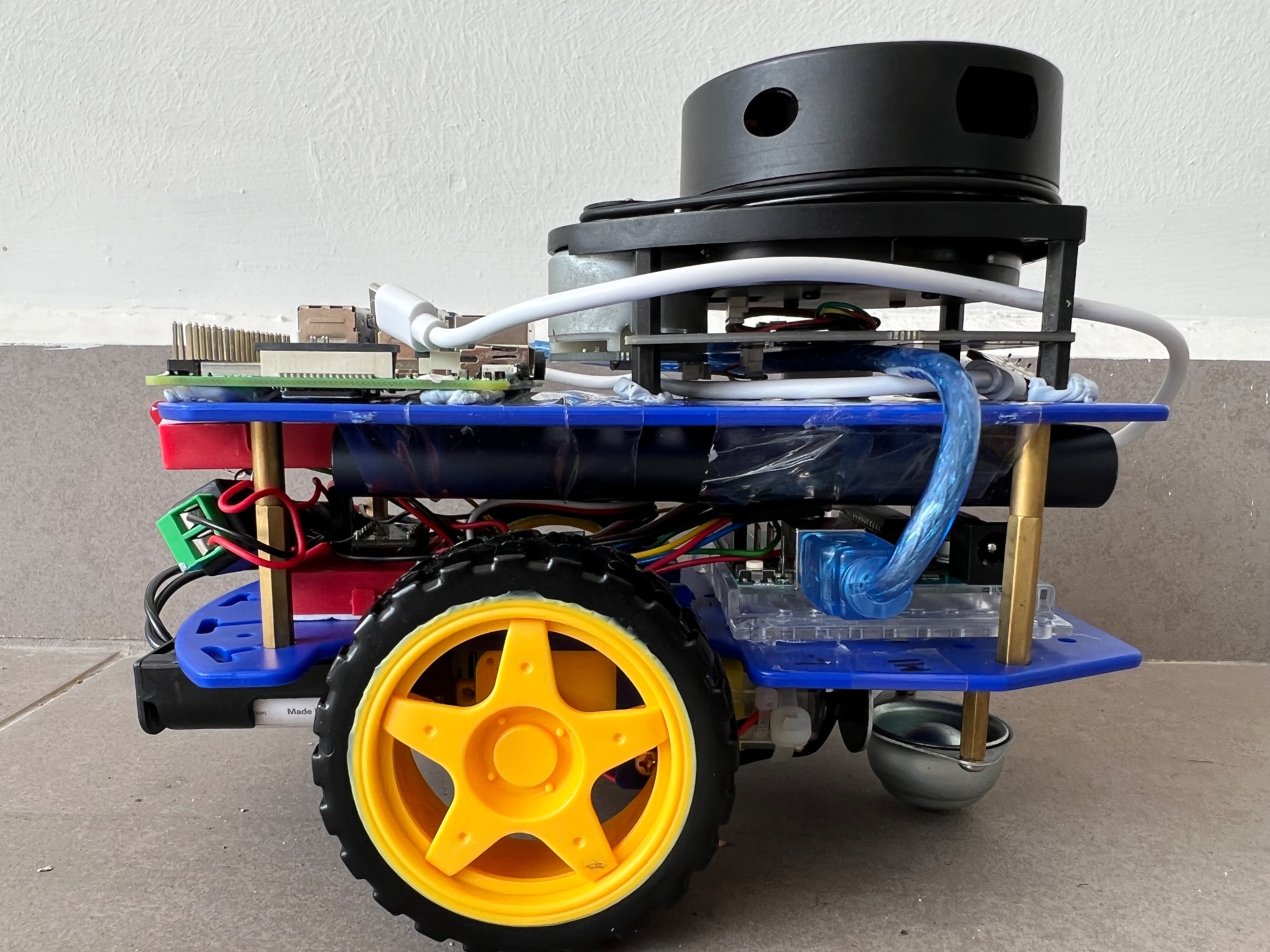


**CG2111A**

**Engineering Principle and Practice 2**

Semester 2 2021/2022



**“Alex to the Rescue”**

**Final Report**

**Team: B04-2B**

**Group Members:**

|  |  |  |
| --- | --- | --- |
| Name | Student No | Role |
| Lee Jun Hao Bryan | A0233090R | Leader |
| Varatharaju Vignesh | A0233677W | Coordinator |
| Shannen Tan | A0238815L | Sub-team leader |
| Udayakumar Nivetha | A0245895L | Quality Checker |

**Table of Contents**

1. Introduction ………………………………………………………………………………………... 3

* 1.1 Search and Rescue Problem
* 1.2 Alex: A Remotely Piloted Cloud Controllable Robot

