**MREN 203**

Technical Update

***Feb. 4th, 2024***

***Prepared for:***

*Kingston Fire and Rescue*

***Prepared By:***

David Cheng:

Coleman Farvolden:

Josh Westlake: 20349406

*“We do hereby verify that this written report is our own individual work and contains our own*   
*original ideas, concepts, and designs. No portion of this report has been copied in whole or in*   
*part from another source, with the possible exception of properly referenced material.”*

***02/04/2024***

Table of Contents

[I. Introduction 3](#_Toc157968100)

[II. Operational Scenario 3](#_Toc157968101)

[III. General Update 4](#_Toc157968102)

[IV. Electromechanical update 4](#_Toc157968103)

[V. Software update 6](#_Toc157968104)

[VI. Autonomy update 7](#_Toc157968105)

[References 9](#_Toc157968106)

[9](#_Toc157968107)

[Appendix A 10](#_Toc157968108)

[Appendix B: 11](#_Toc157968109)

## I. Introduction

This document aims to provide a comprehensive update on the LifeBot project, detailing recent advancements, challenges, and future directions of the project. This document serves as a critical communication link between the LifeBot development team and stakeholders, ensuring alignment with project goals, progress, and expectations. This document will discuss the steps taken thus far to bring the LifeBot to reality. The update will include a report on the state of software, control, autonomy, electrical and mechanical aspects of the LifeBot.

## II. Operational Scenario

This robot is used when there is a cardiac arrest emergency. The LifeBot will solve the problem of finding an AED. When a cardiac arrest occurs, there are many ways in which LifeBot can be automated to come to the scene. Buildings could have switches on the wall to call emergency services as well as the LifeBot to the scene. Once given the location of the scene, the LifeBot would navigate quickly to the victim and provide an AED for atrial defibrillation. The goal is to significantly reduce the time it takes for a cardiac arrest victim to receive an AED.

For the LifeBot to be successful in an operational scenario, key requirements must be satisfied. Namely, the LifeBot must be an extremely efficient and reliable robot as lives could be at stake. Additionally, the LifeBot must be able to perform seamless indoor navigation and the LifeBot must be integrated with emergency response systems to ensure quick delivery of AED.

**Overview of the technical capabilities:**

The LifeBot must be able to navigate reliably, quickly, and autonomously. Software such as ROS, AI tools and Arduino programming will process the I/O to create a sophisticated navigation and mapping system. The LifeBot will use a variety of intelligent sensors inputs to navigate; Lidar, distance sensor and rear encoders. Processing of the inputs on the LifeBot will produce outputs to the motors necessary to navigate to the scene.

**Overview of the emergency calling**

Communication with the LifeBot will be over WIFI and Bluetooth. There are two main ways the LifeBot will be sent a location. Firstly, the LifeBot will be integrated with the 911 system and when a call is made the LifeBot will immediately navigate to the coordinates of the call. The second use case is most effective when the LifeBot system is installed in a building. The main idea is that there will be buttons similar to fire alarms installed throughout the building that when pressed triggers the LifeBot to optimally navigate to the site of the button press.

**Goal for Prototype**

Due to time constraints and financial constraints, only a prototype of the LifeBot will be created with fairly basic functionality. The goal for the prototype is an autonomously navigating rover robot that upon button press navigates autonomously to a desired door. The robot will sound an alarm, have emergency lights, and have an AED attached to it. The prototype will be a proof of concept and precede the full development of the LifeBot.

## III. General Update

Overall, the team has been meeting deadlines and keeping up with project deadlines. Electrical hardware integration has been completed, Modeling and controls have been completed, sensors have been integrated into the software and autonomous navigation techniques have been investigated. The updated project schedule of the LifeBot AED Delivery robot is further detailed in a Gantt Chart found in Figure 7 in Appendix A. This Gantt chart highlights all key milestones, assigned responsibilities to various members of the team, and the status of distributed tasks.

## IV. Electromechanical update

Most key components on the prototype have been connected to the LifeBot, which includes the DC motors via motor shield, encoders, and light sensors.

The motor controller is connected to the motors and to the Arduino and the motors run as intended. By providing a high signal and a low signal to the signal pins, a voltage difference allows the motor shield to provide either a positive or negative voltage to the DC motor, while the E pins determine the voltage strength, effectively allowing the control of power draw on the DC motor, thus controlling the speed of the motor. The pin layout is shown below in Figure 1.

A screenshot of a computer

Description automatically generated

Figure 1: pin assignments for motor shield

Along with the motor shield pins, there are 17 more pins connected, 4 for the encoders attached to the DC motors on both sides of the robot. The encoder data shown in Figure 3 provides a speed estimation for each rear wheel and more precise control of robot turning and speed, allowing for free movement. The 4 pins include the Vcc and ground pins, power the encoder, while two digital pins allow the writing of the encoder. The encoder is connected to the Arduino by these 4 pins and sends back data based on rotations. The pin connections to the Arduino shield are shown below in Figure 2.

A close up of a circuit board

Description automatically generated

Figure 2: Arduino uno wifi rev board with its shield on top and some of the pins layout

A diagram of a sequence

Description automatically generated

Figure 3: Encoder data to Arduino [1]

The light sensors send back data when detecting an object but has been found to return somewhat unreliable data on the distance between the sensor and the object. Each light sensor is assigned 3 different pins, two of which power the sensor, and the last one outputs an analog signal for the Arduino to read.

