CG2111A Engineering Principle and Practice II

Semester 2 2022/2023

"Alex to the Rescue"
Final Report
Team: B01-2A

Name	Student #	Sub-Team	Role
Benjamin Goh Guan Rui	A0251731M	Hardware	Design & Procurement
Bryan Castorius Halim	A0255783R	Hardware	Networking Lead
Chua Qing Wei Ryan	A0254458W	Software	Coding & Debug Lead
Yeo Meng Han	A0251772A	Software	ROS / SLAM Lead

Section 1: Introduction **3**
Section 2: Review of State of the Art **3**
TALON 3
Rotundus GroundBot 3
Section 3: System Architecture **4**
Raspberry Pi Architecture 4
Arduino Uno Architecture 5
Section 4: Hardware Design **5**
Section 5: Firmware Design **8**
High level steps outline 8
Initialisation 9
Receive user commands 9
Section 6: Software Design **10**
Navigation 10
Colour Sensor 11
Play music 12
Mapping 12
Section 7: Lessons Learnt - Conclusion **12**

Section 1: Introduction

Natural and manmade disasters can happen at any time, and rescue workers have to race against time to locate and rescue any survivors in the aftermath of these disasters. However, there are many obstacles and dangers that prevent rescuers from being able to save the victims. Fortunately, the advancements of robots have made search and rescue (SAR) operations more efficient and safer for the first responders.

The aim of the project is to build a tele-operated SAR robot, Alex, to map the surroundings of an unknown terrain as accurately as possible. Using the data from the mapping, we would navigate through the whole terrain as fast as possible, while ensuring we do not bump into the environment. Also, we have to be able to identify the colour of the objects scattered around the terrain that are detected by the Alex.

Section 2: Review of State of the Art

TALON

The TALON robot is a military tactical robot developed by QinetiQ North America, and has been around since the 2000. The TALON has been deployed in different situations, like military, first responder and law enforcement[1]. It uses both an External IP Radio and Gigabit Ethernet Communications Backbone to communicate information with the TALON from up to 800m away.

One strength of the TALON is the gripper which is able to carry up to 50kg to give it more autonomy. The robot also gives the user a comprehensive view of the situation with the different cameras and sensors that can be attached. Another strength of the TALON robot is its battery life. It contains two 300Wh batteries, and can be extended using hot-swappable batteries, which allows it to perform for prolonged periods of time without rest.

One weakness of the TALON robot is its mobility, as it has trouble travelling in uneven terrain. The standard TALON[2] is able to travel up slopes easily, however with the addition of the cameras and sensors, it increases the weight of the arm, which will cause the centre of gravity to increase, making it more likely to flip over when overcoming uneven terrain[3].

