CG2111A Engineering Principle and Practice II
Semester 2 2023/2024

"Alex to the Rescue" Design Report Team: B05-6A

Name	Student #	Sub-Team	Role
Thiru Vageesan	A0269930U	Software	Software Lead
Chen Yiyang	A0271796M	Hardware	Procurement & Hardware Lead
Tay Guang Sheng	A0273178W	Software	Software Programmer
Zhong Shu	A0281729U	Software	Software Debugger
Tong Jia Jun	A0271852B	Hardware	Hardware & Report Editor

Table of Contents

Section 1 System Functionalities

1.1. Introduction

Alex is a teleoperated robotic vehicle with search and rescue functionalities that aims to play a vital role in improving the speed and accuracy of search and rescues, as well as saving lives without risking lives by replacing human rescuers with robotic ones. Equipped with an RPLidar sensor, and the ability to be controlled remotely, the operator can control Alex remotely to navigate the map to search for victims.

1.2. Core Functionalities

Alex's main functionalities can be split into two categories: [1] Movement, and [2] Map Generation.

Movement

Our Alex will be tele-operated from our laptop. We will be able to transfer data wirelessly from our laptop to the Pi via SSH connection. After that, these commands will be processed and translated into actual movement control signals to the Arduino via a serial connection, allowing us to remotely control the Alex. Our Alex will be able to go forward, backwards and turn left/right.

Map Generation

The mounted Lidar will be utilised to scan the surrounding environment of the Alex, with the resulting data packets in the format [timestamp] [angle, distance] being published by its driver node. We will then utilize Hector Slam to subscribe to these data packets and generate a map of the world around Alex. Finally, Rviz will be utilized to subscribe to the Hector Slam node to visualize its output, which we will then view through a VNC connection to the Pi. While other SLAM algorithms might provide more accurate results, we are limited by our lack of other measurement devices to collect odometry data.

1.3. Additional Functionalities

Our Alex's additional functionalities include its ability to sense colour and distance as well as a siren to alert potential victims to Alex's location.

Detection and Avoidance of Obstacles

The RPLidar sensor will be utilised to identify the distance Alex is away from obstacles from all angles. This feature will allow the teleoperator to understand where the obstacles are, and enable the robot to move around the objects while avoiding them.

Colour Sensing

Alex will be able to sense the colours of the 2-3 regular-shaped ~18cm tall objects scattered throughout the rooms. When these objects are detected during navigation, their colours will be identified via Alex's colour sensors. They will be categorised as either healthy victims (green), injured victims (red) or neither through a colour detection algorithm.

Distance Sensing

We supplemented our Alex with an ultrasonic sensor at the front to more accurately gauge the distance of objects in front of us. This is mainly to complement our colour sensing system as the Lidar mapping will not be able to tell us the concrete distance between Alex and the object. The ultrasonic sensor allows us to navigate our Alex closer towards the object and get more accurate and reliable colour readings.

Siren and Communication Device (Microphone)

Alex will be equipped with a speaker. The siren will simulate how emergency vehicles in real life have loudspeakers to inform surrounding survivors of the situation.

Section 2 Review of State of the Art

2.1. PackBot 525

Background & Components. Packbot is a teleoperated robot used for bomb disposal and in hazardous environments. PackBot features a tracked mobility system for diverse terrains, a manipulator arm, and buoyancy control for water operations. It has cameras, laser rangefinders, and chemical sensors for situational awareness, powered by batteries with onboard computers. Its modular design allows for easy customization, making it adaptable for military and law enforcement use. PackBot's software includes an operator control unit (OCU) for teleoperation and real-time sensor feedback. It also has autonomous features for tasks like waypoint navigation and obstacle avoidance, functioning without direct human control. Refer to **Annex A**.

Strength

1. Operating the PackBot remotely from a safe distance mitigates the risk of injury or fatality in hazardous environments.
2. The PackBot offers real-time feedback through its sensors and cameras, enabling operators to receive immediate information.

Weakness

1. The PackBot's performance in remote and inaccessible areas may be hindered by a limited communication range, particularly in regions lacking communication infrastructure.
2. The robot's effectiveness is limited in certain tasks due to its inability to lift objects that are too heavy or large, indicating limited manipulation capabilities.