On board the LifeBot on the Arduino uno Wi-Fi board, there is also a gyro sensor that outputs acceleration shown in Figure 4 whenever the robot moves from its original position. This is ideal for a robot to estimate its location relative to the origin, as a guideline for its navigation system.

A number of numbers on a white background

Description automatically generated

Figure 4: example outputs of the gyro sensor: Acceleration in [X, Y, Z] in units of gravity

## V. Software update

The team has made substantial advancements in developing various PI (Proportional Integral) control system for the LifeBot.

The heart of the control system lies within the PI controller that operates on the bases of well-defined motor and encoder pin configurations, coupled with a variety of interrupt service routines to ensure precise encoder tick counting. This controller dynamically refines the motor PMW (Pulse width modulation) commands, allowing it to respond ‘proportionally’ to variances between the desired wheel speed, and calculated wheel speeds. The main loop orchestrates the calculation of wheel angular rates from the encoder data and calculates the appropriate PMW motor signals to align the vehicle's speed with the predefined target speed.

To verify the controller operates as expected the team has implemented real-time monitoring through means of the serial monitor and serial plotter imbedded in the Arduino IDE. These readings provide insight into the wheel velocities as time goes on to verify the speed of the wheel is operating as Intended. Figure 5 showcases the results of the speed controller, showcasing a controlled decrease in speed from 1.1m/s to 0.6m/s. As the team continues to fine-tune and enhance the control system the expectation is to see a heightened level of stability and responsiveness in LifeBot’s speed regulation.

A graph with lines and numbers

Description automatically generated

Figure 5: Desired Speed PI Controller Simulation - Blue line is the target Speed, Orange and Green lines represent the controlled regulation of the right and left wheels of the LifeBot respectively.

Further measures have been taken to ensure that the wheels operate at the same speed to provide the LifeBot with an accurate trajectory. Due to deficiencies with the provided motors each of the motors operate at slightly different rates causing discrepancies between the speed of the wheels. The implementation of an additional PI system tasked with minimizing the error between the rate of the left and right wheel speeds inspires further confidence in the navigational control of the robotic system. Figure 6 showcases right and left wheel speeds as the orange and blue lines respectively, showcasing the accuracy with which the team has been able to develop the mobility of the robot.

A graph with lines and numbers

Description automatically generated with medium confidence

Figure 6: Wheel Speed Monitoring – The Blue and Orange lines showcase the speed of the right and left wheels speeds respectively; Green Line is the error found between the 2 values

Similarly, a PI controller has been implemented to provide a feedback loop-based control system capable of turning the robot at a previously determined rate. This system operates similarly to the speed control system allowing the robot to constantly make proportional changes to the motor inputs to change the trajectory of the robot when turning.

## VI. Autonomy update

For the final prototype, the goal is to be able to put the LifeBot in a room with all of the tools it requires to autonomously navigate to the nearest place of need. This relies on a high degree of autonomy and autonomous navigation. The expected roadmap and steps required to achieve autonomy are shown below in Figure 7.

A diagram of a software development process

Description automatically generated

Figure 7: Roadmap to achieve autonomous navigation

**Software:**

To accomplish such a level of autonomy, LifeBot will be using ROS software. Ros (Robot Operating System) is an open-source software kit for robotic applications . It is essential to use ROS as it would be unreasonable to develop robotics software from scratch. ROS software can connect to sensors and motors via Arduino to control the Robot as shown below in Figure 8.

Diagram of a computer

Description automatically generated

Figure 8: Communication between Arduino and PC over ROS

**Navigation:**

The LifeBot will use the ROS navigation stack which will enable LifeBot to build maps and navigate a room autonomously. The LifeBot will use Gazebo and advances robot simulator to simulate an environment for autonomous navigation and ensure effectiveness of the robot .

**Computers:**

The software and automation in the LifeBot require sufficient computer power to operate. The LifeBot has a raspberry Pi onboard. A raspberry pi is about 10 times slower than a desktop PC. The Raspberry Pi is too slow to navigate effectively autonomously. In order to have sufficient processing power for autonomy, ROS can be run on a powerful PC connected over WIFI to the Raspberry Pi on the LifeBot.

The general steps to achieve full automation are outlined blow in the Figure 2: Roadmap to ROS. Currently phase 1 of the roadmap has been completed and ROS is working on both the Raspberry Pi and desktop computer and communication between the two is seamless which is shown in Figure 8 in Appendix B. The next phases of the roadmap will be completed as indicated on the project timeline in Figure 7 in Appendix A.

# References

|  |  |
| --- | --- |
| [1] | t. youngblood. [Online]. Available: https://www.allaboutcircuits.com/projects/how-to-use-a-rotary-encoder-in-a-mcu-based-project/. |
| [2] | ROS. [Online]. Available: https://www.ros.org/blog/why-ros/. |
| [3] | Aditat. [Online]. Available: https://atadiat.com/en/e-rosserial-arduino-introduction/. |
| [4] | ROS. [Online]. Available: https://wiki.ros.org/navigation. |
| [5] | gazebo. [Online]. Available: https://gazebosim.org/home. |
| [6] | r. patridge. [Online]. Available: https://www.techaddressed.com/reviews/mini-pc-better-value-raspberry-pi/. |

# 

## Appendix A

A screenshot of a computer

Description automatically generated

Figure 9: Updated Gantt Chart Detailing the Scheduled TimeLine for the Project

## Appendix B:

A screenshot of a computer

Description automatically generated

Figure 10: ROS communicating between PC and Raspberry Pi over SSH