2. Review of State of the Art…………………………………………………………………………4

* 2.1 Quince (Rescue Robot)
* 2.2 ROBOCUE (Rescue Robot)

3. System Architecture……………………………. ………………………………………………...5

4. Hardware Design…………………………………………………………. ………………………6

5. Firmware Design…………………………………………………………………………………..9

6. Software Design………………………………….. ……………………………………………..12

7. Conclusion………………………………………… ……………………………………………..13

8. References………………………………………………………………………………………..14

**Section 1: Introduction**

**1.1 Search and Rescue Problem**

Natural disasters are unpredictable and have a significant cost to human life. It is common to see rescuers scrambling after an earthquake to quickly identify any survivors stuck under rubble. With the “golden period” being just 72 hours, rescuers are pressed for time to find survivors.

The advancements in technology have allowed the use of rescue robots that work with rescuers to assist with identifying potential survivors trapped under rubble. Due to the small size of rescue robots, they are able to easily fit through small gaps and travel between rubble. Then, by communicating with the user, they are able to identify the precise location of survivors which helps make rescue efforts targeted towards the survivor.

The current task is to build ALEX, a robotic vehicle with search and rescue functionalities.

**1.2 Alex: A Remotely Piloted Cloud Controllable Robot.**

Alex can be remotely controlled to map the surroundings. The user will use a Secure Shell Protocol via a laptop to access and control the pi’s terminal over the network.

Alex is mounted with a LIDAR unit which performs 360° environmental scans within a 6 meter range. Information received from the LIDAR is then used to generate a 2D map which would be shown to the user in real time throughout the operation of Alex.

The user will control the robot remotely by sending in commands using the 2D map data that is being generated to navigate through the simulated environment. Constant communication is being engaged with the Master Control Program (MCP) on Pi to translate commands into movements control signals for the Arduino board connected to the Pi. Such commands include moving forward, reversing, turning right and turning left.

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

**2.1 Quince (Rescue Robot)**

Quince is a rescue robot designed for CBRNE disasters, namely Chemical, Biological, Radiological, Nuclear and Explosive hazards. It was unveiled in April 2010 by the Chiba Institute of Technology, Tohoku University, and IRS. Quince weighs 27kg and has 4 moving caterpillar drives which automatically adapt their angular position to the surface beneath.



Quince is equipped with a “bird’s eye camera”, able to move fast, and can be controlled remotely. The operator controlling the robot has to tell it which direction to move, but the robot itself will decide the optimal flipper position for crossing the different surfaces. Newer models are able to collect radioactive dust or ultra-fine particles, and also have 3D scanners. Connection to a wireless network is possible in the event where the connection cable breaks.



**Main Strength:** The 4 caterpillar drives allow the robots to move through different surfaces and is able to move up stairs and detect gasses, radiation or other life-threatening hazards before a human rescue team can search the area.

**Main weakness:** There is no tracking device on the Quince rescue robots and hence may be hard to track its location if they fall out of sight .

**2.2 ROBOCUE (Rescue Robot)**

Robocue is a search-and-rescue robot developed in 2008 by the Tokyo Fire Department. It can be controlled remotely by an operator and used from 100m away. It is tethered by a 328-foot cable and equipped with infra-red cameras, a megaphone and ultrasonic sensors to help search for victims trapped in dangerous places where the rescuers are unable to reach. 

Upon locating the victim it is able to move the victim into its body via a conveyor belt. The body consists of an onboard oxygen canister if needed.

**Main strength:** Active scope camera that is coated with plastic cilia that can squeeze through tiny openings to look for victims that may be trapped in rubble during a disaster

