CG2111A Engineering Principle and Practice

Semester 2 2022/2023

“Alex to the Rescue”

Final Report

Team: B04-4A

Name	Student ID	Role
Siew Shui Hon	A0258613B	Circuitry
Zhou Junmin	A0257966H	Lidar
Tay Pong Hee	A0258125E	Motor Control
Tan Yi Xin	A0252733H	Serial Comms

Section 1 Introduction

The Alex robot comprises an Arduino Uno and a Raspberry Pi 4, along with 2 motors and a LIDAR system mounted to the chassis. The Alex robot will be controlled through laptop commands sent via WIFI through a Virtual Network Computing (VNC) server, in which the data would be sent and received through the Pi. The Pi forwards these commands to the Arduino and the Lidar, to move the robot and get information about the surroundings. The objective of the robot is to navigate and map an area using the information from the LIDAR, and subsequently finish in a 1 by 1 “parking” spot in the “maze”. A colour sensor was added to identify 2 objects of Red and Green placed throughout the area, simulating the earthquake victims. Lastly, a PS4 controller is used to send commands to the laptop for movement controls.

Section 2 Review of State of the Art

Teledyne Flir - SkyRanger:

The SkyRanger is an aerial drone used in aerial search and rescue operations [1]. It is equipped with infrared and daylight cameras that can identify human signs from a bird's eye view. It was designed to be customised by the end user, allowing up to 3.5kg additional weight of equipment, such as gas and radiation detectors, with a software development kit provided to customers to provide custom use cases.

Figure 1: SkyRanger Drone [1]

Strength:

- It is not affected by rough terrain due to aerial based travel.
- Customizable to suit the user's needs.
- Fast and efficient at covering areas.

Weaknesses:

- Only usable in outdoor circumstances, as the drone is unable to travel into rubble and debris.
- Field of vision of its normal camera may be blocked in heavy forested locations.
- Performance is subject to weather conditions, such as wind speeds and turbulence.

Chiba Institute of Technology fuRo - KOHGA3:

The KOHGA3 is a ground robot designed for search and rescue operations in disaster situations, particularly earthquakes. [2] It operates on tracks instead of wheels and is hence able to traverse over rubble and ground debris. It also has a range of sensors, including cameras, microphones, and gas sensors, which allow it to collect information about its environment and detect the presence of survivors. Audio messages can also be transmitted from the operator to the robot to communicate with victims wirelessly. [3]

Strength:

- Able to navigate indoors and over small obstacles or debris, common inside earthquake buildings
- Able to map robot's environment with high resolution
- Allows effective remote communication with victims, hence protecting the operator from unnecessary environmental risk

Weaknesses:

- Cannot traverse over physical obstacles that are significantly bigger than the robot. Less outdoors manoeuvrability than aerial drones.
- Not optimised for tsunami-related rescue operations. Effectiveness is rather limited to inspecting rubbles and damaged structures indoors.

Figure 2: KOHGA3 Robot [2]

Section 3 System Architecture

The Alex robot comprises a wheel encoder, Lidar, colour sensor, Arduino Uno, Raspberry Pi 4 and 2 motors, in which they will communicate through UART, Wifi communications and wires. In order for the Alex to move, the commands would be sent by a PS4 controller and then transmitted remotely through VNC from a laptop, and the components above would need to work in tandem for these commands to be processed correctly. Lastly, the readings from the Lidar would need to be sent back to the laptop for the user to know what the overall maze looks like. The overall algorithm in how the components of the Alex robot works in tandem would be explained in this section.