2.2. Spot, Boston Dynamics Robot

Background & Components. Spot, developed by Boston Dynamics, is a quadruped robot designed for remote operations, capable of navigating diverse terrains, including hazardous environments and disaster areas. Its flexible legs allow it to traverse challenging terrain, collecting detailed 2D and 3D information with onboard sensors. This data is essential for inspections, surveys, and creating detailed maps or models of environments. Spot's hardware components include sensors like cameras, LIDAR, and Inertial Measuring units (IMU). It also contains a payload mounting system that allows additional sensors, tools or equipment to be attached to the robot, to customise Spot's capabilities for different tasks and applications. The software is designed to perform a range of tasks, from basic navigation to complex item-evolving manipulation and decision-making. One of Spot's key software components is Spot API, which enables applications to control Spot and access sensor information. The API uses a client-server model, where communication with services running on Spot occurs over a network connection. Refer to **Annex A**.

Strength

1. Agility. Spot's four-leg design enables it to navigate diverse terrain with ease, making it suitable for various environments.
2. Versatility. Spot's modular design allows for easy customisation with different payloads, sensors and tools, making it highly customisable so as to adapt to a wide range of tasks and application

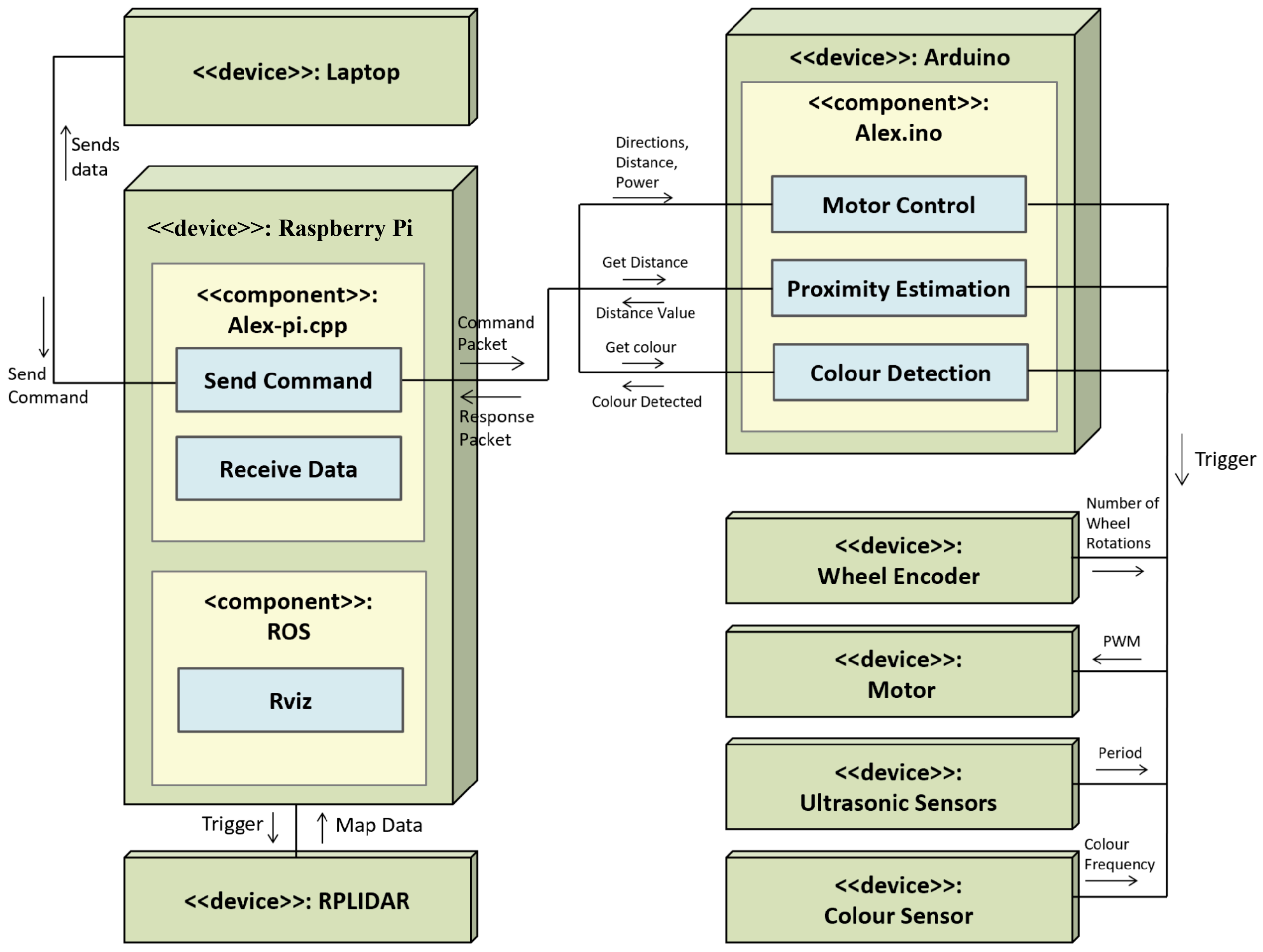
Weakness

1. Payload Capacity. While Spot can carry various payloads, its weight capacity is more limited compared to larger robots, which may restrict the types of tasks it can perform.
2. Battery Life. Spot Battery provides power for about 90 minutes of normal operation, hence it needs to be recharged regularly.

Section 3 System Architecture

There are a total of 8 devices in Alex, namely Arduino Uno, Raspberry Pi, LIDAR, Motors, Wheel Encoders, Colour Sensor, Ultrasonic Sensor and the laptop of the teleoperator. The System Architecture of Alex is detailed in **Figure 1**.

Figure 1: System Architecture



Section 4 Component Design

4.1. High-Level Algorithm

1. Initialise communication between Raspberry Pi, Arduino and PC
2. Generating Map using Hector SLAM, visualized with Rviz
3. Pi collects user input and relays the corresponding command to Arduino
4. Arduino processes and executes the command
5. Repeat Steps 2 - 4 until navigation is complete

4.2. Further Breakdown