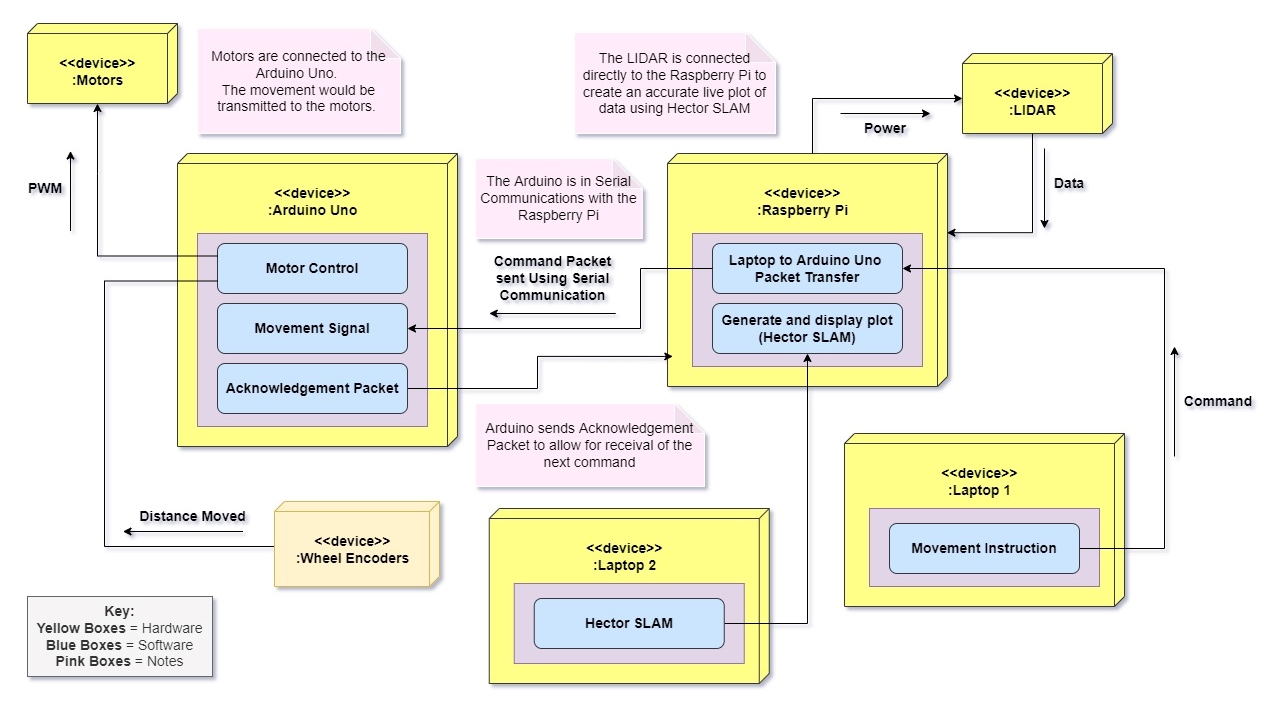
**Main weakness:** If victims are trapped in the rubbles and are stuck, the robot will be unable to extract the victims and rescue teams will still be required to head down to the area and perform the evacuation themselves

**Section 3: System Architecture**

**Overview:**

Figure 1 below gives a brief breakdown of the different components that help Alex run its functions. For the hardware connections, the LIDAR is connected directly to the Raspberry Pi. The LIDAR will then collect data of its surroundings using a 360 degree laser scanner and send the data collected to the Pi. This allows the Pi to plot an accurate graph of its surroundings