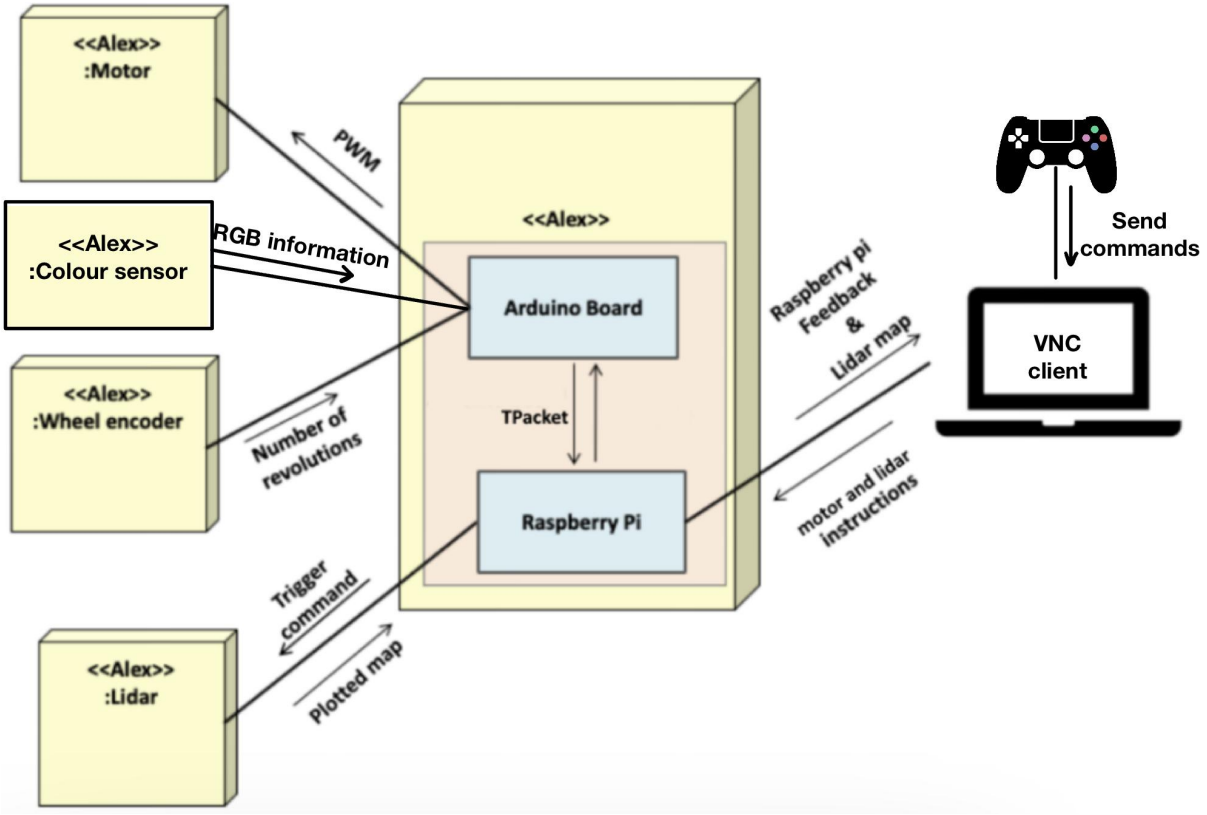

Figure 3: Overview of interactions between devices in the system architecture

PS4 controller & Laptop

The user would use the PS4 controller with buttons and joysticks mapping to specific combinations of keyboard keys, to send commands to the laptop. The controller is connected via BlueTooth which enables us to send commands even when we are away from the keyboard.

Laptop & Pi

The laptop would communicate with the Pi through wifi by VNC, in which data can be transmitted and received on both ends.

Pi & Arduino

The Raspberry Pi 4 and Arduino Uno are both mounted on the Alex chassis. These 2 components are connected via a USB cable, with a transmission of information through UART serial communication. Commands such as movement would be sent from the Pi to the Arduino, and acknowledgment would be sent back from the Arduino to the Pi if the data received is correct.

Pi & Lidar

The Lidar would be directly connected to the Pi, in which the data that was read by the Lidar would be sent back to the Pi to plot a map on the gnuplot interface whenever the robot is stationary. This data would then be displayed through a VNC viewer, whereby the operator would thus be able to draw a map of the robot's environment remotely.

