allow MCU only goes to sleep mode when SLEEP instruction is executed.
* SMCR\_IDLE\_MODE\_MASK: set bit 1 to bit 3 of SMCR to let MCU into the idle sleep mode.

Firstly, the Watchdog Timer should be turned off. Watchdog Timer (WDT) is used as a timer counting cycles for other components that have no relevance to our project. However, in Alex, this extension seems redundant and needs to be turned off by proper setting and assignments of controlling bit sequences as shown from the code below.

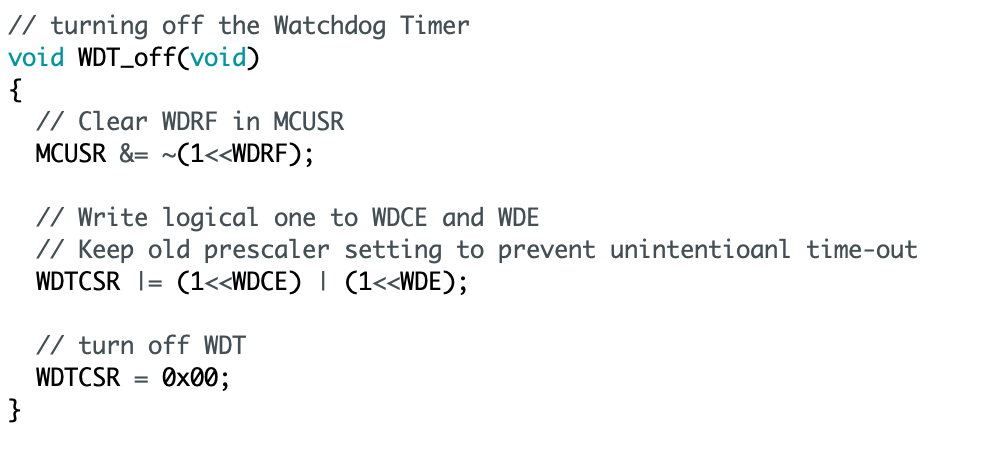


Fig. 7 Turning off Watchdog Timer

Furthermore, unused modules such as TWI, SPI and ADC should be disabled. SMCR is also set to choose the idle sleep mode on Arduino.