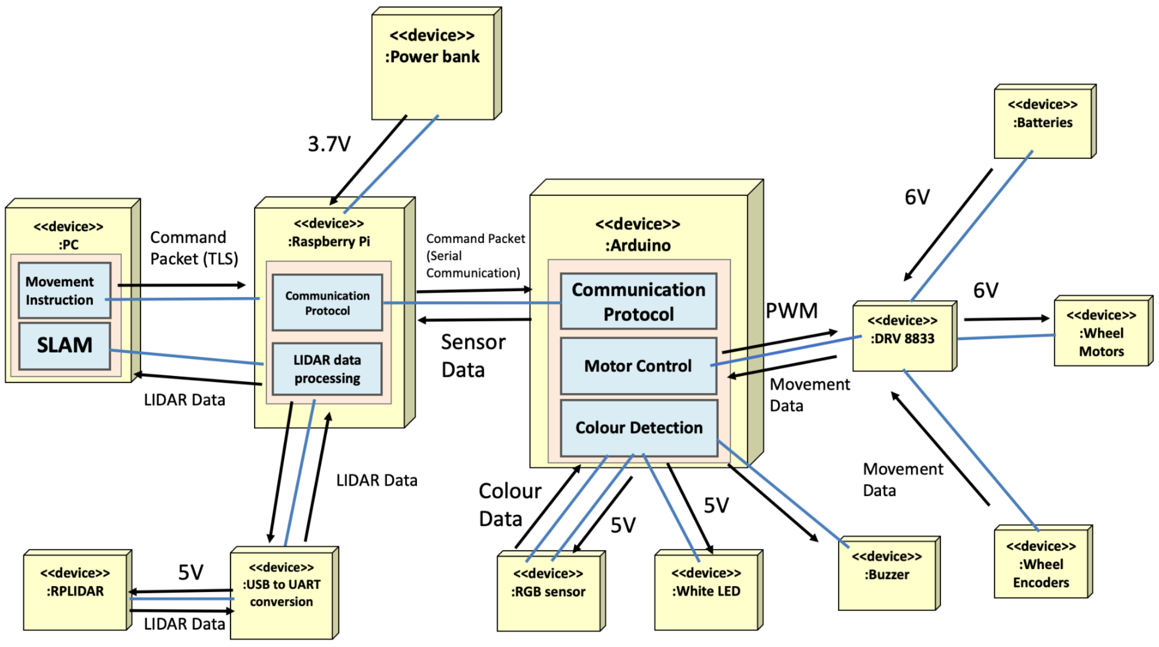
Rotundus GroundBot

The Rotundus GroundBot is a remote controlled spherical robot, which is capable of traversing in different environments such as land, mud, sand, snow and even on water. It was originally used for military operations, but has been applied to fit in the context of civil security[4]. It can be controlled remotely or via a programmed autonomous GPS-based system with its Two dual-band (L1 and L2) GPS receivers[5].

Strengths of the GroundBot includes its adaptability as it is able to equip different types of cameras for different missions, such as wide-angle cameras, night vision cameras, as well as a variety of sensors like gas, radioactivity and fire. All these cameras and sensors are well protected against sand, mud, water and gas, making it suited to handle harsh environments. Another strength is its reliable communication, as it usually uses wireless connection with Wi-Fi, but it also has a separate secure radio link when using emergency/power control.

A weakness of the GroundBot is its lack of arms or claws to interact with the environment, making it lack in utility. Thus, outside of surveillance and searching for victims, it cannot complete complex tasks, which would still require the assistance of a human[6].

Section 3: System Architecture



Yellow Box: Hardware
Blue Box: Software
Blue Lines: Relationship between boxes

Figure 1: System architecture diagram of the Alex robot

Raspberry Pi Architecture

Our Raspberry Pi (RPi) is powered by the 3.7V power bank and controls the Arduino Uno and the LiDAR attached through UART (Universal Asynchronous Receiver / Transmitter) communication. The Arduino is used for both movement and colour detection and is hence connected to the motors, encoder sensors (to detect rotation) white LED and RGB colour sensor.

The RPi and the PC are connected to the same wireless network (one of our teammate's mobile hotspot). We've set our mobile hotspot to 2.4Ghz as it has a wavelength that has a longer range and can penetrate solid objects more easily than the 5 GHz band, making it ideal for devices that are taken from room to room or are more distant from the router. Using Transport Layer Security (TLS), the PC sends movement instructions and SLAM-related information to the RPi, and the RPi sends back sensor (LiDAR and colour) data after executing movements or when the PC interrupts or requests for sensor data.

The Arduino Uno and LiDAR are connected to the RPi. The RPi powers the LiDAR, which collects data from its surroundings and feeds it back to the RPi through a USB and UART conversion module. The RPi then sends this data to the Robot Operating System (ROS) master node on the PC. The ROS Visualization (RVIZ), a 3D visualisation tool, processes and displays this data so that the PC operator may see it and navigate through the course. To prevent Alex from colliding with any of the walls, this feature also functions as a proximity detector. The reason for off-loading the ROS from the RPi to the PC is to prevent the RPi chip from overheating which may lead to substantial battery depletion, resulting in less precise control over Alex's movement controls. The RPi's CPU is also not very powerful, and thus moving the Alex too quickly might result in the CPU of the RPi not being fast enough to process the data quickly, which can cause the ROS to crash.

Arduino Uno Architecture

The Arduino Uno, connected to the RPi, performs the function of colour sensing and movement.

The colour sensing is utilised to detect the colour of the obstacle. The Arduino Uno will power on the white LEDs and the respective photodiodes of the colour sensor and to obtain the required colour data. After obtaining the colour, the corresponding music will play to signify a victim has been found.

For movement, the operator inputs the key bind (WASD for the 4 directions of travel) corresponding to the direction, power, and distance we intend for Alex to travel, which is then translated through the Arduino into the respective values required, and subsequently relayed to the motor driver. The motor driver, which is powered by 4 AA batteries, drives the wheels to spin in specific directions. Data, such as how many revolutions the wheel has spun, is captured by the motor encoder. This data is relayed back to the Arduino, and through the RPi to the PC using TLS.

Section 4: Hardware Design

There are different hardware components present on Alex with different functions that complement each other. Figures 2, 3, 4 and 5 showcases the various parts of Alex. The details of each component are explained in Table 1.

