CG1112 Engineering Principle and Practice

Semester 2 2018/2019

"Alex to the Rescue"

Final Report

Team: 02-03-01

Name	Student #	Main Role
Heng Hong Chuan	1	Firmware
Karnati Sai Abhishek	2	Software
Ivan Andika Lie	3	Software
Jess Teo	4	Hardware
Lee Raiyan	5	Hardware

- Use at least 1.15 line spacing.
- Font size = 12 pt
- Maximum number pages = 12 pages (excluding cover, references and appendices)

The main aim of this report is to document your final product. Aim to give a succinct and clear account of your approaches and decisions.
Do not use brackets
Section 3 Needs some words, not just the photo

Section 1 Introduction

The "search and rescue" task that Alex is designed to tackle involves Alex traversing a simulated environment (of area $3m^2$) that resembles a maze. This environment consists of two to four rooms, and each room has been set with various objects that act as obstructions. Alex must then find its way the one clear path that brings Alex from the first room to the last in order to fully map the area.

As a rescue operation, the problem calls for Alex to be remotely controlled from our laptops to move forward (and optionally backward); as well as to turn left and right. Additionally, Alex must be able to identify the colour of regularly-shaped objects that are at least 18 cm tall and that have been placed in the maze. It must then decide if the objects are red or green.

Section 2 Review of State of the Art

GETbot (Autonomous Rescue Robot)

Hardware:
The GERbot is based on a Pioneer 3-AT, which is a four-wheel skid steering drive system. It has sensors mounted to provide odometry information, as well as a laser scanner, monocular camera and a 3D camera to monitor the environment at a fixed level. The GETbot also has a thermal camera to measure the temperature of objects present in the environment, allowing it to detect heat from survivors among the rubble.

Software:
The GETbot uses object-oriented control software, allowing for data reading and writing. It also employs the SLAM framework to map the explored areas and uses RGB-D data to form a 3D representation of the surroundings.

Strengths	Weaknesses
1. The GETbot uses victim detection algorithms that are able to search for victims by detecting signs of human life (e,g, human body temperature, skin color, motion) 2. The GETbot is able to perform autonomous exploration of large areas	1. The GETbot does not always work in real-life scenarios

NuBot (Autonomous Rescue Bot)

Hardware:
The Nubot has an Intel Core i7 to provide processing ability to deal with large amounts of data and robustness when traversing challenging terrains. It has a LiDAR with a field of view of 270°, scanning distance of 30 m and scanning frequency of 40 Hz and also a pan tilt USB video camera for visual perception

Software:
A Robot Operating System (ROS) is used to build the software for the rescue robot. 2D SLAM is also employed to allow for robust scan matching, focusing on the real-time estimation of the robot movement

Strength	Weakness

1. The NuBot is capable of fully autonomous exploration	1. There is inaccuracy of the robot in following the exploration path in real-life scenarios when the disaster sites are filled with unstructured terrains.

Section 3 System Architecture

The high level system architecture of Alex's final system can be described by Figure 1 below. As illustrated, Alex has two major subsystems: the Arduino Uno, the Raspberry Pi. The Arduino is connected to components that require little processing power to function such as the motors, infrared sensors, and the wheel encoders. On the other hand, the Raspberry Pi handles functionalities that are heavy on its processor such as SLAM and the remote access software.