through the distance data provided by the LIDAR which is analyzed and processed using the SLAM algorithm. HECTOR Slam and Rviz is used to visualize the data provided by the LIDAR which allows the user to navigate ALEX.

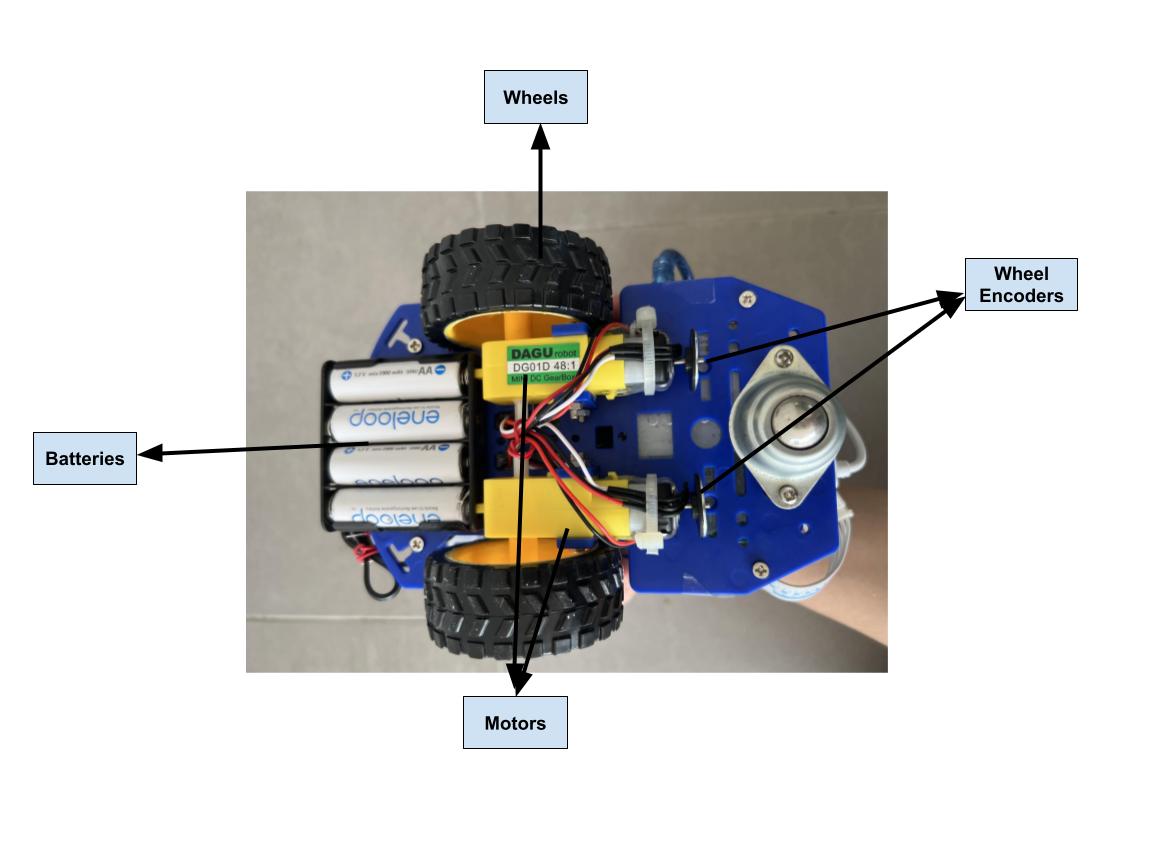
The Motors are connected to the Arduino which is in serial communications with the Pi. The user uses the Pi to transmit commands to the Arduino. Once a command is given to the Pi, the Pi responds with “Command OK” when the command is received successfully. Afterwards, through serial communication, a serialized buffer is sent to the Arduino. The arduino receives the command to be handled by deserializing this buffer.

