Lev Tours

Dr. Michael Leverington - Sponsor
David Failing - Mentor

Members
Erik Clark
Kyle Savery
Alexis Smith
David Robb
Ariana Clark-Futrell

Software Design Document Version 1.1

2/12/2021

Overview

This document lays out the design of each module in our project “Thirty Gallon Robot Part III: The Smiling Tour Guide”.

Table Of Contents

1. Introduction

In the field of computer science, mobile robotics are capable of increasingly complex tasks, and the necessary hardware for these tasks has become far more accessible in recent years. As a result, the costs associated with these materials have also decreased, allowing for a greater number of individuals and organizations to take part in the research and development of mobile robotic platforms. Most significantly, educational organizations can give students the opportunity to interact with robotics either from a mechanical or programmable standpoint. To take full advantage of mobile robotics, one of the main goals of our project, “Thirty Gallon Robot Part III, The Smiling Tour Guide”, designed by our sponsor Dr. Michael Leverington, is to demonstrate that an institution's budget for robotics can be further reduced, and used more effectively, by utilizing inexpensive materials. For Dr. Leverington’s robot, one of the most inexpensive components is the thirty gallon barrel that all of the other components are built around. When this project is complete, the robot will be an example of how instructors can get students involved in the world of robotics, which could lead to an expansion of the field.

Beyond computer science the industry of robotics is involved with numerous fields such as mechanical and electrical engineering, medicine, agriculture, and manufacturing. In 2019, the global robotics industry was valued at $62.45 billion [1]. A subset of this industry that is of particular relevance to our project is known as mobile robotics. This field is responsible for creating robots that can move in 3 dimensional space without the need for human assistance. The market size for mobile robotics was approximately $9.34 billion in 2018 [2].

The Thirty Gallon Robot project is in its third year of development. The first team created the foundation for Robot Assisted Tours (R.A.T) by programming the robot to respond to commands from an Xbox controller. The second team built upon R.A.T by creating a way for the robot to generate its own maps by navigating around the building. A stretch goal for the second team was to develop a framework for Wi-Fi localization capable of directing the robot in a building, but due to the COVID-19 pandemic they were unable to test their design within the engineering building and so that development was not completed at the time. Therefore, this task now falls upon our team. Wi-Fi localization will allow for the robot to navigate on its own throughout a building without the need for an Xbox controller. Requiring user control defeats the robot’s purpose of being an automated tour guide.

To complete this project’s third stage of development, our team consists of 5 members, as well as a mentor and a sponsor:

- Dr. Michael Leverington, Team Sponsor
- David Failing, Team Mentor

- Erik Clark, Team Lead
- Kyle Savery, Team Architect
- Ariana Clark-Futrell, Team Communications/Recorder
- David Robb, Team Release Manager
- Alexis Smith, Team Coder

2. Implementation Overview

In order to fulfil Dr. Leverington's project vision, the proposed software must tell the robot where it is and where it needs to go in a building. Specifically, our solution must incorporate the following:

- A system that can be run and tested independently of the robot on a laptop.
- A system that takes in Wi-Fi signals from available routers.
- A system that can navigate between any two accessible points in the building.
- A GUI to display the status of the map and accept user requests.

The software will be built on a laptop at first to demonstrate its functionality and modularity. The core components of our design will use routers and their respective signal qualities to calculate the position of the device. Using these stable data points we will create a navigational system that works within any building. Finally, a Graphical User Interface (GUI) must be built that displays localized information, such as the building map, the device's current location on that map, and possible destination points.

Our implementation is split up into a backend portion that deals with wireless network scanning and building navigation, as well as a frontend portion which contains the GUI. Both sections of our system use Python. The library we use for the backend is called RSSI (Relative Received Signal Strength) which allows us to scan for access points. For the frontend we are using *tkinter* to build our GUI.

3. Architectural Overview

The front and backend components work together to create a smooth user experience as the robot guides them through an unfamiliar building.