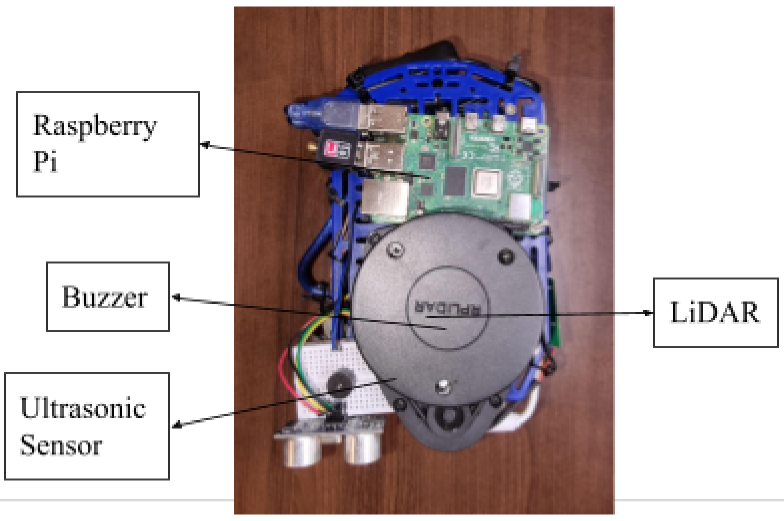


Figure 2: Upper Deck of Alex

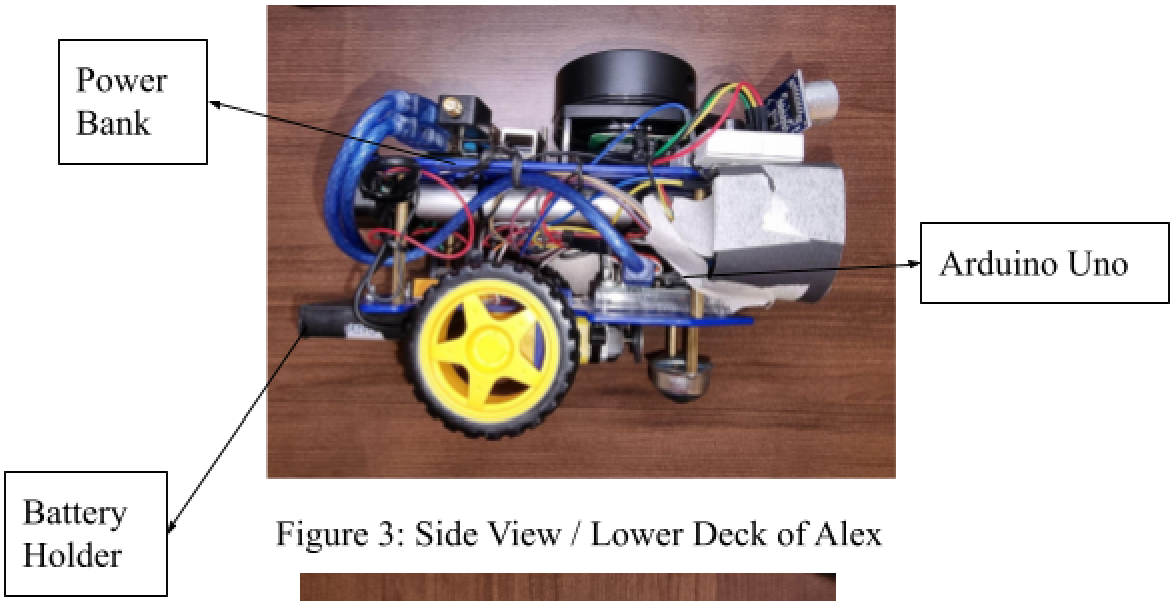


Figure 3: Side View / Lower Deck of Alex

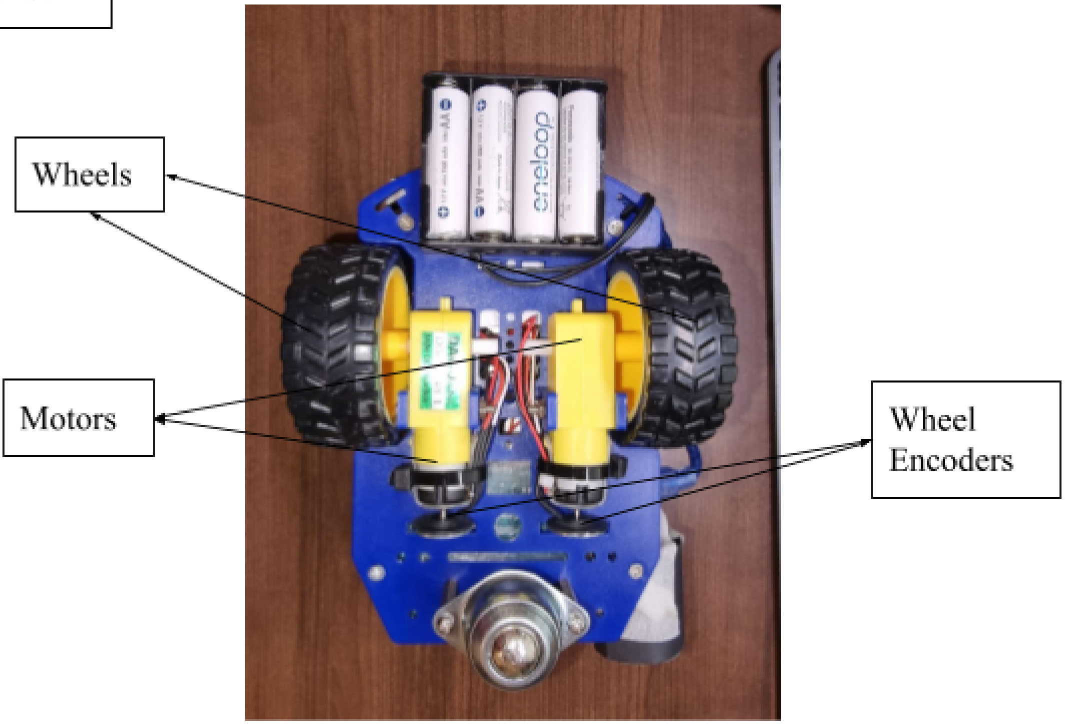


Figure 4: Bottom View of Alex

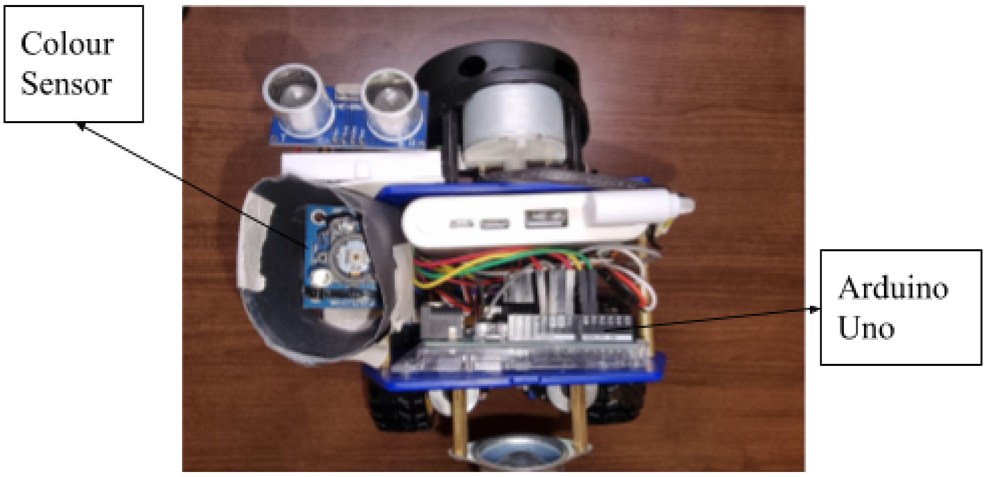


Figure 5: Front View of Alex

Name	Function	Placement
Raspberry Pi	1. Sends information over to the Arduino to control motors, Ultrasonic Sensor and Colour Sensor 2. Transmit power to the LiDAR and Arduino Uno 3. Displays the map data obtained by LiDAR via Rviz	Upper deck of Alex, secured with 2 screws, to allow for easy access when plugging in HDMI cable / accessories
LiDAR	Maps the surrounding environment and transmits data to the Raspberry Pi	Upper deck of Alex to ensure nothing is blocking the sensor
4AA Batteries + Battery Holder	Provides power to the 2 motors	Bottom of Alex, secured with screws, for easy accessibility when changing out batteries
Power Bank	Provides power to the Raspberry Pi	Lower deck of Alex, secured with cable ties
Arduino Uno	Provides instructions from Raspberry Pi and controls the motors, ultrasonic sensor and colour sensor when required	Lower deck of Alex, secured with double-sided tape
2 Wheel Encoders	Measures the degrees of rotation and translate to the distance travelled within the program	Connected to the front of each motor
2 Motors	Creates a continuous rotating motion which allows the wheels to spin hence allowing Alex to move	Lower deck of Alex, secured with screws and locked in place with the chassis.
2 Wheels	Standard wheels to allow for Alex's movement	Connected to the motors
Colour Sensor	Provides colour sensing capabilities to allow Alex to detect the colour of the bottle	Connected to the upper deck and Alex's front, secured by cable ties
Ultrasonic Sensor	Provides a secondary means for Alex to detect distance between Alex and the nearest front-facing obstacle	Connected to the upper deck and Alex's front, mounted directly to the breadboard
Buzzer	Provides a ringtone when detecting any of the two coloured victims	Connected to the upper deck's breadboard.