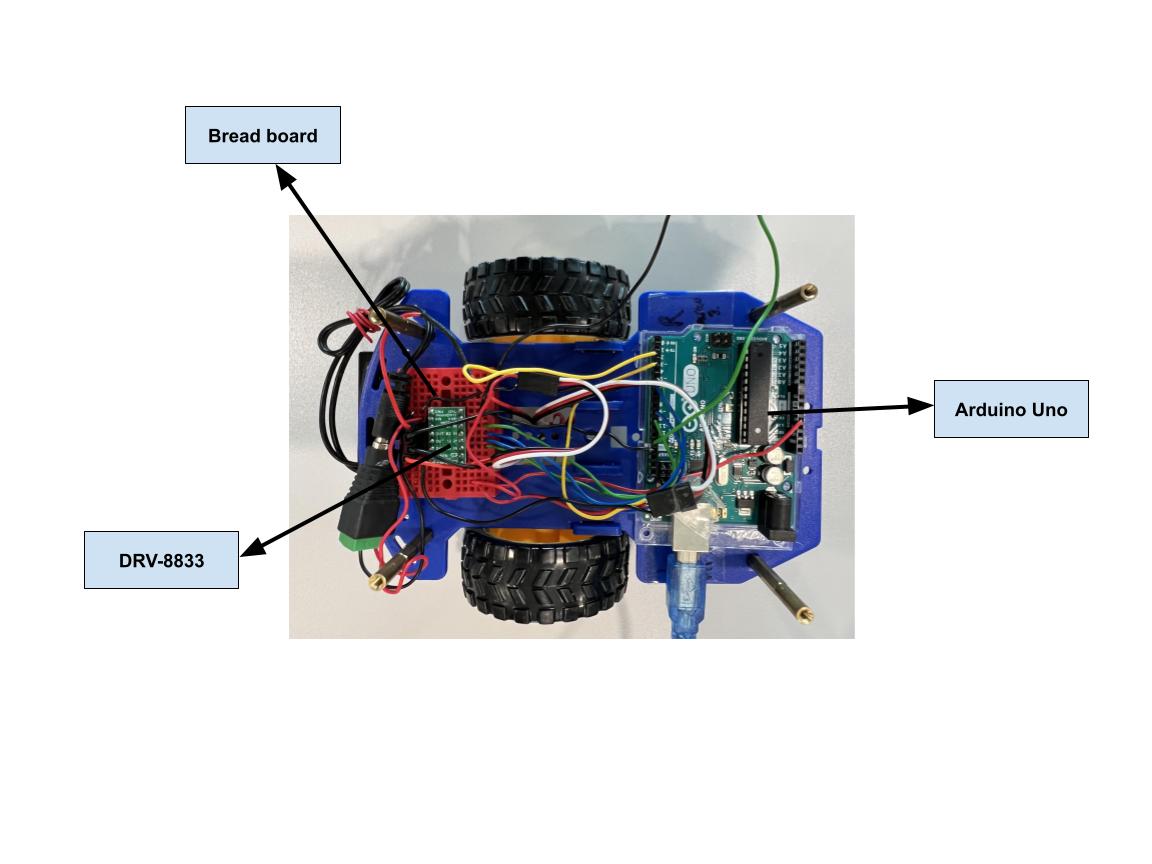


**Figure 3: Overview of Alex systems**

**Section 4: Hardware Design**

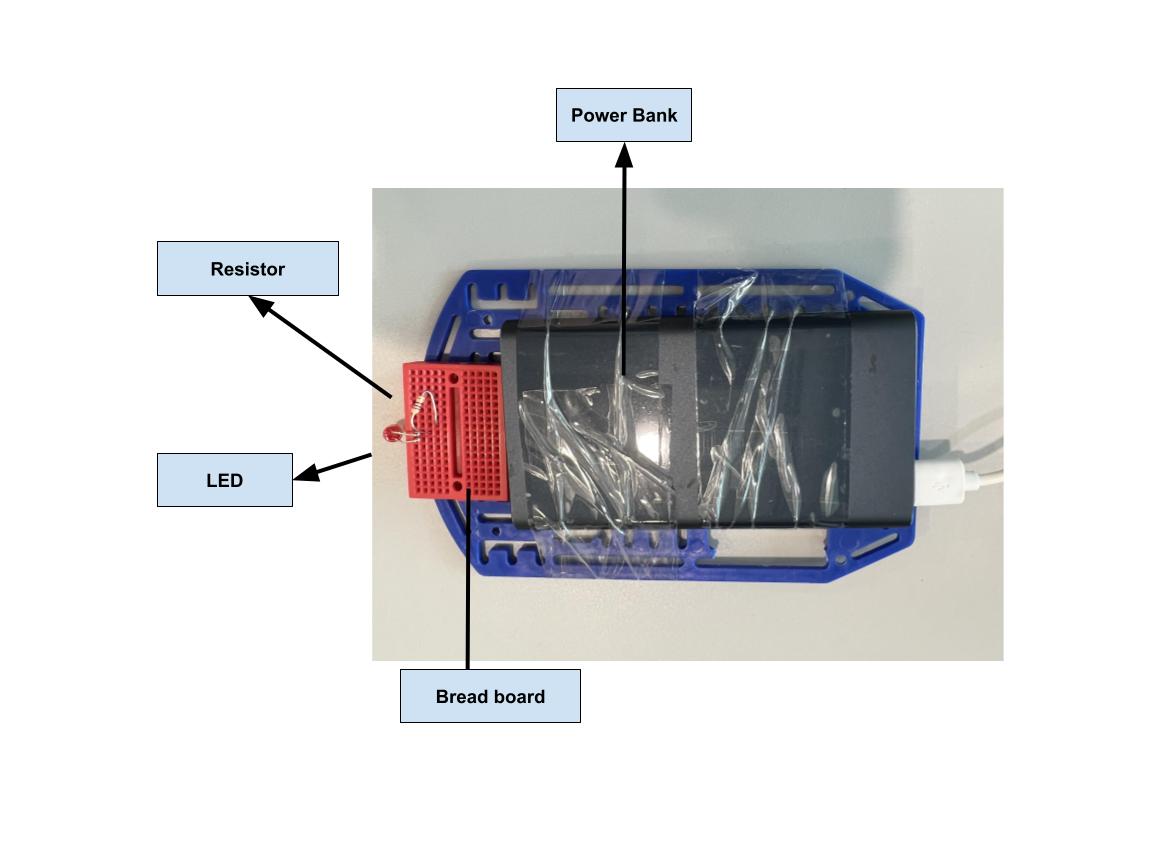
**Overview:**

The Hardware component of Alex consists of the robot kit, Arduino board, bread boards, power bank, battery holder with batteries, motors, Lidar, Pi, wheels and wheel encoders.