Arduino & Motor

Commands received by the Arduino through the Pi would be sent to the DRV-8833 Motor Driver Carrier on a circuit board, which in the pins would be connected as shown below:

Pin	Description	Connect to:
VIN	Motor Power Supply VCC	+ screw of the 2.1mm female jack
GND (right side)	Motor Power Supply GND	- screw of the 2.1mm female jack
BOUT1	Motor 2 +ve	Red wire from RIGHT motor
BOUT2	Motor 2 -ve	Black wire from RIGHT motor
AOUT1	Motor 1 +ve	Red wire from LEFT motor
AOUT2	Motor 1 -ve	Black wire from LEFT motor
GND (left side)	GND from Microcontroller	Arduino pin GND
BIN1	Motor 2 control	Arduino pin 10
BIN2	Motor 2 control	Arduino pin 11
AIN1	Motor 1 control	Arduino pin 5
AIN2	Motor 1 control	Arduino pin 6

This DVR chip would then supply the necessary power to the motors to allow the robot to move.

Arduino & Wheel Encoder

The wheel encoders mounted on the left and right motors would have the white wires connected to the PINS 2 and 3 of the arduino respectively, which corresponds to the interrupt ports on the Arduino. Whenever an interrupt is being triggered, the variable “ticks” on either motor would be incremented, which would be processed by the Arduino to calculate the distance travelled or angle turned by Alex.

Arduino & Colour sensor

LEDs from the colour sensor would be powered by the Arduino board, and then Arduino sends pulseIn commands to get RGB readings from the colour sensor. The pin connections from the colour sensor to the Arduino UNO is shown below:

Pin	Description	Connect to:
VCC	Colour Sensor Power Supply VCC	Arduino pin 5V
GND	Colour Sensor Power Supply GND	Arduino pin GND
S0	Output Frequency Scaling Multiplier	Arduino pin A3
S1	Output Frequency Scaling Multiplier	Arduino pin A2
S2	Photodiode Type Multiplier	Arduino pin A1
S3	Photodiode Type Multiplier	Arduino pin A5
OUT	Output pin to the Arduino UNO	Arduino pin A4
LED	LED control	Arduino pin A0