Table 1: Details of the individual parts of Alex

As seen in figures 2 to 5, we have kept the shape of our Alex compact, by ensuring most of the wires and components are kept within the confines of the chassis. This was an issue initially as the parts of the Alex would get caught up with the walls of the obstacle course. We used a combination of zip ties and twist ties to ensure that the wires are secured tightly and centralised the electrical parts of the Alex. The only wiring we were not able to keep within the body of Alex was the Arduino to RPi USB cable. Doing so ensures that Alex is as compact as possible, limiting the number of collisions. When adding the colour and ultrasonic sensor, we also made sure they were kept within the dimensions of Alex as much as possible. Moreover, we have arranged the parts of Alex as seen above to maintain its stability. Furthermore, the WiFi dongle was removed as it was also very unstable and would move about as Alex was traversing the course, which caused it to clash with the surroundings several times.

Section 5: Firmware Design

There is a communication protocol between the RPi and Arduino through a packet. The packet is read, and dealt accordingly by the Arduino before sending a reply packet. The algorithm is shown in Figure 6 and the communication protocol is shown in Figure 7.

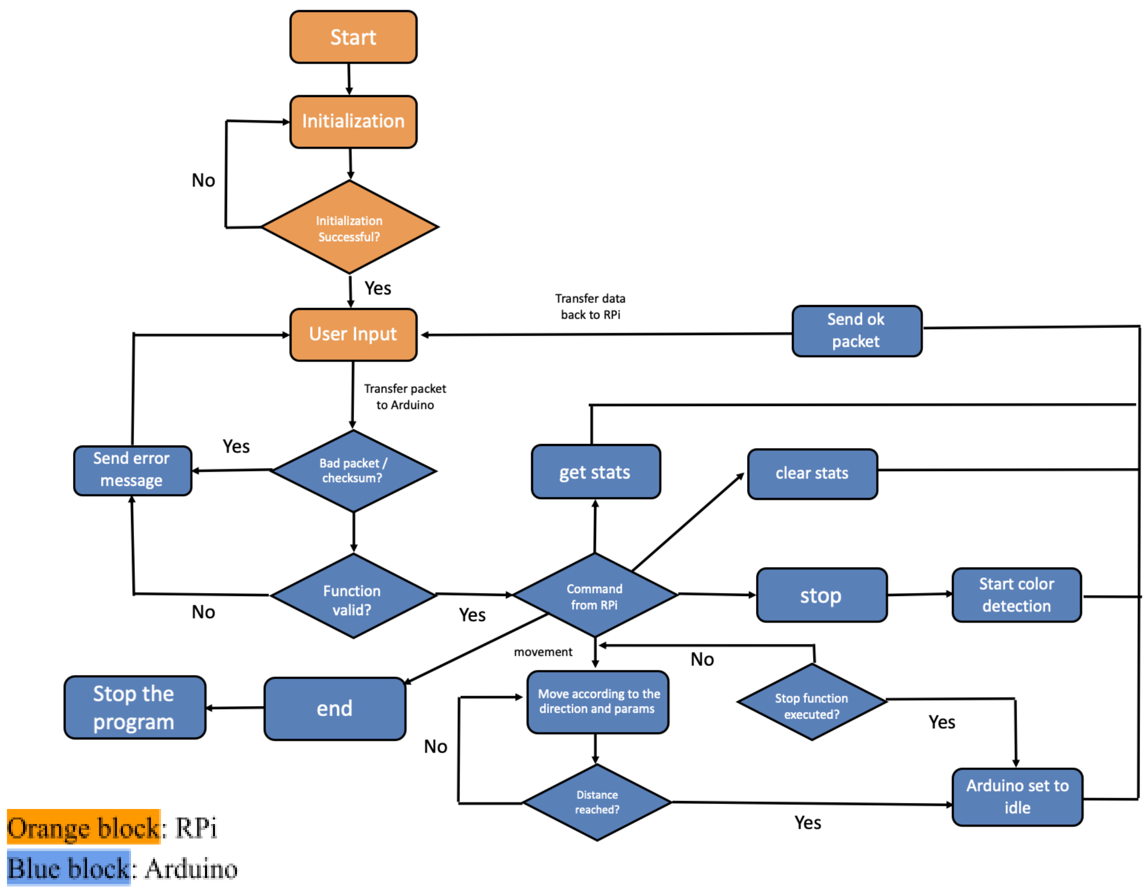


Figure 6: High level steps of Alex

High level steps outline

1. Initialisation
2. Receive user commands
    a. Navigation

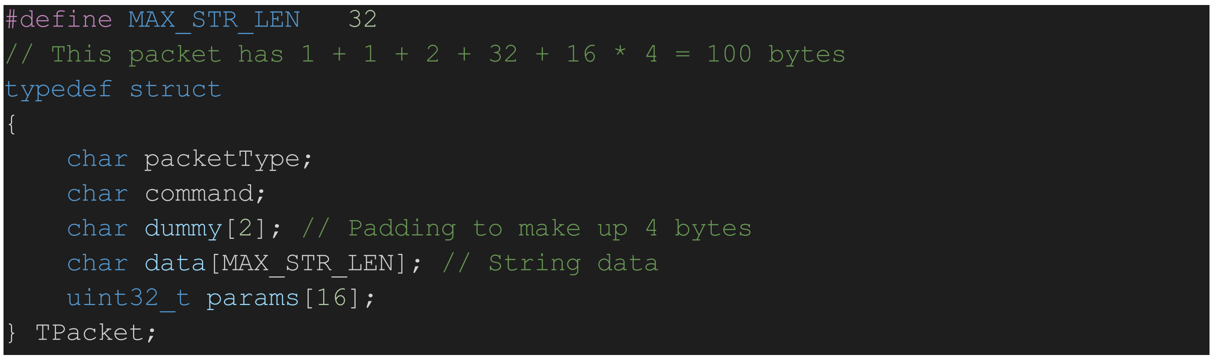
    b. Colour sensor
    c. Play music
    d. Mapping
3. Repeat step 2 until mapping is complete

Initialisation

1. Booting up RPi (Alex), which powers the other devices, including the Arduino
2. Arduino will initialise its necessary sensors, outputs and communication protocols
3. RPi performs a handshake with Arduino by sending a predetermined word (“hello”) and the Arduino, which already has the same word hard-coded, will compare the packet received from the RPi to check if they are the same.