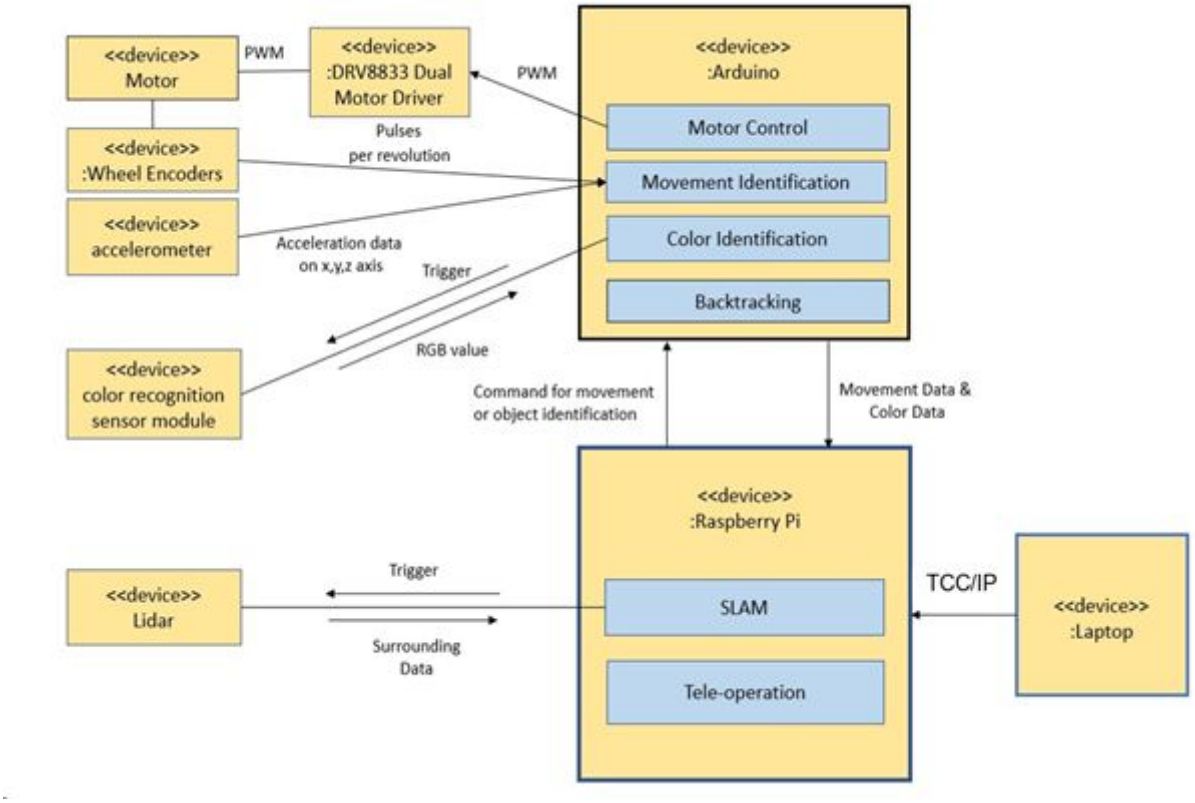

Figure 1. Overview of Alex

Section 4 Hardware Design

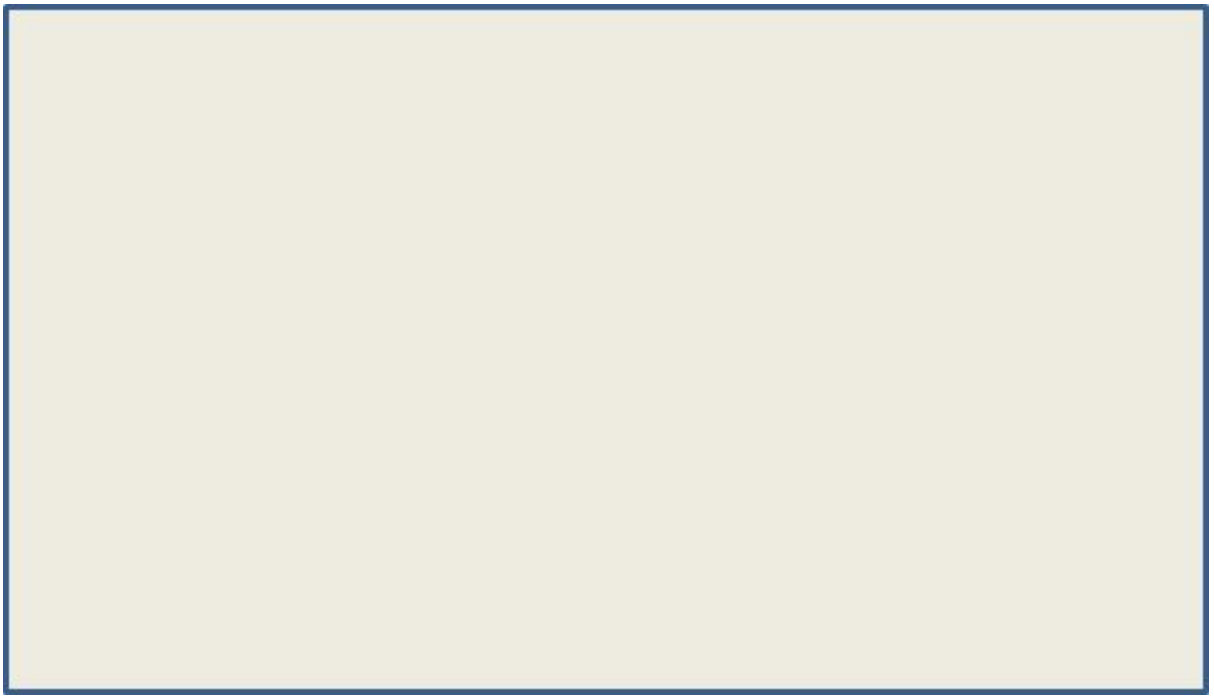

Figure 2. Final design of Alex with hardware components labelled (to be inserted later)

Alex's final configuration is depicted in Figure 2 above. Two motors that turn Alex's wheels are attached to the underside of Alex's chassis, with a wheel encoder attached to each motor.

The middle portion of Alex's chassis houses the Arduino Uno. Due to the multiple connections that need to be made between the Arduino and various components, we decided to keep it in the centre of Alex to allow for the ease of access and orderly wire management. The portable power source for the Raspberry Pi, as well as the battery pack that drives the motors are also placed in the middle section of Alex. This was done to ensure that Alex was well-balanced, and the load that it carries is evenly spaced.

Finally, mounted to the top of Alex is a RPLiDAR as well as the Pi itself. We have designed Alex to be 16cm tall from the, ensuring that the top of the RPLiDAR does not exceed the height of the obstructions placed in the maze (18 cm). The RPLiDAR is connected directly to the Pi, which is placed directly behind it.