Section 4 Hardware Design

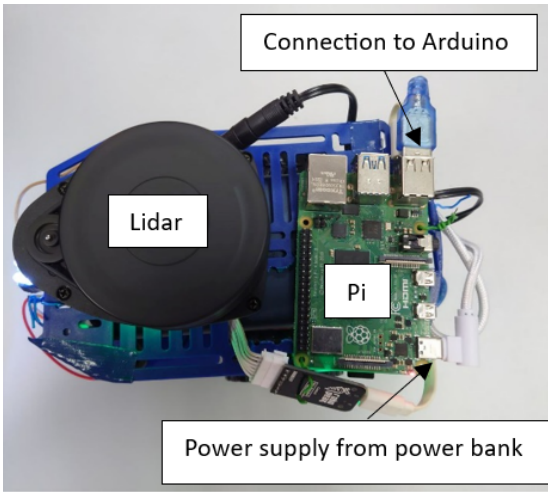

Figure 4: Bird's Eye view of Alex

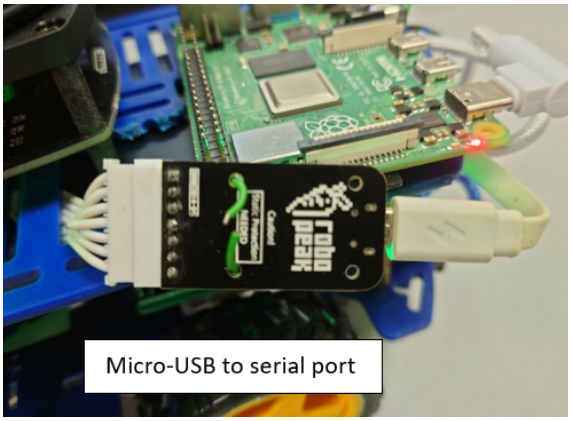

Figure 5: Close-up of serial connection

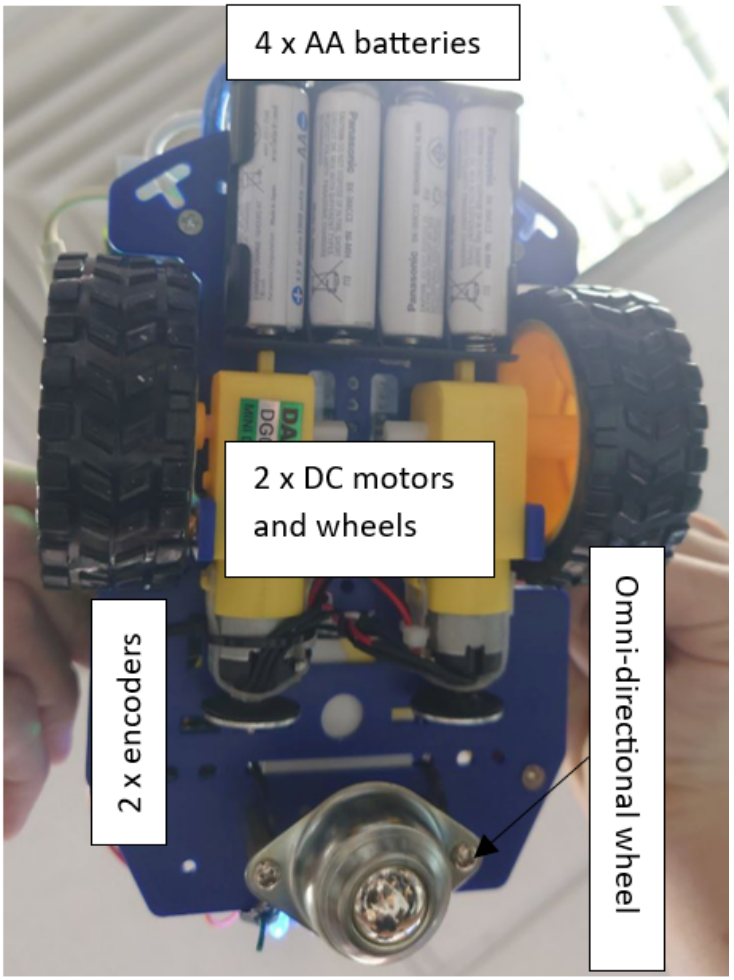

Figure 6: Bottom view of Alex

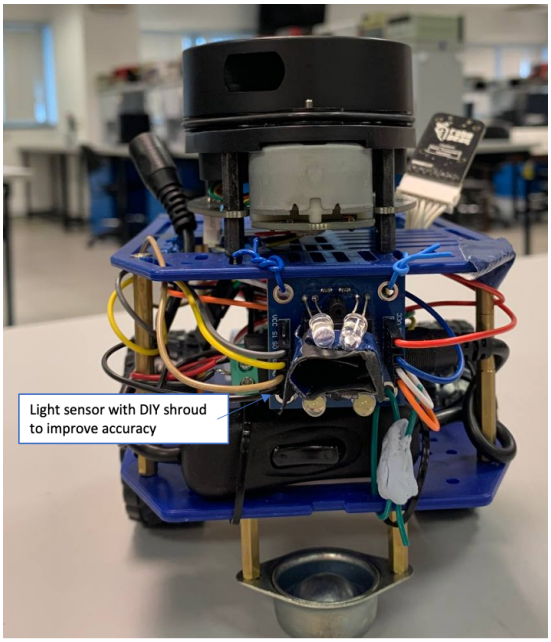

Figure 7: Front view of Alex

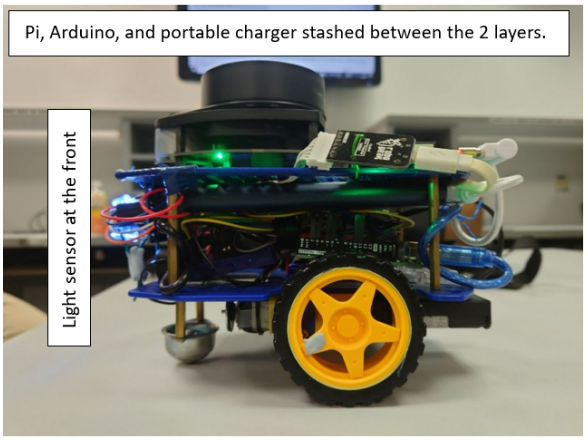

Figure 8: Alex's Side profile

Section 5 Firmware Design

High Level Steps:

1. Arduino Algorithms
2. Communication Protocol

Arduino Algorithms

1. Arduino receives a command from pi.
2. If the command is valid, the pi returns an acknowledgement packet.
3. If not a movement command, it will execute the clearing, or retrieval of the wheel encoder ticks, and send it back if required with the colour sensor values.
4. If a movement command, the Arduino will start the motors, and keep checking the encoder ticks, until the total number of ticks has gone past a calculated number of ticks, which corresponds to the distance given in the command.
5. To account for ticks occurring whenever the wheel turns, the appropriate counter only increases when the corresponding direction is active.

Communication Protocol

Each command the Arduino and Pi sent to each other was in the form of a predetermined packet. The first packet, the command packet, is formed via user inputs or the sensors connected to the arduino.

Component	Packet Type	Command	Dummy padding	Data	Parameters
Size / Byte	1	1	2	32	64

Afterwards, the sending side converts this packet into a standardised serialised form as seen below. The command packet is copied into the buffer component of the communications packet.

Component	Magic Number	Data Size of packer in Buffer	Buffer (where the command packet is)	Checksum	Dummy Padding to standardise compiler behaviour.
Size / Byte	4	4	128	1	3

This communications packet is sent over the serial port to the receiving side, which would first check for the Magic Number. If valid, it would then assemble the packets and check for the checksum, returning an error if there are any missing or wrong bytes. If the receiving side is the Arduino, it would also send an acknowledgement packet back to the pi.

