CG1112 Engineering Principle and Practice
Semester 2 2018/2019

"Alex to the Rescue"
Final Report
Team: 03-02-02

Name	Student #	Sub-Team	Role
Rohan Arya Varma	A0180364H	Software	
Lim Chee Seng Brian	A0183158Y	Firmware	
Ng Jing Kiat	A0183217H	Software	
Mohamed Riyas	A0194608W	Hardware	
Parvathi Ranjith Menon	A0194448R	Hardware	

Section 1 Introduction

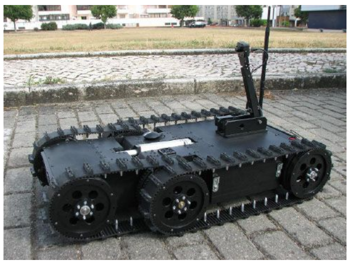
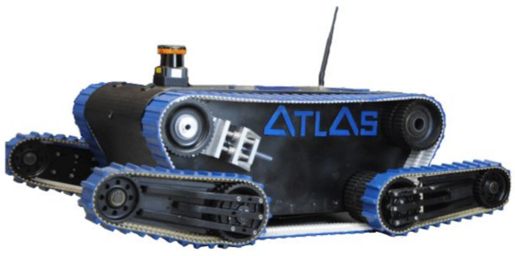
Our project aims to build a tele-operated robot, Alex, which can be used to perform search and rescue operations effectively. In order to achieve this purpose, Alex comes with a variety of specifications and requirements that will be further explored in this report.

In order to demonstrate Alex's capabilities, Alex is placed in a 3 m^2 maze filled with obstacles that simulates a disaster area. These obstacles are designed to be at least 18 cm in height which is also the typical Light Detection and Ranging (LiDAR) mounting height. In addition, the maze contains 2 - 4 rooms of which Alex needs to map out.

Alex is expected to move according to the commands (forward, backward, left or right), map out the maze and identify objects during the course of the simulation. There will be 2 - 3 red or green regularly shaped coloured objects (e.g. cube, cylinder) that will be scattered throughout the rooms. Upon locating an object, Alex will have to determine the colour of the object and report the information to the operator.

The project objectives are met when Alex has successfully performed the above mentioned tasks without faults and collisions.

Section 2 Review of State of the Art

Name with picture	 Fig. 1. RAPOSA [1]	 Fig. 2. ATLAS Urban Search & Rescue Robot [2]
Description	RAPOSA is a Search and Rescue (SAR) robot, designed to operate in outdoors hazardous environments.	ATLAS is a SAR robot, built with the ability to provide support to both victims and responders after disasters.
Software Features	It uses an open source middleware known as 'Yet Another Robot Platform' to support Inter-Process Communication, image processing and device drivers.	It uses Arduino to drive the motors and Robotic Operating Systems (ROS) to interface the electronic hardware and sensors.
Hardware Features	It is equipped with temperature, humidity and gas sensors. In addition, it has a tethering cable to bring power and wired connection if wireless connection fails.	It is equipped with a Light Detection and Ranging (LiDAR) sensor, front and rear cameras, an Inertial Measurement Unit and a Carbon Dioxide sensor.
Strengths	Docking and undocking the robot from the tether can be done remotely by the operator via the camera located inside the robot.	The use of flippers on the lower wheels enable ATLAS to easily move in dangerous terrains and even climb up and down extreme obstacles.
Weakness	RAPOSA lacks the autonomous capability to make split second decisions in the face of danger/emergencies which can further hinder the robot's effectiveness in search and rescue.	The Lithium Polymer batteries used in ATLAS may cause fumes and release Hydrogen if not handled and treated with care. Besides damaging the robot and equipment, this may also result in significant injuries to personnel within the vicinity of the robot.

Section 3 System Architecture

This section explains how Alex's components are connected and how communications between the devices are handled.