<u>Step 1: Initialise Communication between Raspberry Pi, Arduino and PC</u>

1. The Pi initiates serial communication with the PC using the TCP/IP protocol over Wifi. This enables remote access to the Pi from the PC.
2. The Pi calls the function startSerial() function to establish serial communication with the Arduino at a baud rate of 9600 bps and using the 8N1 frame format.
3. Pi creates a receiver thread to receive packets from Arduino.
4. Pi creates and serialises a “Hello” packet within the TComms packet and sends it out to the Arduino. The Hello packet contains information specific to the application vehicle the TComms packet includes additional overhead information.
5. Arduino receives the packet and polls to check if the packet is complete. If the packet is incomplete, Arduino waits for the packet to be complete.
6. Arduino deserialises the TComms packet and checks the packet for errors. If the packet is invalid with a bad magic number or wrong checksum, it will return an error packet namely RESP_BAD_PACKET and RESP_BAD_CHECKSUM to Pi.
7. If the packet has no errors, Arduino will send an OK packet back to Pi to confirm that the command was received.

<u>Step 2: Generating Map using Slam</u>

1. LIDAR scans its surroundings, capturing distance data to obstacles in all directions. This data in the form [timestamp][angle, distance] is published by its driver node.
2. The Slam node subscribes to the driver node and uses the Hector Slam algorithm to generate and publish a map which represents the environment detected by LIDAR.
3. The rviz node subscribes to the Slam node and uses the incoming data to continuously visualize the generated map for the user to see.

<u>Step 3: Pi Collects User Inputs and Relay the command to Arduino</u>

1. The user operating from the PC enters a key on the keyboard connected to the PC via a USB port. The character of the pressed key will be sent from the PC to the shell program on Pi via Wifi.
2. It maps user commands to commands defined in the communication protocol between the Pi and Arduino.
3. The command is written to the packet and will be serialised.
4. Pi will send this serialised packet to the Arduino via serial communication.
5. At the Arduino, it deserialises the packet. If the data is error-free, the Arduino sends an “OK” packet back to the Pi, indicating that it is ready to carry out the command. If the Arduino detects any errors, it sends an “Error” packet back to the Pi, ensuring that the wrong command will not be executed.