4. If they are the same, the Arduino will reply with a packet containing its own hardcoded word (for example, “world”). The RPi would receive this packet from the Arduino and compare it with its own hardcoded word (“world”). If the 2 words are the same, the RPi will indicate to the operator that the handshake was successful, otherwise the handshake process will restart.

Receive user commands

Commands sent by the user in the form of packets consists of the following structure:



```
#define MAX_STR_LEN   32
// This packet has 1 + 1 + 2 + 32 + 16 * 4 = 100 bytes
typedef struct
{
    char packetType;
    char command;
    char dummy[2]; // Padding to make up 4 bytes
    char data[MAX_STR_LEN]; // String data
    uint32_t params[16];
} TPacket;
```

Figure 7: C++ code for the TPacket Struct

1. “packetType” describes the packet information, which would correspond to 0 for a command type, as stated in constants.h.
2. “command” to store any command given i.e. WASD for movement. “Params” takes in and stores parameters as 32-bit integers such as the distance / degree and power for Alex to move. “dummy” is padding to make up the 4 bytes as shown above to ensure reliable packet size
3. “data” strings point 1 to 3 together to send to the Arduino

These data packets will be received by the Arduino at a baud rate of 9600, with 1 start bit, 8-bit data, no parity bits, and 1 stop bit, which corresponds to a 8N1 data frame.

Arduino will check whether its a valid packet via its magic number before doing a parity byte checksum to ensure the integrity of the transmitted packet. Based on both the magic number and checksum, Arduino will do the following as written in the Alex.ino program:

1. If the Arduino receives a bad packet/checksum, it sends a bad packet / bad checksum to the RPi indicating that the packet is invalid by sending a BadResponse packet (error) and it does not understand the RPi instructions.
2. If both the checks are good, the packet is well received, Arduino will execute the command given by the RPi and send back an "OK" packet as the response.

