CARLETON UNIVERSITY
MAIL DELIVERY ROBOT

By
Stephen Wicklund, Emily Clarke
Supervised by: Dr. Babak Esfandiari
October 2021

A Fourth Year Project Report
submitted to the Dept. of Systems & Computer Engineering
in partial fulfillment of the requirements
for the degree of
Bachelors of Engineering

Contents

List of Figures

List of Tables

List of Abbreviations

VoIP	Voice over Internet protocol
MRI	Magnetic resonance imaging

Chapter 1

Introduction

1.1 Objectives

1.1.1 System Power & Charging

The system will be able to run wirelessly through batteries onboard the robot. These batteries will be able to be recharged autonomously when the robot returns to the station. Minimizing down-time and allowing the robot to charge when at any station autonomously will allow the robot to maximize the time it is able to deliver mail.

1.1.2 User Interface

A user should be able to request delivery robots, make requests for the robot to deliver mail, and monitor delivery requests through a web-based application.

1.1.3 Beacon Navigation

The robot will be able to determine a general location based on the signals from nearby active beacons. The robot does not need to be able to determine a specific geographic location, but should be able to make correct navigation decisions based on this information. The beacons are actively powered, therefore continuous operation of

the system requires replacement of the batteries in the beacons. The required maintenance of the system can be minimized by detecting dysfunctional or low on battery beacons early and autonomously through the robots. Furthermore, the amount of beacons used will be kept to a minimum to reduce the initial cost of installation and upkeep.

1.1.4 Tunnel Navigation

The robot will be able to successfully navigate from point A in the tunnels to point B. This involves creating a path from point A to point B based on a map and making the correct turns at tunnel junctions.

1.2 Relationship to Degree Program

A discussion of how the project relates to the degree program of each student. A student in undergraduate program X should be able to demonstrably state that their planned role in the project is primarily in relation to X.

1.3 Required Skills

A discussion of how the group, collectively, has the skills required to undertake the project.

Chapter 2

Background

A brief background of the project. Background should identify what has been done to address the problem, what the state of the art is, etc.

Chapter 3

Methods

A brief background of the project. Background should identify what has been done to address the problem, what the state of the art is, etc.

3.1 Project Management

3.1.1 Timeline

A proposed timetable for completion of the project including major intermediate milestones.

3.1.2 Project Risks

A discussion of possible project risks and mitigation strategies.

3.1.3 Required Components

A list of special components and facilities that you require.

3.2 Project Planning and Organization

The previous team struggled with closely collaborating on all aspects of the project and integrating individual work together. The result of this is that their completed project is disjointed and it is unclear how to run it all at once. In order to avoid these issues, the following project management specific principles should be met:

- Every team member generally understands all aspects of the project
- To-do tasks are clearly laid out and who is currently working on what is visible
- A historical view of progress is available for review
- A strong emphasis on agile development principles
- No completely independent changes

3.2.1 Issue Tracking

To ensure all necessary tasks are tracked, every todo item will be created as a GitHub issue. Issues will be assigned to project member(s) so that it is clear who is working on what at which time. In addition, an individual member can quickly filter only their issues. As well, only issues without an assignee can be filtered in order to see if there are any issues that are being ignored or forgotten about. This should decrease the likelihood that important issues are missed and make it much easier to view who is working on what.

3.2.2 Issue Filtering

Issues also have the ability to be labelled for quick filtering. One custom label that will be implemented is a “have discussion” label. This will be used during team meetings to quickly pull up a list of what team members want to discuss. This will ensure that issues that need a team discussion won’t be forgotten about and potentially pushed into the future.

3.2.3 Weekly Project Board

GitHub issue tracking will be integrated with a weekly GitHub project board. This will show which issues are being worked on for the week in an agile development Kanban view. This is an ideal view for quickly getting an idea of project progress, making it easier and faster to make decisions for what to work on or while reviewing the past week's progress. In addition, we will automate the GitHub project board to match actions in the repository. For example, automatically moving an issue to the "done" column and closing it when the linked pull request is approved and merged. This will reduce project management overhead and free up time to focus on coding. Past boards will remain view-able so project management issues can be identified and solutions implemented on a recurring, agile basis.

3.2.4 Code Review and Collaboration

The GitHub repo will be set up to prevent a merge to main without approval from a different team member. This will force code review and increase general understanding of the project for all team members. Importantly, this will prevent only a single team member from having knowledge of a piece of code.

3.2.5 Miscellaneous

The project management goal will be for any issue to be worked on by any member interchangeability. It is also vital that our project management strategy is reviewed on a recurring basis to ensure that it is working. As a result, the above guidelines are a starting implementation and will likely change in an agile manner as improvements are discovered.

3.3 Build Automation System

Currently, there are five repositories with code for the project.

1. WebServices-Client
2. WebServices-Server
3. Roomba
4. Distance-from-rssi
5. Web-App

Each requires different dependencies and methods for building. This makes it difficult to develop and build the software for all components.

3.3.1 Repository Structure

There is not a need to create multiple repositories for different components of the project since no component is completely independent of one another. Furthermore, by organizing all components into a single repository, we have a "single source of truth". This will make it easier to more closely collaborate, share and verify code, and refactor when needed. The organization of the project will be a single mono-repo with each component being a top-level directory.

3.3.2 Github Actions (Continuous Integration Pipeline)

In order to simplify the building process for all software components of the system a GitHub actions scheme will be set-up to automate the build process. On a push the GitHub action will automatically build all the components using docker. Furthermore, this system can run unit tests and integration tests, verifying the code on every push. With this in place, all the components of the system can be kept up-to-date and tested continuously.

3.3.3 Docker

Docker will be used to package the software components with the libraries and dependencies required to run them in any environment. It will allow easy sharing of source code without having to set-up or change our development environments. With a DockerFile setup the code of any component can be built through the GitHub action scheme or locally

3.4 Web Application

The web application is the user interface for the entire mail delivery system. It should primarily allow users to:

- Request a delivery robot
- Make delivery requests
- Monitor delivery requests

The web application should:

- look polished
- be intuitive to users
- be secure from malicious activity
- be available to users through any web browser
- function properly on a mobile device

3.4.1 Angular

The web application will be built using Angular, a typescript-based web application framework.

PWA Angular allows for the creation of progressive web apps (PWA) which are applications delivered through the web but intended to deliver app-like experiences across any platform.

Code Generation Angular has many tools for code generation, allowing for rapid creation of polished looking interfaces.

Templates Angular has a powerful template syntax which allows for creation of complex UIs without having to reinvent the wheel at every step.

3.5 Robot Operating System

References

[1] T. Me and R. You, "A great result," *Wonderful Journal*, vol. 5, no. 9, pp. 1–11, 1998.

[2] J. Him and K. Her, "An even better result that you won't believe," *Best Journal Ever*, vol. 4, no. 8, pp. 55–66, 2002.