<u>Step 4: Arduino Processes and Execute the Command</u>

Arduino executes the command accordingly. During execution, the shell waits for the command character, which is in the key link to a certain command. The following are the commands present in Alex.

Movement

We have 2 modes for movement, namely manual and auto. In manual mode, the speed requires 3 parameters, Speed, Distance and Direction. The user would need to key in the abovementioned parameters in sequence followed by an enter and the robot will move in the particular direction at that speed and distance. After that, Alex will stop and wait for the next command.

Table 1: Movement Manual Mode

Parameter 1 Speed	Char	Parameter 2 Distance	Char	Parameter 3 Direction	Char
5cm/s	1	5cm	4	Forward	W
10cm/s	2	10cm	5	Left	A
15cm/s	3	15cm	6	Backwards	S
				Right	D
				Stop	F

Instead of the manual commands where the shell would expect a string of characters (3 parameters: speed, distance, direction) to execute the command, we have implemented an auto mode. In Auto mode, we only provide shell with a single parameter: Direction. The default movement configuration is set to auto where the Speed and Distance are fixed. In this case, the speed and distance will correspond to the 5cm/s and 5cm respectively. The user only needs to key in the direction key (WASD) which corresponds to Forward, Left, Backwards and Right for Alex to move. When a different key is keyed, Alex will change its direction and move in that direction.

Data Querying

The char for Data Querying is E. Upon receiving the data query command from the teleoperator, the Arduino will send data packets to the Pi. These packets contain information such as the rotation count of each wheel, colour values scanned by the colour scanner and ultrasonic sensor. This will be displayed on the shell running on the PC. The Pi will format the data in the data packet to first display the wheel information and then the colour information on the terminal.

Table 2: Data Querying

Wheel Encoder	1. Ardunio creates a response packet containing the leftForwardTicks, and rightForward Ticks. 2. This response packet will be serialised into a TComms packet and then sent back to the Pi via UART. 3. Pi prints it on the terminal.
Colour Sensor	1. Arduino configures the colour sensors to read the red and green frequency of the colour paper. 2. By comparing the values of the red and green frequency of the paper, Adruino determines the colour of the paper. This means when the red frequency > green frequency, the colour is red, and vice versa. 3. Arduino creates a response packet with a colour code. In this case,

	0 is red and 1 is green. 4. This response packet will be serialised into a TComms packet and then sent back to the Pi via UART. 5. Pi prints it on the terminal.
Ultrasonic Sensor	1. To send the pulse, Arduino sets the trigger pin 2. To read the echo signal and get the period, Arduino sets the echo pin to high. 3. Using the following formula, Arduino converts the period to the distance between the ultrasonic sensor to the wall. $$Distance = \frac{Speed\ of\ Sound\ x\ Time\ Taken}{2}$$ 4. Ardunio creates a response packet containing the leftForwardTicks, and rightForward Ticks. 5. This response packet will be serialised into a TComms packet and then sent back to the Pi via UART. 6. Pi prints it on the terminal.

Data Clearing

The char for Data Clearing is R. Upon receiving the data clearing command from the teleoperator, the Arduino will overwrite all data values stored in the data array to 0.

Terminate Serial Communications with Arduino

The char for terminating serial communication with Arduino is Q.

Section 5 Project Plan

Week	Deliverables / Milestones
8	1. Assemble the main chassis of Alex along with the motors 2. Plan the placement of hardware components in the chassis of Alex and the circuitry layout 3. Set up and test the wheel encoders 4. Ensure the motors are working properly 5. Test Alex's directional capabilities using the Arduino library
9	1. Set up Arduino Libraries on the Pi 2. Resolve Cross-Platform Communications 3. Calibrate the wheel encoders and the code to give accurate directional commands to Alex through inputs from the Serial Monitor 4. Install the colour sensor onto Alex
10	Design Report Due (31 March 2024) 1. Setup ROS and SLAM on the pi for Lidar mapping
11	1. Set up ultrasonic sensor onto Alex 2. Calibrate colour sensor to accurately detect colour 3. Calibrate ultrasonic sensor to give accurate distances 4. Review of Design Report Feedback
12	1. Further refinement of Alex's movement capabilities 2. Add cool features to Alex
13	Trial Run + Final Demo 1. Ensure Alex can complete tasks within the given time successfully. 2. Final adjustments to any software and/or hardware components
Reading Week	Final Report Due (26 Apil 2024) 1. Conduct after-action review for final demo and complete final report for submission