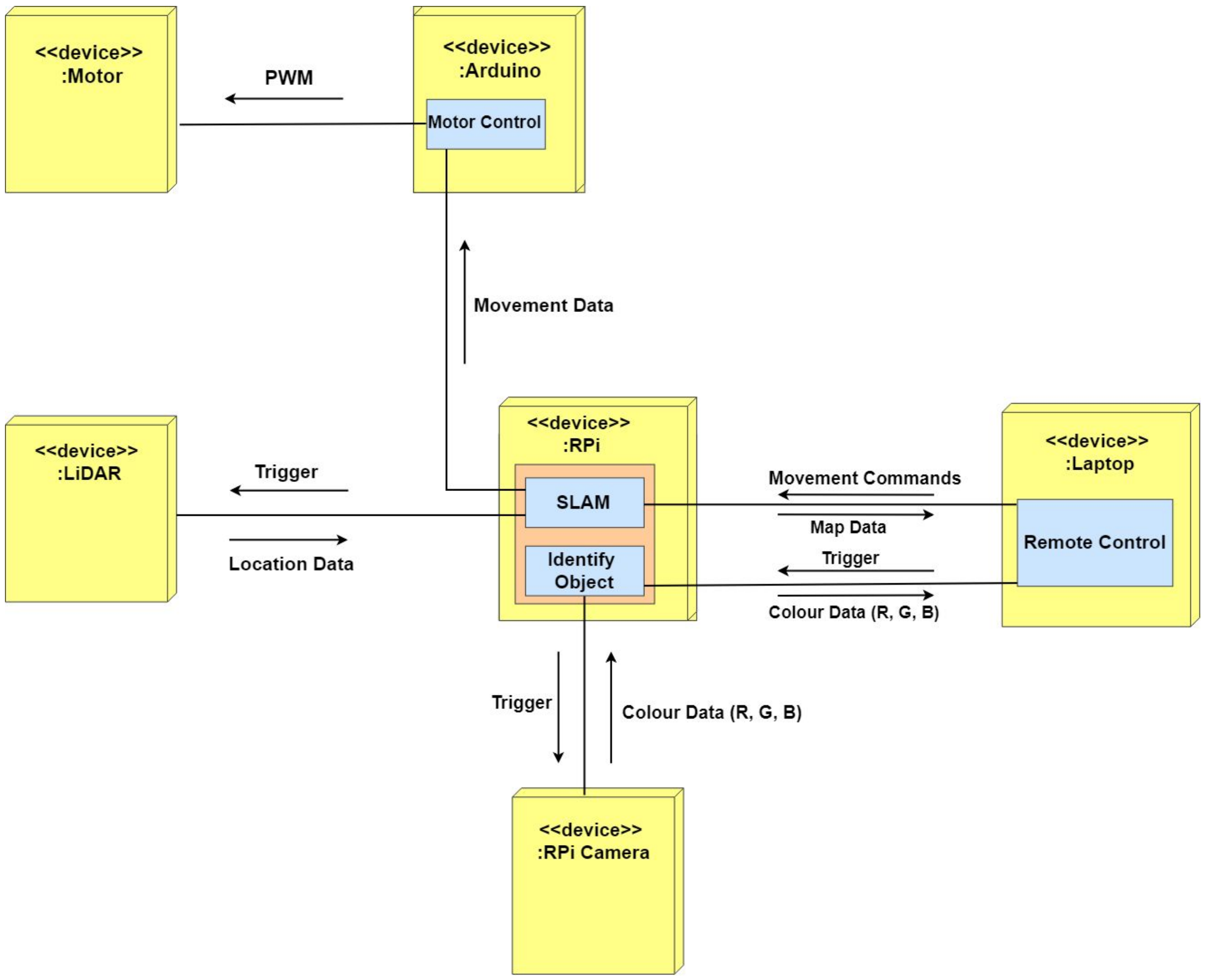


Fig. 3. Flow Diagram of Alex's System Architecture

Communications between the Laptop, Raspberry Pi (RPi) and subsequently all the other components are shown in Fig 3. Communications between the Laptop and RPi will be conducted through a secure Transport Layer Security (TLS) connection to prevent other unauthorized users from controlling Alex. The RPi (server) will receive commands from the Laptop (client) and subsequently relay the commands to all other connected peripherals (Arduino, RPi Camera, LiDAR, Motors).

Section 4 Hardware Design

The various components that are attached onto Alex are displayed in Fig 4.