In addition to all the typical hardware, we decided to include additional non-standard components that help us achieve Alex's functionalities, described in the tables below:

1. Infrared Sensors

Three infrared sensors are mounted onto Alex. The first two are on the left and right on Alex, and the other two are placed on the front and back of Alex (Figure 3). The left and right IR sensors have been tuned to detect when there is an object less than 3 cm away. The front IR sensor, however, has been tuned to detect objects less than 5 cm in front of Alex. This provides a safe buffer distance for Alex to stop and turn if necessary.

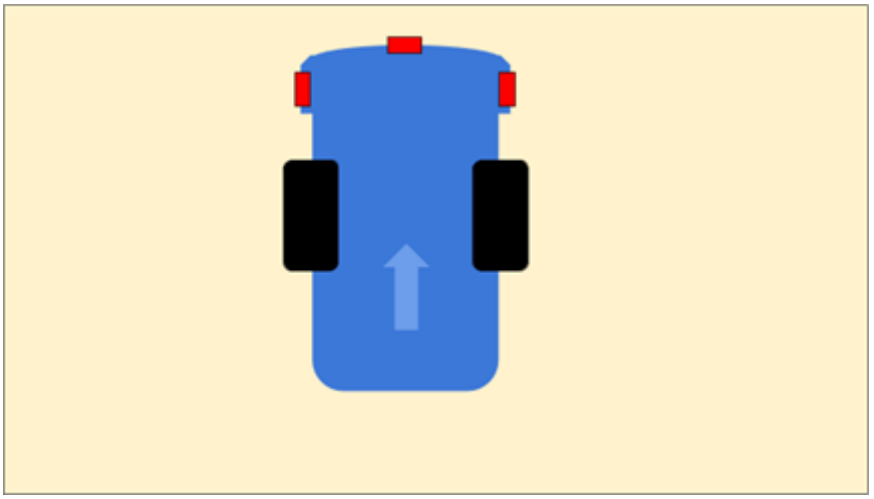

Figure 3. Position of IR sensors on Alex

2. Raspberry Pi Camera Module

The Raspberry Pi Camera Module is placed at the front of Alex, allowing it to capture still images of the area in front of Alex, allowing us to detect green and red objects in the maze (Figure 4)