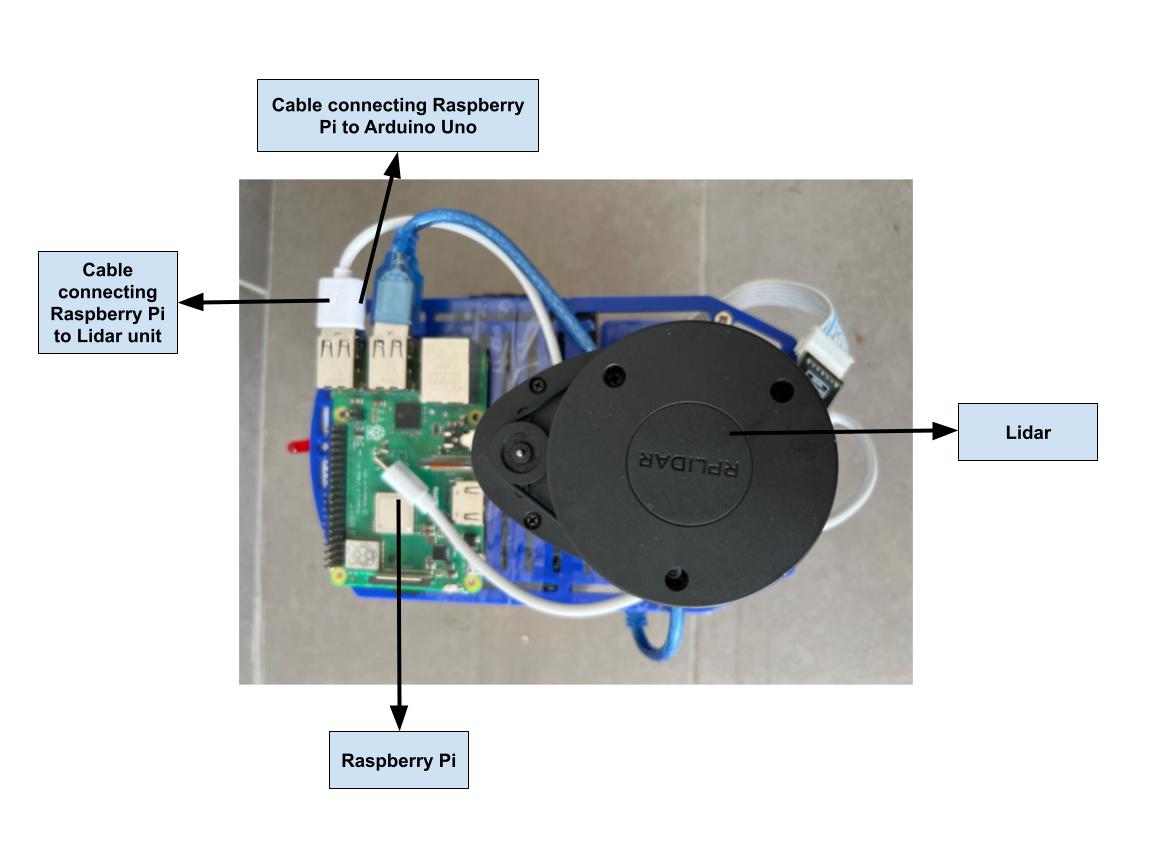


**Figure 4.1: Bottom View of Alex**

**Figure 4.2: Middle segment bottom deck of Alex**



**Figure 4.3: Middle segment top deck of Alex**

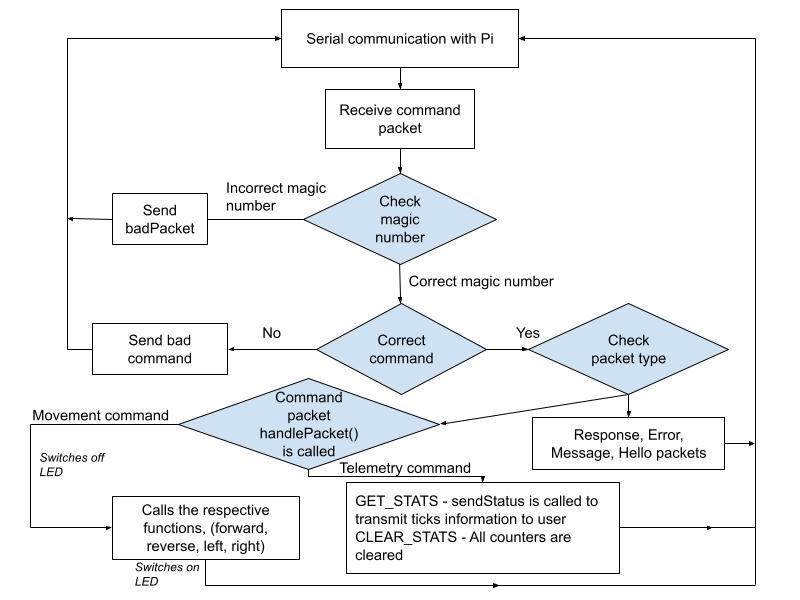


**Figure 4.4: Top view of Alex**

**Table 1: Hardware components of Alex**

|  |  |  |
| --- | --- | --- |
| **Name** | **Function** | **Placement** |
| Wheels X2 | Connected to the rotating shaft of the motor. Allows alex to move when the motor moves | Connected to the motors at the bottom deck of Alex |
| Motors | Provides rotational power from electrical power received from the battery to allow the wheels to move | At the bottom deck of Alex |
| Wheel Encoder | Measures the rotation of the magnetic disk that is attached to the motor | Attached to the front of the motors |
| AA Batteries X4 | Provides electrical power for the motors | Placed at the bottom deck of Alex |
| Arduino Uno | Receives instructions from the Pi and executes the command given by user | Placed in the middle segment of Alex at the deck |
| DRV8833 Dual H-Bridge Motor Driver | Receives instructions from the Arduino and powers the motor using the batteries | Placed in the middle segment of Alex mounted onto the bread board. |
| 10 000 MAH power bank | Used to power the Raspberry Pi and the Arduino Uno | Mounted at the middle segment at the top deck of Alex |
| RPLidar A1M8 | Uses light pulses to map the surroundings and sends the data received to the pi | Mounted at the top deck of alex at the back |
| Raspberry Pi | Receives command and transmits map data to the remote user via Wifi. Receives data regarding the map from Lidar.  Sends commands to arduino to operate other hardware components eg: motors/LED | Mounted at the top deck of Alex |
| Red LED | Used to determine if Alex is moving | Placed at the middle segment of Alex facing the front. |
| 2700 Ω resistor | Used to reduce the current supplied to the LED | Placed at the middle segment of Alex facing the front. |