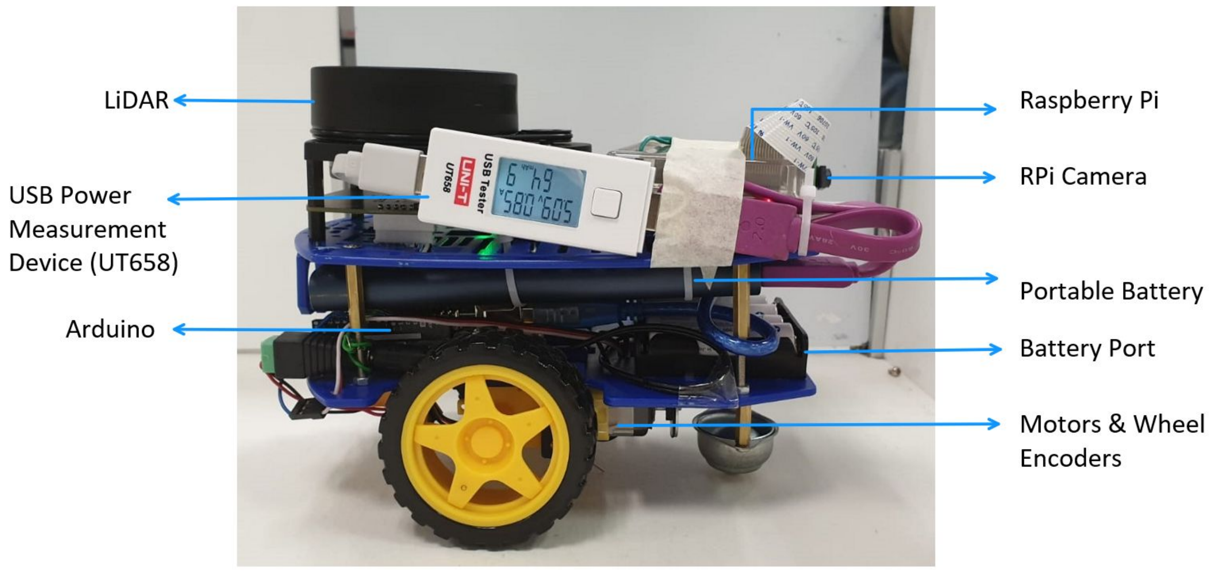


Fig. 4. Finalized design of Alex

The LiDAR was mounted on the top of the chassis such that readings could be obtained without obstruction by other hardware components. The RPi Camera was attached on the Raspberry Pi casing as it provided good support for the camera. In order to optimize the space available, the Portable Battery was inserted above the Arduino and Battery Port. Furthermore, a USB power measurement device (UT658) was included to provide a gauge on Alex's power saving capabilities.

Section 5 Firmware Design

This section discusses the firmware design aspects of Alex.

5.1 High Level Algorithms of Arduino Uno

To begin with, the Arduino acts as an intermediate between the peripherals - the motors and the wheel encoders - and the RPi.

Depending on the instructions received, the Arduino signals the motors to change its Pulse Width Modulation (PWM). PWM is used to control the wheels' direction and speed of rotation. This movement is aided by the wheel encoders that are attached to the motors. The wheel encoders track the number of revolutions of each wheel and send the information to the

Arduino, helping it track the angle of distance remaining to travel when given a certain command. This will allow tracking of the distance travelled by Alex as well as the rotational speed of each individual wheel.

5.2 Communication Protocols

As Alex is a tele-operated robot, commands are required to control the distance, speed and direction of Alex. Hence, a means of communicating with the Arduino is needed. This communication process is known as serial communication and, for the Arduino, is handled by a serial port known as Universal Asynchronous Receiver/Transmitter (UART). The RPi and Arduino communicate with each other by sending information in the form of 1's and 0's. In order for this communication to take place, both the computer and microcontroller must agree on a pre-set baud rate (in this case, equivalent to the number of bits sent per second) and frame (the size and structure of a single unit of data transmission).

In order to ensure that data sent by the RPi to the Arduino is meaningful, packets are used to group bytes of data together so that the microcontroller will be able to understand the command and run the motors accordingly. The following are the packets that are exchanged between the RPi and Arduino.