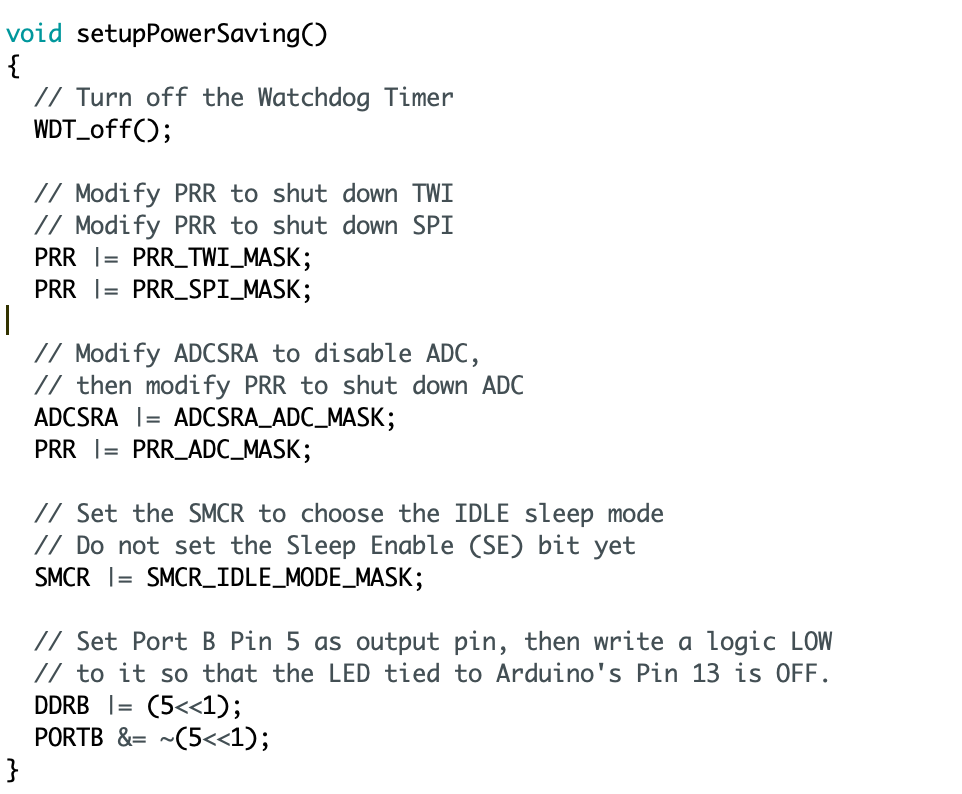


Fig. 8 Setting up Power Saving

Finally, Arduino should be put into idle mode when the motors are stopped, and the Arduino does not receive any commands from RPi.

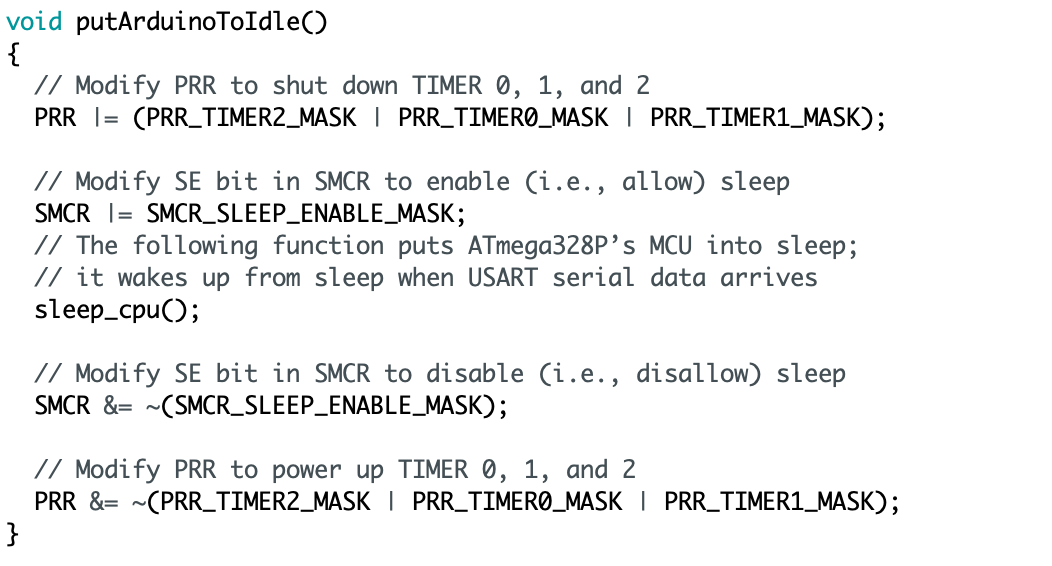


Fig. 9 Putting Arduino to Idle

Figure 8 and Figure 9 show the coding section for two functions setupPowerSaving() and putArduinoToIdle() functions which disable unnecessary features that might waste a high amount of power.

1. Power management on LiDAR:

The LiDAR operates on three modes in total. When the LiDAR totally stops spinning and scanning, the current drawn by the LiDAR is 0.18A. When the LiDAR is spinning but not scanning, the current drawn is 0.26A. Finally, when the LiDAR is spinning and scanning, the current drawn is 0.38A. Hence, efforts have been made to change the mode of the LiDAR when Alex is searching around in the maze. The change in the code is shown below.

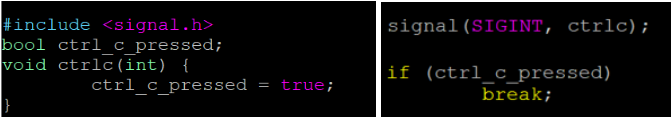


Fig. 10 Checking whether ctrl-c is pressed to jump out of while loop

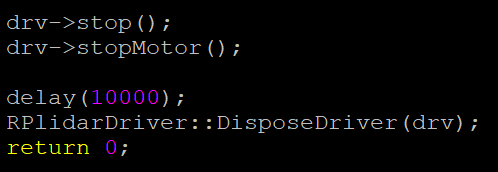


Fig. 11 LiDAR will delay in 10 seconds after ctrl-c is pressed

Initially, LiDAR spins but does not scan. When Alex moves to another location and the user executes the program to start LiDAR’s scanning, GNUplot will update the data. After the scanning is done and GNUplot is updated, the user will press “ctrl + C” and LiDAR will stop spinning and stop scanning in 10 seconds, given spare time for the user to update their drawing map and decide on the next command. After 10 seconds, LiDAR will spin again and the process repeats. Hence, by changing between three modes, the power consumption of LiDAR will be greatly reduced.