Section 6 Software Design

High Level Steps:

1. Motor control
2. Colour sensing
3. Controller to Laptop Communication
4. VNC to Pi communication
5. Lidar Mapping of surrounding obstacles
6. Conversion of Timers and UART into bare metal code

Motor control

For the motors to move, we had our PS4 controller send commands to the laptop via BlueTooth. In total, we had 11 functions on the PS4 controller.

1. Basic movements: The 4 directional buttons send the command to move forward/backwards for a specified distance of 5cm or pivot left/right 30°. These actions are at 90% output power.
2. “Fine-tune” movements: Similarly, the 4 buttons here send commands to move forward/backwards for 2.5cm or pivot left/right for 15°. These are at 90% output power.
3. Turbo: The right joystick activates a “turbo” function for forward and backward movement, for a distance of 20cm and at 100% output power.
4. Get-status: The left-back button (LB) is used for the get-status command, which returns the individual RGB values of the environment and the victim that we are trying to identify.

Colour sensing

The colour sensor returns the period of the light detected via each of the diodes, Clear, Red, Green and Blue, depending on the logic written to pins S2 and S3. The Arduino then returns the ratio of the individual RGB periods to the clear periods to the user, which can then be read as a distinct colour. The colour values were not hard-coded to account for different lighting conditions.

Controller to Laptop Communication:

1. The controller is connected to the laptop via BlueTooth.
2. Using a software we found on GitHub called AntiMicro, we were able to map macros from the keyboard to the buttons on the PlayStation 4 controller.
3. The Marcos includes a fixed distance and power output for each command.
4. These macros mapped to keys on the controller helps us to execute commands on the VNC, such as forward, turning and get status.