1. Raspberry Pi to Arduino

   There are only two types of packets that are being sent from the RPi to Arduino:

   a. "Hello" Packet
      The "Hello" Packet is sent to the RPi upon establishing the initial handshake. A successfully sent packet signifies that the initialization process was successful and that the operator is now able to send commands to control Alex.
   b. Command Packet
      The Command Packet contains various keystrokes that controls Alex's movement and speed once connection between RPi and Arduino has been established. It contains the following parameters to control the motors: Direction of movement, Distance to travel/Degree to rotate and Speed.

2. Arduino to Raspberry Pi

The following packets sent from the Arduino to RPi are the responses to inform the operator if the packets were successfully sent over. Whenever a bad response is encountered, the operator is required to resend the packet in order for the command to be carried out.

a. sendOK

   The sendOK response informs the operator that the packet was successfully received by the Arduino and that it is ready to receive the next packet.

b. sendBadChecksum

   The sendBadChecksum response informs the operator that the packet received was corrupted during transmission and it is unable to process the command.

c. sendBadCommand

   The sendBadCommand response informs the operator that the packet received contained invalid command that does not match with existing commands.

d. sendBadPacket

   The sendBadPacket response informs the operator that the packet received was incomplete.

Section 6 Software Design

6.1 High Level Algorithm

1. Initialize
2. Scan room and send map data to laptop
3. Carry out Simultaneous Localization and Mapping (SLAM)
4. Execute command given from laptop
5. Repeat Steps 2 to 4 until all the rooms have been mapped

Further Breakdown:

1. Initialization
    a. Run the server code on the RPi and the client code on the laptop
    b. Initialize ROS and start visualizing Hector SLAM on the visualization software RViz
2. Scan room and send map data to laptop
    a. Send data from LiDAR to the laptop via the secure communication channel
    b. Set the ATmega328P chip in the Arduino to Idle sleep mode using the Sleep Mode Control Register (SMCR) to save power
3. Carry out SLAM on the laptop
    a. Extract possible landmarks from LiDAR Data
    b. Associate observed data points with possible landmarks
    c. Estimate location
    d. Re-observe landmarks with estimated location and observation data in order to update current location and map
    e. Update map
4. Execute command given from laptop
    a. Make the Arduino exit Idle sleep mode by manipulating SMCR
    b. Use polling to obtain user input from the laptop
    c. If there are objects present in the room,
        i. Move Alex towards possible object to obtain RGB colour values via RPi Camera
        ii. Determine colour using check_red and check_green functions
        iii. Send colour of object from RPi to laptop via the secured communication network
        iv. Repeat Steps 4.b.i and 4.b.iii for every possible object in the room, keeping track of overall angle of rotation
        v. Rotate Alex to original orientation by rotating in the opposite direction of the overall angle of rotation
    d. Navigate Alex towards the exit (one of the exits if there are multiple exit options)
5. Repeat Steps 2 to 4 until all rooms have been mapped

6.2 Additional Software Features

Hector SLAM

As compared to traditional SLAM algorithms, which uses odometry data in order to estimate the current position of the robot, Alex carries out a variant of SLAM known as Hector SLAM, which uses the observed data in order to estimate its location [3]. Hence, integration of odometry data obtained from the wheel encoders into the SLAM is not required to be carried out on the laptop. While it does make programming Alex easier, the primary disadvantage of Hector SLAM is that quick rotations will result in inaccurate estimation, shifting the robot and the newly generated map in unexpected locations on RViz. In order to counter this, a more precise control of movement is needed.

Transport Layer Security

In order to ensure secure communications between the Laptop and RPi over the internet, Transport Layer Security, a cryptographic protocol providing end-to-end communications security over networks, is used [4]. This means that private and public keys for both Alex and Laptop are required and thus were generated and signed. The private keys for Alex and Laptop are securely and remotely kept while the public key and Alex's certificate are shared between both devices. In general, devices impersonating the Laptop will not be able to control Alex as the private keys used are different.