1. Implementation of Bare Metal programming:

With the aim of gaining deeper control of the microcontroller, bare metal programming is performed. The implementation of bare metal programming is mainly in two parts, which are Pulse Width Modulation (PWM) and Serialization.

1. PWM part

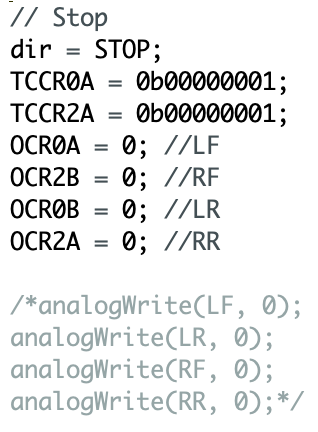
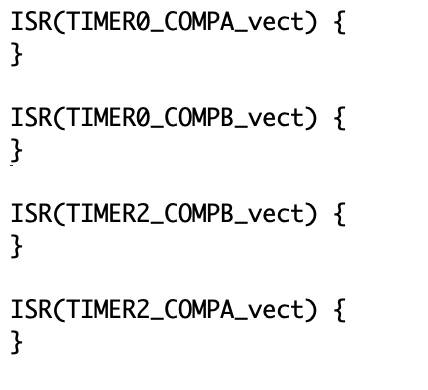


Fig. 12 Enabling ISR Fig. 13 Bare-metal Programming for stop()

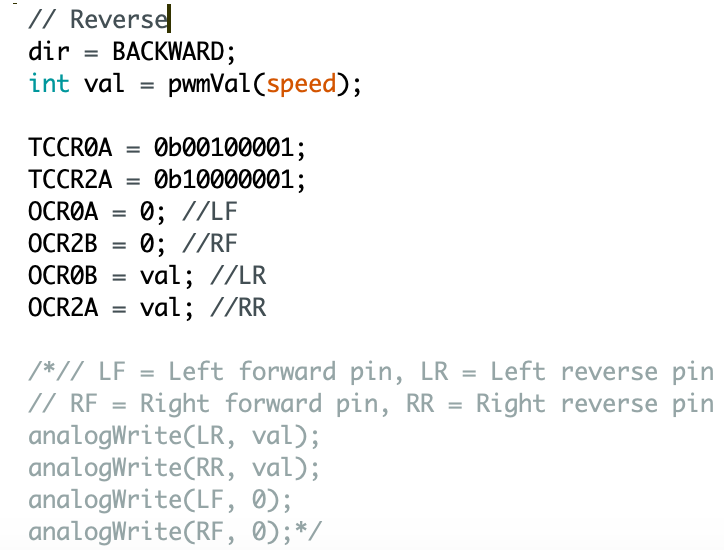
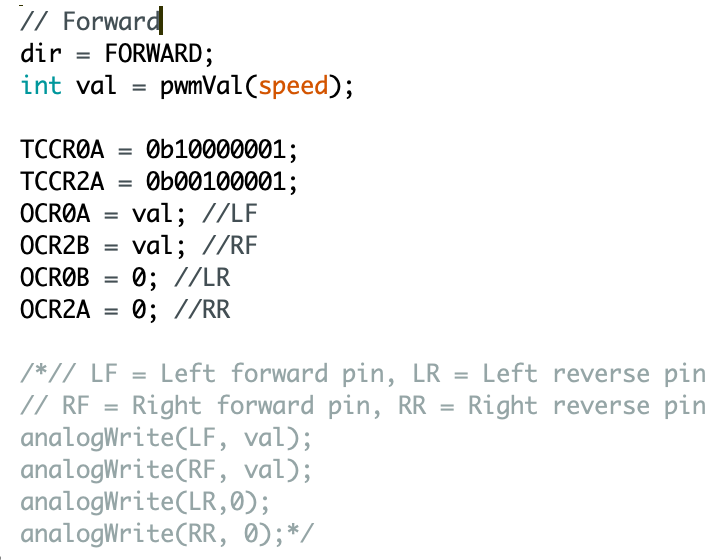