Reference

1. Boston Dynamics. (2024). *Spot Specifications*. Boston Dynamics Support Center. Retrieved March 31, 2024, from https://support.bostondynamics.com/s/article/Robot-specifications
2. Boston Dynamics Support. (2024). *Spot anatomy*. Boston Dynamics Support Center. Retrieved March 31, 2024, from https://support.bostondynamics.com/s/article/Spot-anatomy
3. ELP. (2023). *PackBot 525 remote manipulation vehicle*. ELP GmbH. Retrieved March 31, 2024, from https://www.elp-gmbh.com/en/produkte/teledyne-flir-packbot-525-rremote-manipulation-vehicle/

Annex A

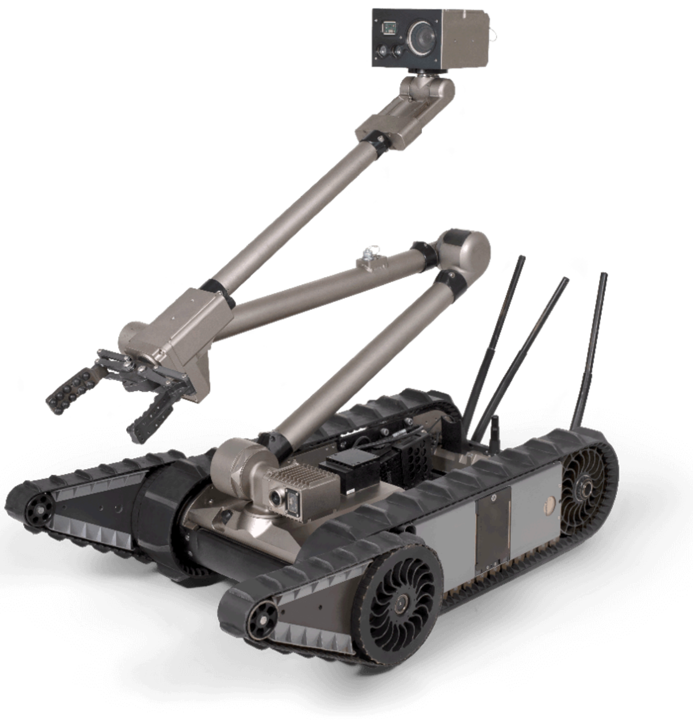


Figure A-1: Image of Pack Bot 525

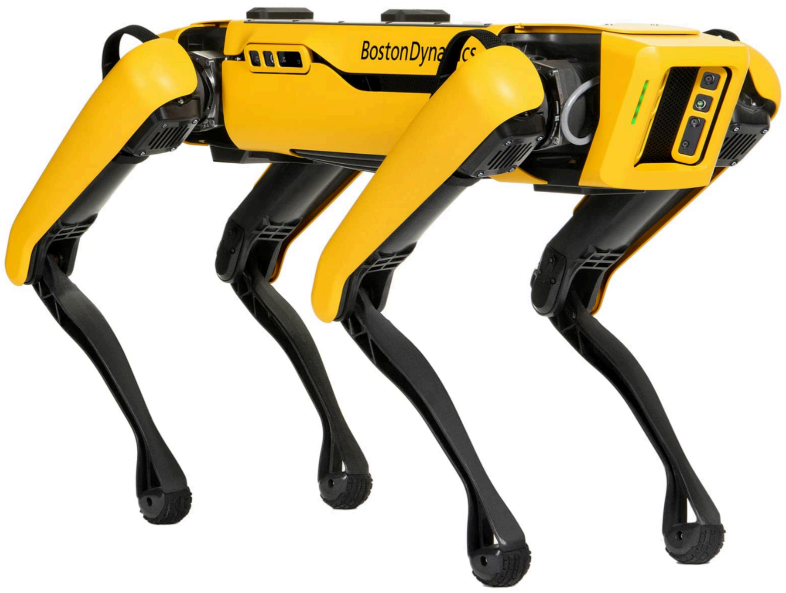


Figure A-2: Image of Spot