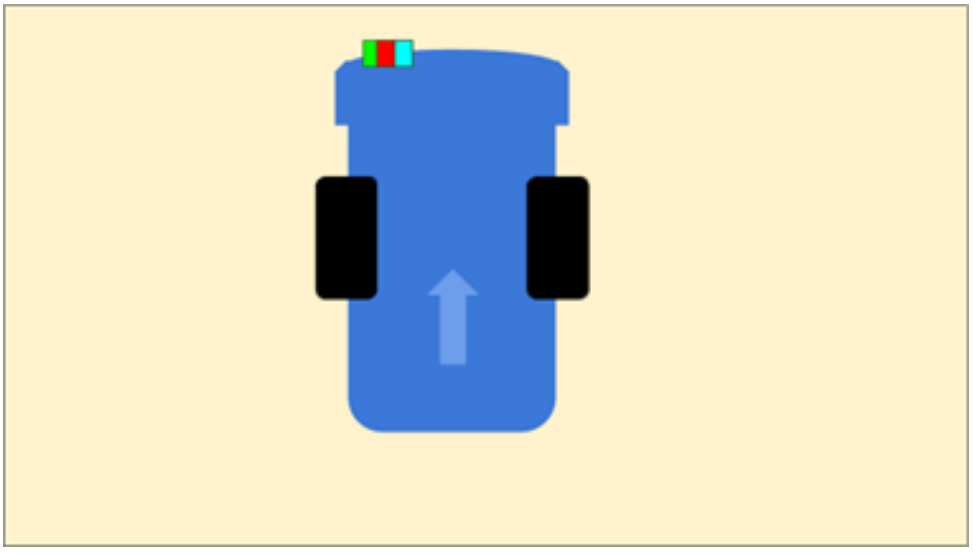

Figure 4. Position of Pi Camera on Alex

Section 5 Firmware Design

<Remove this in actual report>

Please describe:

1. High level algorithm on the Arduino Uno
2. Communication protocol (format of messages and responses).
3. [Optional] Additional noteworthy software-related stuff.

</Remove this in actual report>

Arduino
When a command is received, the Arduino sends a message back to the Raspberry Pi if the full command packet has been successfully received. If it has not been successfully received, the Arduino sends a packet back to the Pi with a message of either "RESP_BAD_PACKET" or "RESP_BAD_CHECKSUM". However, if the packed is good, the Arduino then checks to ensure that there are no obstructions on the left, right, and in front of Alex. If there is an obstruction detected on the left, Alex will take action to shift its movement back to the right. On the other hand, if there is an obstruction detected on the right, it Alex moves to the left. The Arduino receives and sends information to the Pi using USART communication. The commands consist of character output from keys and a float variable. The char controls the direction that we are going (e.g. <w>(forward), <a>(turn right),<s>(back),<d>(turn left) etc) whereas the float variable is used to indicate the distance or rotations for each command is carried out (e.g. go forward for 2 seconds). Conversely, USART communications also let the Arduino communicate data involving Alex's position in the maze, the area mapped by LiDAR and various diagnostic tests back to the Raspberry Pi.

Section 6 Software Design

<Remove this in actual report>

Please describe:

1. High level algorithm on the Pi to handle:
a. Teleoperation
b. Color detection
2. [Optional] Additional noteworthy software-related stuff.

</Remove this in actual report>

Raspberry Pi

The Raspberry Pi allows us to perform teleoperation of Alex, done mainly by using SSH from a laptop and Raspberry Pi. This, in turn, maintains a USART Operation with Arduino which allows us to move Alex.

Firstly, SSH must be enabled in Raspberry Pi. Using Wifi and SSH client software such as PuTTY, we connect to the Raspberry Pi. Afterwards, we set up a virtual network computing(VNC) by allowing Raspberry Pi to interface through VNC. The high level algorithm that the Pi follows is as follows:

Initialisation of Pi

Once the Raspberry Pi has been booted up, the Pi performs handshake with the Arduino in order to initiate the process. Any inactive modules would then be deactivated in order to make power consumption more efficient. Next, the Pi will then run the diagnostics algorithm to check if it is functioning properly. These tests include a LiDAR health check, sensor checks and a motor check. The Pi then goes on to ask for an initial reading of the environment. In response to this, the LiDAR uses the SLAM algorithm to scan the surroundings.

Receiving User Commands

Using a laptop connected to the Pi via VNC, the user would decide direction of navigation base on the map display and input a command into the Pi, instructing Alex to carry out one of the following actions:

Input	Command
W	Move forward
S	Move backward

D	Turn right
A	Turn left
5	Stop
6	Object Detection

The Pi then sends the respective commands to Arduino to be carried out.

Carrying Out Commands
Once the Arduino code is uploaded from the Pi to the Arduino Uno, the Arduino carries out the tasks based on the instructions in the code.