Section 6: Software Design

Link to Github: https://github.com/yeo-menghan/CG2111A

Navigation

The movement command is entered by the user within the terminal of the RPi, whereby it uses communication protocol and high-level algorithm to send the data to Arduino which will translate the data to the motors to move it. The RPi will receive an acknowledgement when the Arduino receives the message.

The following commands are used to navigate Alex: forward (w), backward (s), turn left (a), turn right (d), stop command (f), go over ramp (r), toggle manual mode (m) to instruct Alex to move in a desired direction.

- For the forward and backward commands, the operator will input the respective direction commands which will cause Alex to move forward or backwards by 5 cm each at 100% power.
- For the left and right commands, the operator will input the respective direction commands which will cause Alex to turn left or right by 10 degrees at 100% power.
- An automatic stop is encoded in the program and sent to stop the motors after the desired turning angle or movement is reached.
- The stop command instructs the Arduino to stop all motors. The Arduino will be in idle mode.
- The go over ramp command instructs Alex to move forward by 30 cm at 100% power.
- The toggle manual mode command toggles between manual and auto mode. Auto mode is the one movement described above. In manual mode, for the forward and backward commands, the operator will input 2 additional parameters after the initial direction command to indicate the distance to travel (in cm) and power (as a percentage out of 100). For the left and right commands, the operator will also input 2 additional parameters after the initial direction command to indicate the degree to turn and power (as a percentage out of 100).

To retrieve the telemetry of the sensors - ultrasonic, colour & encoder, we have the following commands:

- "Get stats" command (g) to retrieve the telemetry of Alex - movement (wheel encoder ticks), for the operator to determine distance moved.
- "Get colour" command (v) will activate the colour sensor and determine the colour of the object sensed.
- "Play music" command (n) will play music depending on the colour detected.

- “Get distance” command (b) will activate the ultrasonic sensor and determine the distance of the nearest obstacle in front of Alex. This feature is implemented as a failsafe for the LiDAR as we realise the LiDAR may not be as responsive to slimmer obstacles and the Rviz does not tell us the exact distance of Alex from the nearest obstacle. Furthermore, it helps with accurate distance sensing when nearing the coloured obstacles as we need to be within a specific distance (10cm) to measure the colour accurately
- “Clear stats” command (c) will clear all stored information on movement and colour sensing on the Arduino. This is essential for the user to complete the mapping task

Colour Sensor

An RGB sensor is connected to the arduino, which will be activated through the “get colour” command. The colour of the object will be determined using the following algorithm:

```
// take 5 readings and return average frequency of Alex's colour sensor
int avgFreq() {
  int reading = 0;
  int total = 0;
  for(int i = 0; i < 5; i++){
   reading = pulseIn(colourOut, LOW);
   total += reading;
   delay(colourAverageDelay);
  }
  return total/5;
}

int determineColour() {
  int redColour = redFreq;
  int greenColour = greenFreq;
  int blueColour = blueFreq;
  if (redColour > 240 && redColour < 300) { //red detected
    if (greenColour > 300) {
      if (blueColour > 200) {
        return 1;
      }
    }
  }
  if (redColour > 290) {  //green detected
    if (greenColour < 300) {
      if (blueColour > 205) {
        return 2;
      }
    }
  }
  return -1;
}
```

Figure 8: Partial C++ code for colour sensor

5 readings are taken for each of the Red, Blue and Green frequencies and averaged out. This average reading will be fed into the determineColour() function to check against the values that our team has calibrated from the field. If a colour profile is matched for either Red or

Green, the arduino will send the corresponding value to the RPi - 1 for red and 2 for green and the RPi will thereafter print either "Red" or "Green" depending on the integer. If a dummy target is detected, a negative integer "-1" is returned and the RPi will print "Invalid".

Play music

A buzzer will be played after detecting the colour of the victim, through the "play music" command. A different music will play corresponding to the colour detected, which are either red or green, to indicate that a victim has been found. If it is a dummy, the buzzer will not play any music. The buzzer itself is mounted on the breadboard on the upper deck of Alex and connected to one of the digital pins on the arduino, and shares a common ground with the ultrasonic sensor.

Mapping

The LiDAR is used to map the environment for us, which constantly transfers data to the RPi. The LiDAR scans the 360-degree environment around Alex and sends the data to the RPi. The RPi publishes this data as a node using ROS to the ROS master. On the operator side, the laptop acting as the ROS master subscribes to the LiDAR data and uses the Simultaneous Localisation and Mapping (SLAM) algorithm, particularly the variant Hector SLAM, to create a 2D map of the area. Hector SLAM uses new observation data and attempts to match it to the known map to figure out where Alex is. The map is then visualised using Rviz, as shown in Figure 9, to obtain the final environment mapping. The user can note down the surroundings and move Alex appropriately.