Fig. 14 Bare-metal Programming for forward() Fig. 15 Bare-metal Programming for reverse()

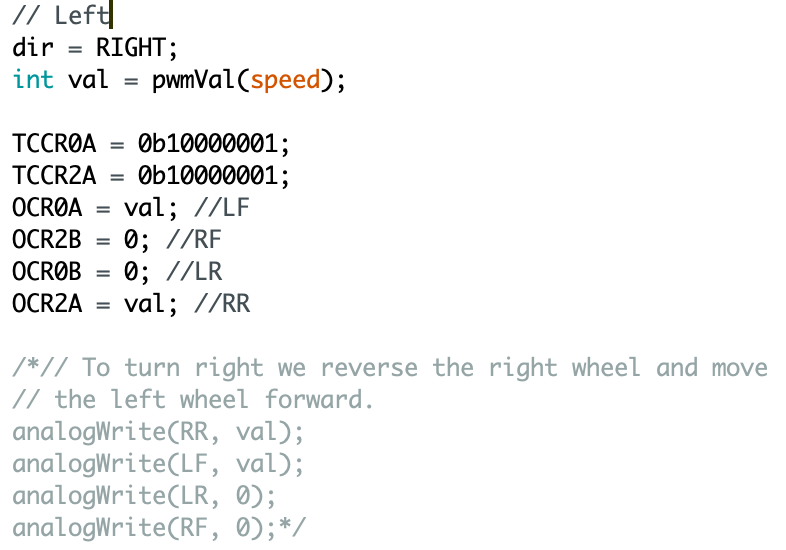
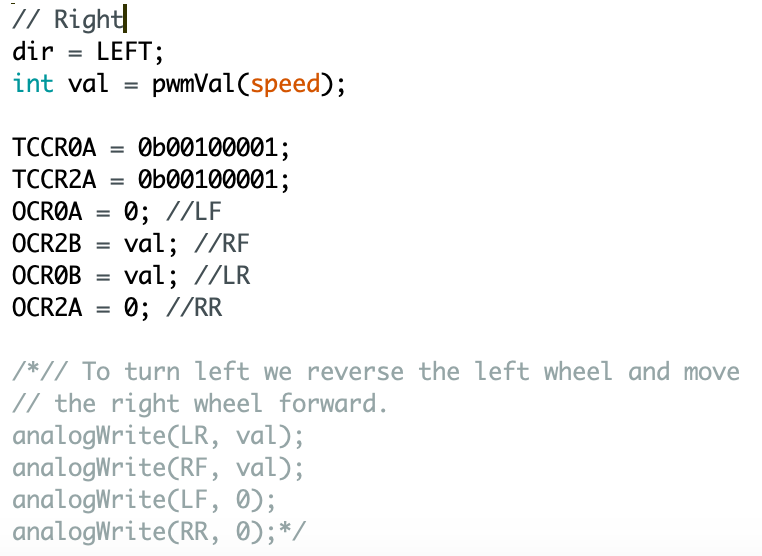


Fig. 16 Bare-metal Programming for right() Fig. 17 Bare-metal Programming for left()