Raspberry Pi Camera

Acting as the eye of Alex, the Raspberry Pi Camera Module V2 is used to determine the colour of identified objects. In order to achieve this, a library called Raspicam, developed by Rafael Muñoz Salinas from the University of Córdoba is utilized [5]. This library provides the ability to interact with the camera using C++, and the handling of the camera using this library is carried out in the server code running on the RPi. Once a command corresponding to the camera is received, the RPi will open the camera and retrieve the camera data using `Camera.open()` and `Camera.receive()` respectively. The data consists of 3686400 unsigned characters, with each triplet of values corresponding to the Blue-Green-Red (BGR) values of a pixel. Since the dimensions of the image are 960 pixels tall and 1280 pixels wide, there are 1228800 such triplets. In order to obtain the RGB values corresponding to each pixel, Blue and Red values of each triplet of array values have to be first swapped and then typecasted into unsigned integers. However, the array is one-dimensional, and it is difficult to conduct a pixel-by-pixel analysis. In order to simplify the analysis, a structure called Pixel is created with the attributes of red, green and blue, as shown below:

```
typedef struct Pixel{
    unsigned int red;
    unsigned int blue;
    unsigned int green;
} Pixel;
```

After which, a two dimensional array which stores 960 by 1280 pixels known as `Image` is created. In addition, a special function was constructed in order to transfer data from the original one-dimensional array of data retrieved by the camera to the two dimensional Image array. Once finished, two functions `check_red()` and `check_green()` are used in order to detect the presence of red or green pixels respectively. A corresponding message is then sent to the laptop by the server code running on the RPi so as to inform users of the identified colour of the examined object.

While it was difficult to understand the library and integrate the code into the currently existing code, the use of the camera results in a significant improvement in the efficiency of Alex, as the high resolution of the camera and method of analysis used enables determining the colour of an object from a far distance, which would not be possible if a conventional colour sensor was used. By not having to steer Alex too close to the object for colour identification, time and power consumption can be saved.

Precise Movement

While the method of movement provided worked, a few issues surfaced. Firstly, the code to stop the motors failed to take into account the inertia of Alex, which consistently caused the actual distance or angle travelled to be greater than what was specified. Additionally, since the motors provided are not entirely identical, commanding Alex to move straight or backwards for a large distance would inevitably cause Alex to stray far away from its intended path, which could have caused unintended collisions. Thirdly, as previously mentioned, Hector SLAM can cause unexpected behaviours on RViz when movements and rotations are too quick. As such, there is a need to identify a method of moving Alex precisely.

Such precise movement is achieved through ncurses, a library by GNU Project which allows the terminal window to handle keyboard inputs instantaneously, as compared to the conventional method of input, which requires users to press the ‘Enter’ key after keying in a set of values [6]. The ncurses library is integrated into the client code running on the laptop, resulting in the capability to enter Drive Mode, where the keys ‘w’, ‘a’, ‘s’ and ‘d’ correspond to movement in the forward, leftward, reverse and rightward direction respectively. Each time such a key was pressed, the direction corresponding to the key, an appropriate power level and a small value of direction or angle, was sent to the RPi as a command. The small directional values or angle specified would result in minimum possible movement, hence enabling users to hold or press the keys repeatedly in order to manoeuvre Alex, and instantaneously change the direction of Alex where required. This results in more precise control of the robot, thereby making it more efficient and less likely to collide with walls or obstacles.

Remote Arduino Reset

Even with precise controls, the Arduino can face issues which may inhibit it from working properly. Of which, performing a reset was the most efficient solution to rectify problems faced by the Arduino. However, since the ‘Reset’ button on the Arduino was inaccessible during a SAR operation, it was crucial to find an alternative which allowed remote resetting of the Arduino.

The key to doing so was through Raspberry Pi GPIO programming. Notably, the RPi has 40 GPIO pins [7]. In order to enable resetting through voltage input, the Arduino has an active-low reset pin [8]. This allows normal operation by the Arduino when the voltage input to the pin is read as a logical HIGH. However, the Arduino will reset when the input is read as LOW for a small amount of time and then back to HIGH. Using the library wiringPi, which was included in the operating system of the Raspberry Pi, as well as a wired connection between Pin 2 of the RPi and the reset pin of the Arduino, inputs of LOW and

HIGH patterns to the reset pin can be executed, enabling the RPi to be remotely reset when using the server. While the voltage output of the RPi is 3.3 V, as compared to the typical 5 V input and output voltages of the Arduino, the Arduino still reads the 3.3 V input to its reset pin as a logical HIGH, which removed the need for voltage amplification.