Furthermore, to minimise the risk of Alex bumping with the obstacles, we adjusted the dimensions of the default "Pose" arrow from Rviz to the following settings: Shaft Length = 0.9, Shaft Radius = 0, Head Length = 0.15, Head Radius = 0.2. We have adjusted the size of the arrow to be proportional to Alex's actual dimension, however as we are limited to the shape of the triangle, the sides of Alex may come in contact with the obstacles.

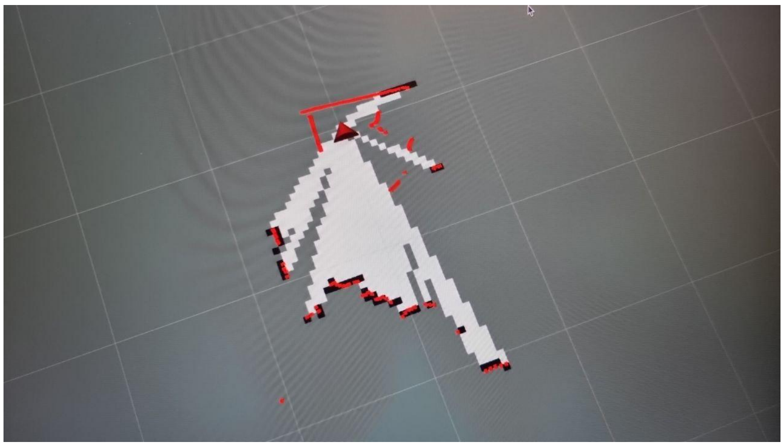


Figure 9: Visualisation from RViz

Section 7: Lessons Learnt - Conclusion

We’ve learnt that time management is crucial. Although we planned a robust timeline, we were not able to follow it after many instances of unknown and unfixable bugs that took up too much of our time - we ended up spending way too much time fixing bugs as opposed to implementing new features and fine-tuning them. Hence, it is vastly important to set aside buffer time within our timeline to account for the time spent on fixing bugs.

One more important takeaway from this lesson is to learn how to seek an expert’s help after a certain amount of time spent on trying to fix the bug - it is often times better to ask the TA / professor for help if we’ve tried our best in trying to resolve the problem; the solution may be something obvious or that the TA / professor have experienced something similar and can provide an immediate solution.

One major mistake that we had was not practising sufficiently for the run under timed conditions. Throughout our lab sessions, we are always testing the minute details - whether the motors can rotate to our liking, trying out how to make a robot mesh out on RVIZ and many other smaller things. However, we failed to consider the human factor in controlling and coordinating as a team for the actual run. As a result of not sufficient practice, we are not decisive and wise enough to prioritise our order of operations and we failed to park Alex in the end due to not having sufficient time.

Lastly, we felt that we could have spent more time on innovative improvements such as 3D printing our mounts for the colour sensor and ultrasonic sensor, experimenting more with the RVIZ platform to fine-tune the signature of our robot, understanding the intricacies of the code behind to make it less error-prone and more responsive to our control inputs etc. This happened because we followed too closely with the rubrics and did not incorporate sufficient time to play around with additional features.

Reference

[1]“Talon tracked military robot,” *Army Technology*, 04-Aug-2020. [Online]. Available: https://www.army-technology.com/projects/talon-tracked-military-robot/. [Accessed: 30-Mar-2023].

[2]QinetiQ “TALON Modular CBRNE Kit.” [Online]. https://www.epequip.com/wp-content/uploads/Product_Datasheet/Unamnned_Ground_Systems/UGV_Talon-HAZMAT.pdf. [Accessed: 29-Mar-2023].

[3]K. Berns, A. Nezhadfard, M. Tosa, H. Balta, and G. D. Cubber, “Unmanned Ground Robots for rescue tasks,” *Search and Rescue Robotics - From Theory to Practice*, 2017.

[4]“Rotundus GroundBot spherical surveillance robot broadcasts live in 3D,” *New Atlas*, 02-May-2015. [Online]. Available: https://newatlas.com/rotundus-groundbot/20259/. [Accessed: 29-Mar-2023].

[5]“Specifications” *Rotundus*. [Online]. Available: https://rotundus.se/specifications/. [Accessed: 29-Mar-2023].

[6]“FAQ,” *Rotundus*. [Online]. Available: https://rotundus.se/faq/. [Accessed: 30-Mar-2023].