**Section 5: Firmware Design**

****

**Figure 5.1: Brief Firmware Design Flowchart**

Firstly, serial communication is initialized between the Pi and the arduino. The serial communication is done at a baud rate of 9600 with a frame format of 8N1. (No parity bits and 1 stop bit). The Pi waits for a command from the user before transmitting it to the arduino as a serialized packet.

The Tpacket being sent through serial communication has the following components,

* char packetType;
* char command;
* char dummy[2]; // Padding to make up 4 bytes
* char data[MAX\_STR\_LEN]; // String data
* uint32\_t params[16];

The arduino deserializes the packet and receives the command to be executed. After receiving it successfully, it acknowledges by sending an okPacket to the pi. The arduino executes the command after identifying the command type in the handleCommand function.

The handleCommand function is split into the cases,

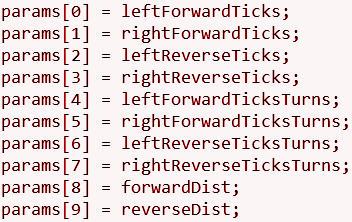
* PACKET\_TYPE\_COMMAND
* PACKET\_TYPE\_RESPONSE
* PACKET\_TYPE\_ERROR
* PACKET\_TYPE\_MESSAGE
* PACKET\_TYPE\_HELLO

If a command packet is given, (PACKET\_TYPE\_COMMAND), the “handleCommand” function is then called to execute the command.

With regards to command packets, the packetTypes are

* COMMAND\_FORWARD = 0,
* COMMAND\_REVERSE = 1,
* COMMAND\_TURN\_LEFT = 2,
* COMMAND\_TURN\_RIGHT = 3,
* COMMAND\_STOP = 4,
* COMMAND\_GET\_STATS = 5,
* COMMAND\_CLEAR\_STATS = 6,

COMMAND\_GET\_STATS is used to obtain odometry data from the Pi and COMMAND\_CLEAR\_STATS reset the data for each counter to 0. “uint32\_t params[16]” array is used to store the odometry information.



**FIGURE 5.2: uint32\_t params[16] array**

The rest of the commands are used to move forward, reverse, turn left, turn right and stop. In each case, the respective function is called to execute the command. For example, for the case COMMAND\_FORWARD, the forward() function which takes in the distance and power as input is called to execute the command.