Mapping The Environment
While navigating, the LIDAR constantly maps the surroundings using SLAM algorithm. This map is updated and saved using a RBPF SLAM algorithm. Using a Stack ADT, the movement of Alex will be stored in a combination of direction and distance travelled in that direction. (e.g left, 30 cm). An interrupt will be triggered once the room has been mapped, notifying the user to carry out the backtracking steps.

Additionally, SLAM also allows Alex to observe regularly shaped objects (e.g. circles, squares, rectangles) which will be identified as “Miss Scarlet”. This would allow the operators to launch the algorithm for colour detection and correctly identify the color of “Miss Scarlet”.

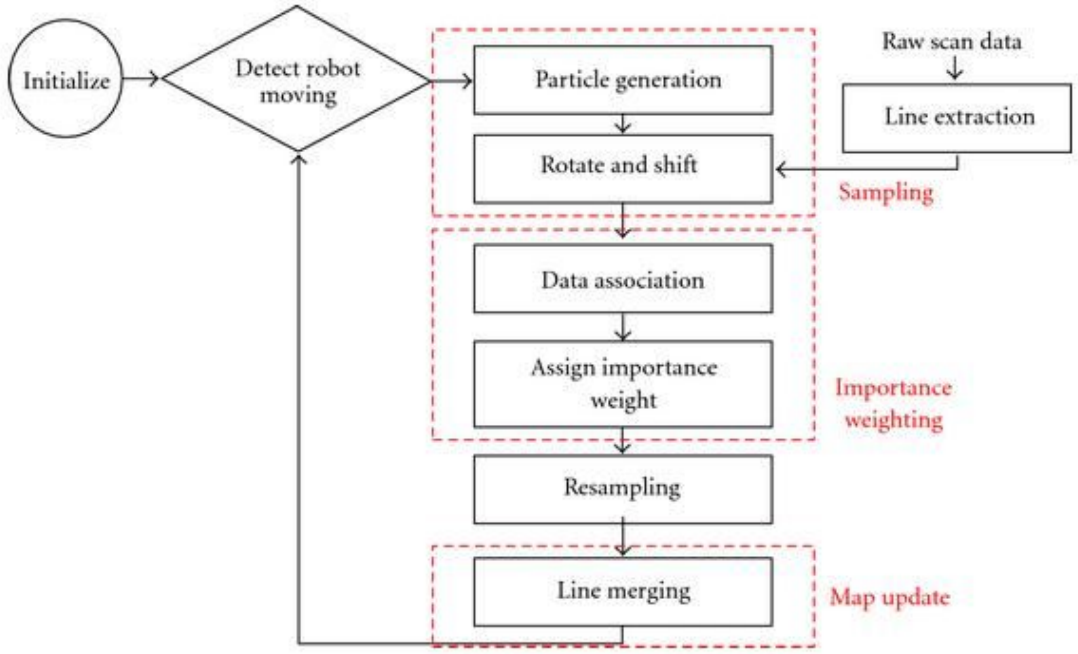

Figure 1: Overall flowchart of a particle filter-based algorithm.

Color Sensing

~~~~?
~~~~

Section 7 Lessons Learnt - Conclusion

<Remove this in actual report>
Give two most important lessons learned in this project and the 2 greatest mistakes you made as a group.
</Remove this in actual report>

1) Do a sanity check regularly even though it takes more time.
2) It is important to plan the wiring of the hardware.

By performing a sanity check regularly, it allows us to debug our code easily as the mistakes will be more localised, it allows us to spot our mistakes quickly. When we were working on our project, we did not test it regularly while we were working on it. This resulted in a tough time trying to rectify our errors. Such problems can be avoided if we constantly check for

References

1. Fischer, Dirk, et al. "RoboCup Rescue 2016 team description paper team GETbots (Germany)." *RoboCup Rescue Robot League Symposium*. 2016.
2. Jacob, Gregoire, et al. "JACKSTRAWS: Picking Command and Control Connections from Bot Traffic." *USENIX Security Symposium*. Vol. 2011. 2011.
3. Liu, Yi, et al. "The design of a fully autonomous robot system for urban search and rescue." *2016 IEEE International Conference on Information and Automation (ICIA)*. IEEE, 2016.
4. Bor-Woei, Kuo, et al. “A Light-and-Fast SLAM Algorithm for Robots in Indoor Environments Using Line Segment Map” *Journal of Robotics* Vol. 2011, Article ID 257852, 12 pages.