Section 7 Lessons Learnt - Conclusion

Alex is a fully functional search-and-rescue robot capable of effective environmental analysis, localization and mapping. Through its construction, the two greatest lessons came in the form of the importance of managing files properly and that of understanding the code. When it came to file management, Git was used extensively as a form of version control, which saved us at many occurrences, such as when the RPi failed to reboot and we had to redownload the Operating System. Organizing the relevant files in folders and using appropriate nomenclature were critical decisions that positively impacted our productivity and communication, especially since the project was of a large scale and included multiple sections of code working in unison.

Before integrating more complicated concepts such as bare-metal programming and socket programming, we ensured that we understood the libraries given to us. This turned out to be a great decision, since we were able to fix bugs more quickly and become more productive.

However, we also made mistakes that cost us a significant amount of time. In order to connect to the school's Wi-Fi without the need for Ethernet, we considered downloading a network manager. Despite having followed step-by-step instructions on the downloading process obtained from reliable websites, we were still unable to obtain the required network manager. As one of the steps involved purging the "dhcpcd5" package, which was essential to the connection process, we realised that we were unable to connect to the internet at all despite the RPi being able to detect various connection sources.

The first major lesson we learnt was that simply doing a "sudo apt install" on the RPi does not mean that the files required will be successfully downloaded despite having used the "sudo" keyword. We have to perform a "sudo apt-get update" on the RPi before entering the "sudo apt install" command. After rectifying this, we were able to download the network manager successfully. In addition, we learnt that we should always research on package properties before performing any purging which can lead to irreversible consequences. Thus, we manually downloaded the “dhcpcd5” package from online and performed an offline installation, thereafter successfully connecting to the school Wi-Fi without the use of an ethernet cable.

All in all, the project was a great experience to not only implement the concepts we learnt in this module to a real-life, miniature ‘Search and Rescue’ robot building, but also to witness the whole lab coming together to assist one another and bond over common interests when working on the robot.

Reference

[1] C. Marques et al. *"A search and rescue robot with tele-operated tether docking system,"* Industrial Robot: An International Journal, vol. 34, no. 4, pp. 332-338 [Accessed: 22-Apr-2019].

[2] Baker, G. et.al. *ATLAS Urban Search and Rescue Technical Report 2016/2017*. [online] Warwick.ac.uk. Available at: https://warwick.ac.uk/fac/sci/eng/meng/wmr/projects/rescue/reports/1617reports /wmr_technical_report_2016-17.pdf [Accessed: 21-Apr-2019].

[3] "hector_slam," *ROS.org*, 17-Apr-2014. [Online]. Available: http://wiki.ros.org/hector_slam. [Accessed: 22-Apr-2019].

[4] Kerravala, Z. (2018). *What is Transport Layer Security (TLS)?*. [Online] Network World. Available: https://www.networkworld.com/article/2303073/lan-wan-what-is-transport-laye r-security-protocol.html [Accessed 22-Apr-2019].

[5] R. M. Salinas. "RaspiCam: C API for using Raspberry camera with/without OpenCv," *Aplicaciones de la Visión Artificial*, 16-Feb-2017. [Online]. Available: http://www.uco.es/investiga/grupos/ava/node/40. [Accessed: 22-Apr-2019].

[6] "Announcing ncurses 6.1," *GNU Project*, 28-Jan-2018. [Online]. Available: https://www.gnu.org/software/ncurses/. [Accessed: 22-Apr-2019].

[7] "Raspberry Pi 3 Model B ," *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. [Accessed: 22-Apr-2019].

[8] M. James. (2018, Oct. 31). "How to use an External Reset Button with Arduino :: Viewer Question #6," *Programming Electronics Academy*. [Online]. Available: https://programmingelectronics.com/how-to-use-an-external-reset-button-with-a rduino/. [Accessed: 22-Apr-2019].