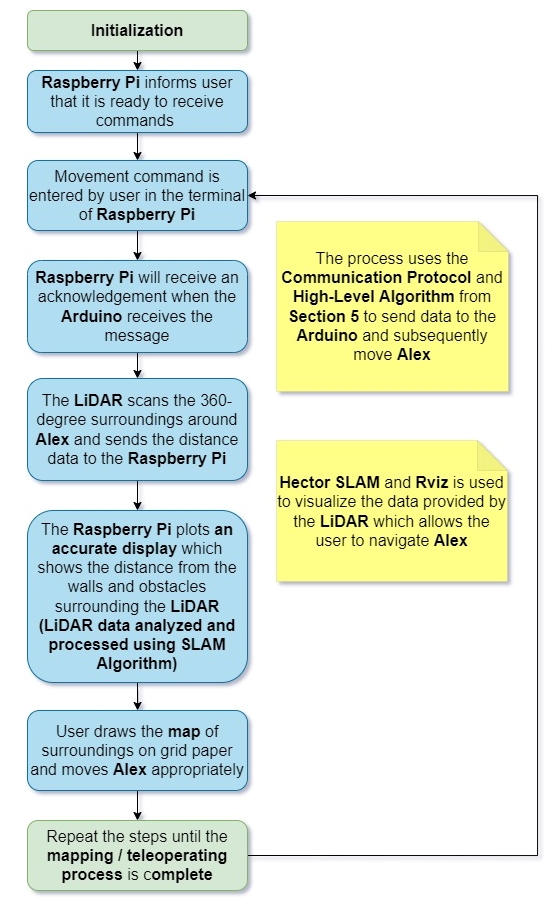
Additional

* The parameters for moving forward and reversing are preset at 4cm at 90 percent power while the turns are both 4° at 80 percent power. Hence, the user does not need to state any parameters when the command is given.
* A Red LED has been connected to Arduino pin 13, PB5. When a movement command is called, the Red LED is switched off. Once the movement has been executed, the Red LED is then switched back on.
* This allows for a physical response from ALEX to indicate when a movement command has been completed by ALEX.

**Section 6: Software Design**

**High Level Steps for Teleoperation:**

1. Initialization
2. Pi informs the user that it is ready to receive commands
3. Pi receives command from user
4. Pi executes command received
5. Repeat steps 2 to 4 till navigation is over



**Figure 6: Environment Mapping Process**

**Additional Noteworthy Features**

The movements moving forward and reversing had fixed parameters of 4 cm distance at 90 percent power, which were tagged to the commands ‘W’ and ‘S’ respectively. The movements turning left and turning right had fixed parameters 4° at 80 percent power, and were tagged to the commands ‘A’ and ‘D’ respectively.

**Section 7: Lessons Learnt - Conclusion**

The two most important lessons we learnt were that we should be flexible, and to always have a backup plan.

We learnt that we should be flexible as our plans changed multiple times, and it was crucial that we embraced the changes. One change we made was rotating the direction that the LIDAR unit was facing (from facing the left of Alex to facing forward). This was for a more convenient and accurate mapping of the maze on the visualiser Rviz. Another change was extending the height of the middle deck, by using additional spacers. This was to accommodate the power bank, Arduino and breadboard (with the DRV8833 motor driver) on the middle deck. These changes came with us having a better understanding of how to better complete the project, and it was not in our favor to resist such changes.

Another lesson we learnt was to always have a backup plan. Before the trial demo, we managed to use gnuplot to plot the map of the maze, so we planned to use gnuplot for the trial demo. However, gnuplot crashed many times during our trial demo. Since we had not set up SLAM, we did not have a backup to use during the trial demo, which resulted in us wasting a lot of time during the trial demo trying to restart gnuplot. This taught us not to rely on a single method, and we set up SLAM soon after that.

The two greatest mistakes we made were prioritizing the wrong activities, and not being decisive.

We prioritized activities such as compacting Alex by fitting all the hardware components and tying all the wires together, and setting up the softwares (SLAM and gnuplot), which took up more time than it should have. As a result, we barely had time to practice tele-operating Alex and drawing the map from the visualizer Rviz. This meant that our team lacked the coordination necessary for us to work as a team during the final demo. Had we prioritized practicing controlling Alex and drawing the map, we would have been able to navigate the maze and draw a better map during the final demo.

During the final demo, we had to complete navigating the maze in 8 minutes. This meant that we had to know what decisions to make, and to make them quickly — any delay in the decision-making process would delay the time taken to park Alex. We were thrown off course when Alex swerved while crossing the hump during the final demo, as the sudden change in direction caused the map to no longer be accurate. Subsequently, we were unable to refresh the map (which we were relying on to decide how to navigate the maze), and our indecisiveness took too much time for us to complete the maze and park Alex.

**References**

* fuRo:Quince. (n.d.). Furo.org. Retrieved April 22, 2022, from <https://furo.org/en/works/quince.html>
* Over rough and smooth -Robots on a mission to explore. (n.d.). Retrieved April 22, 2022, from <https://www.maxongroup.in/medias/sys_master/8810220879902.pdf?attachment=true>
* Rescue Robots | Sci-tech | Trends in Japan | Web Japan. (n.d.). Web-Japan.org. Retrieved April 22, 2022, from <https://web-japan.org/trends/09_sci-tech/sci100909.html>