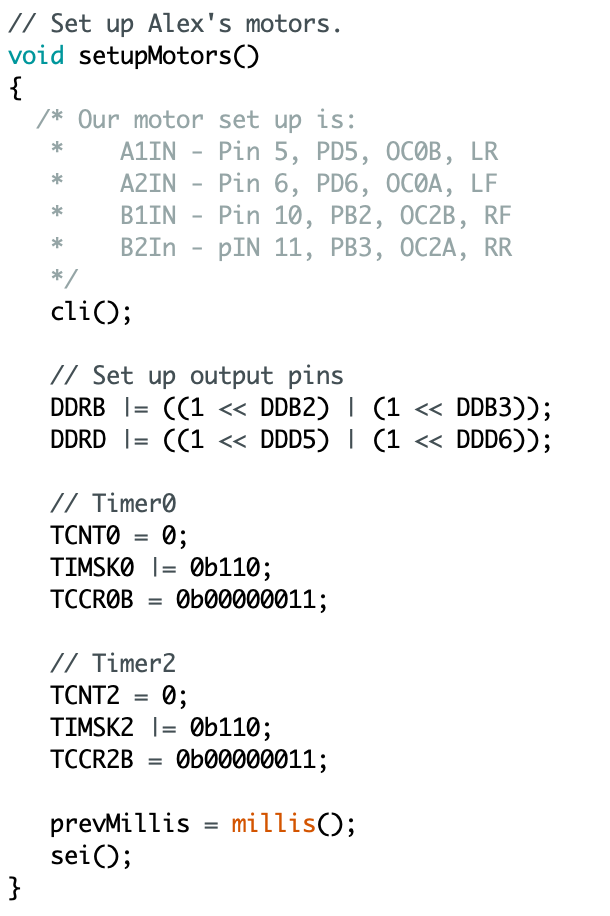


Fig. 18 Bare-metal Programming for setupMotors()

1. Serialization part

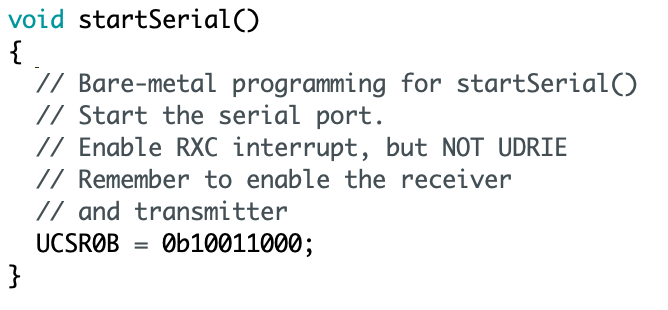
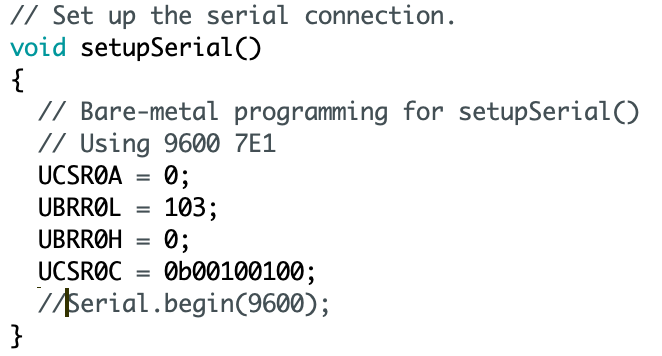


Fig. 19 Bare-metal code for setupSerial() Fig. 20 Bare-metal code for startSerial()

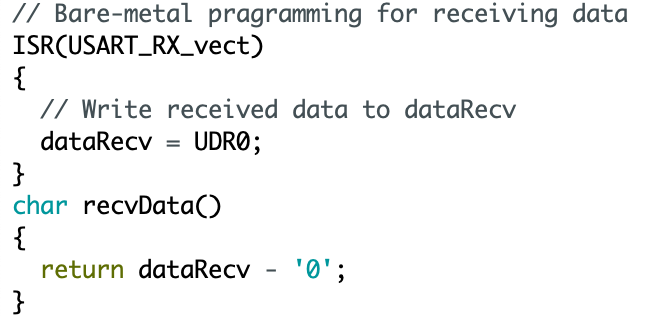
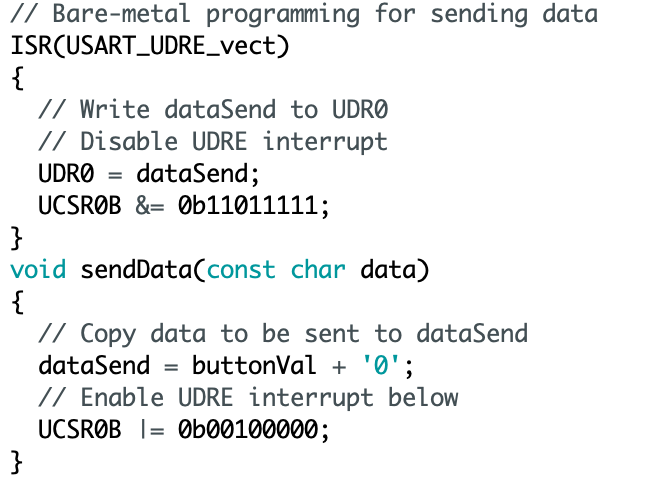