VNC to Pi Communication:

1. The Pi communicates to the laptop via a VNC viewer.
2. When an input from the laptop is sent to the Pi, either the Arduino or the Lidar would be triggered.
3. If there are no inputs being sent from the laptop, the Pi would operate normally and run any software that has been configured to it prior, with the VNC server running in the background of the Pi.

Lidar mapping of surrounding obstacles

1. The user starts the ROS programs configured to communicate with the lidar.
2. Rviz is launched and the surroundings are mapped by the data returned from the lidar through the use of Hector-slam.
3. The map is updated whenever movement is detected. The software tries to predict Alex's new position based on previous surrounding data.
4. The arrow given in the Rviz program meant to correspond to Alex’s current position had its parameters adjusted such that it formed the “hit-box” of the robot. This way, the operator could be very certain about not hitting any obstacles.

Conversion of Timers and UART into bare metal code

1. The bare metal code can be found in the other file labelled baremetal.ino.

Section 7 Lessons Learnt - Conclusion

One lesson we learnt was the importance of foresight and pre-planning Alex's build. Whilst it is enticing to jump straight into building the robot (just like lego), doing so would mean that dismantling and repetitively rebuilding it again should we decide to make subsequent design changes or implement new features. This appeared evident when we subsequently implemented our colour sensor. To make the connections to the arduino board, minimally the top half of Alex had to be dismantled. In terms of pre-planning, we also preemptively attached jumper cables to the arduino for further connections or alterations to be made after assembly. These choices minimised the time wasted due to inefficiencies.

Another lesson was the importance of practice and thorough calibration with Alex. During the lab sessions leading up to the final run. We extensively simulated the project conditions and had many practice runs in navigating the maze. We even reorganised the maze to make our own trial runs harder. This helped us on the actual project run during which we felt more comfortable navigating with Alex, allowing us to navigate maze without as much unfamiliarity and park the Alex within the time limit.

One of the mistakes our group made was not rewiring the motors to a different port before assembling the rest of the board. By following the assembly guide provided, we had to implement all 3 timers of the Atmega328p chip, which required more code than if we changed one of the pins of the motor controllers. If we changed the motor pin on pin 11 to pin 9, we would only have had to set up 2 timers, which would save us time in setting up the PWM and debugging our bare metal implementation later on. However, this was relatively minor and easily fixed without extensive changes on design and software code.

Another area for improvement is not being too tunnel-visioned on test requirements. We were very focused on each specific objective in the assessment procedure and altered our approach solely to meet these requirements, failing to take into account the bigger picture of Alex being a rescue robot. For example, whilst we extensively tested the light sensor and colour detection algorithm, we hyper-focused on recognising only the specific colours we tested in this particular lighting condition. This lack of flexibility was reflected during our actual test run where an unknown shade of green was utilised and we failed to confidently identify it. This was attributed to a difference in ambient lighting conditions during the actual run as well. We realised that under real practical environments, conditions differ greatly from that of the practice environment. Our project goals should not become specific metrics and we ought to maintain a broad perspective in the development of future projects, keeping in mind the actual end-goal in the first place.

References

[1] Teledyne Flir, “SkyRanger R70”, [Online]. Available: https://www.flir.asia/products/skyranger-r70/?vertical=uas&segment=uis
[2] Erico Guizzo, “Japan Earthquake: More Robots to the Rescue” https://spectrum.ieee.org/japan-earthquake-more-robots-to-the-rescue
[3] Erico Guizzo, “Japanese Robot Surveys Damaged Gymnasium Too Dangerous for Rescue Workers” https://spectrum.ieee.org/japan-earthquake-robot-surveys-damaged-gymnasium
[4] Travis Nickles, “Antimicro”, https://github.com/AntiMicro/antimicro