Fig. 21 Bare-metal code for sending data Fig. 22 Bare-metal code for receiving data

4. Secure Alex with TLS Programing:

Since Alex is controlled through teleoperation, it is possible that Alex can be hijacked by other computers. Therefore, additional methods should be made to secure Alex. Particularly, in the RPi and the user’s laptop, TLS programming is implemented in order for both sides to recognize each other before the user gives commands for Alex. By implementing TLS programming, Alex is not only secured but it can also be controlled directly through the user's laptop without using VNC or SSH to connect to the RPi.

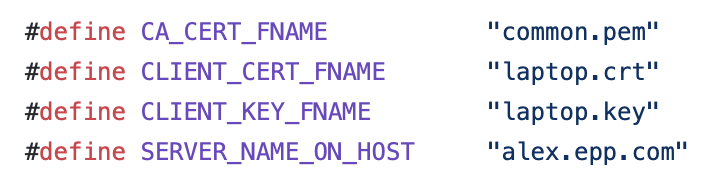
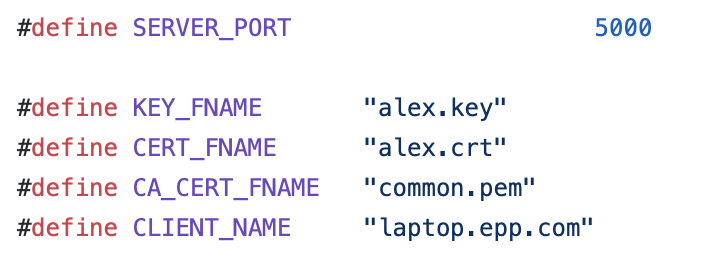
****

Fig. 23 Code from alex-pi.cpp Fig. 24 Code from laptop-client.cpp

After the implementation of TLS programming, Alex and the host can identify each other, leading to a more secure communication.

**Section 7 Lessons Learnt - Conclusion**

There are two lessons learnt from mistakes that occurred during the project. One of the important lessons learnt is that it is crucial to implement power management on Alex as shutting down some unused features can reduce power consumption of Alex to a significant extent. The other lesson learnt is that it is significant to use different remote administration protocols to send commands to LiDAR and RPi. Otherwise when data received from LiDAR is updated, it will affect the insertion of command lines for the movement of Alex. By analyzing the mistakes, improvements have been made to Alex to achieve higher quality and better functionalities.

**References**

[1] Eu-robotics.net. 2020. *RAPOSA NG – A Search And Rescue Land Wheeled Robot*. [online] Available at: <https://www.eu-robotics.net/sparc/success-stories/raposa-ng-a-search-and-rescue-land-wheeled-robot.html?changelang=2>

[2] Marques, Carlos & Cristóvão, João & Alvito, Paulo & Lima, Pedro & Frazão, João & Ribeiro, Isabel & Ventura, Rodrigo. (2007). A search and rescue robot with tele-operated tether docking system. Industrial Robot: An International Journal. 34. 332-338. 10.1108/01439910710749663.

[3] Robots.ieee.org. 2020. *Spot - ROBOTS: Your Guide To The World Of Robotics*. [online] Available at: <https://robots.ieee.org/robots/spotmini/>

[4] Bostondynamics.com. 2020. *Spot® | Boston Dynamics*. [online] Available at: <https://www.bostondynamics.com/spot>

[5] usatoday. 2020. *Boston Dynamics' Spotmini Robot Dog Can Pull More Than Its Weight*. [online] Available at: <https://www.usatoday.com/story/tech/2019/04/17/boston-dynamics-spotmini-robot-dog-can-pull-more-than-its-weight/3494510002/>

[6] Jeffrey, C., 2020. *Boston Dynamics' Spot Is Now On Sale For The Price Of A 'Luxury' Car*. [online] TechSpot. Available at: <https://www.techspot.com/news/82045-boston-dynamics-spot-now-sale-price-luxury-car.html>