3.1 Frontend Overview

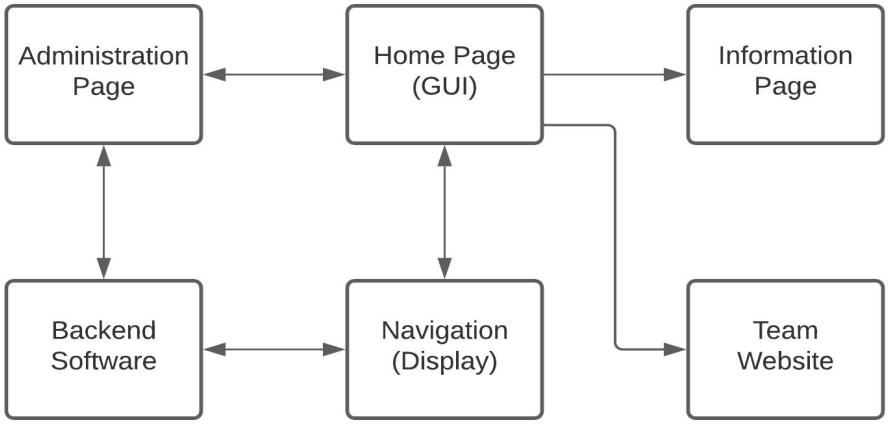

Figure 3.1: Overview of Frontend

The frontend's interface utilizes a home page which branches off into four other pages. This GUI is responsible for interacting with backend and frontend users. Backend users are given special privileges and are capable of interacting directly with stored Wi-Fi information about any building. Frontend users are members of the public who are only allowed to use the software as a guide through a building. Referring to the high-level overview of the frontend system (Figure 3.1), these other pages are as follows:

- Administration
    - The administration page is used by backend users to not only set up the software in a new building, but also allows the existing information about the current building to be edited. This editing includes adding/removing a location in the building's set of registered points.
- Navigation
    - The navigational page is where the software will typically be interacting with a frontend user. This page will display a map of the building, the device's current location, and allow the user to make destination requests to any of the available locations. The user can also select a "tour" mode which will take them to important locations throughout the building. Both the navigation page and administration page consult the backend software in order to update the user's display.
- Information
    - The information page will supply the user with basic information such as frequently asked questions and how to operate the software.
- Team Website
    - This page simply links to our team's website, which will show the user all of our detailed documentation about this project.

3.2 Backend Overview

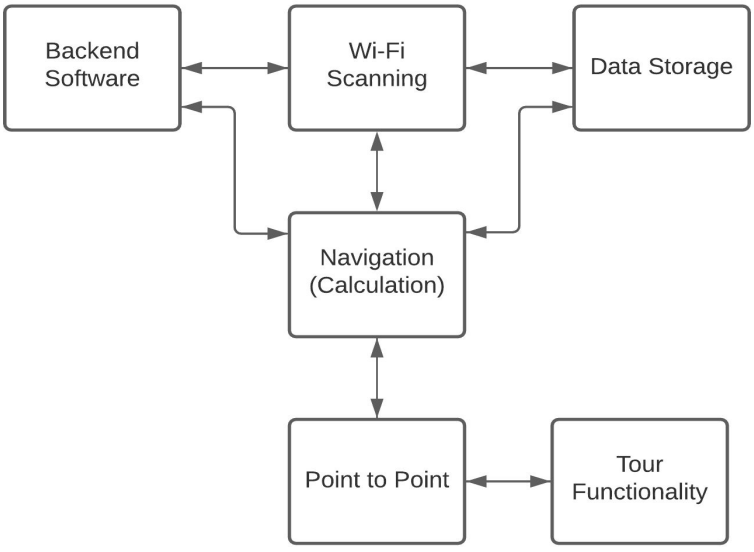

Figure 3.2: Overview of Backend

This system's backend software can be broken down into three main parts:

- Wi-Fi Scanning
    - The Wi-Fi scanning module is used to monitor nearby wireless access points for changes in their signal strengths. Wi-Fi is either scanned during navigation or setup. During setup, the device is kept in the same area while the nearby signals are scanned repeatedly to develop a data set containing the average signal values for each access point that was detected. During navigation, scans are repeated while the device is moving through the building until the destination is reached. After each scan, these current signal values are compared against data sets created in the setup phase to determine the device's location.
- Data Storage
    - The data sets returned by the Wi-Fi scanning module are saved in a text file so they can be accessed on startup of the software. Whenever a backend user updates these data sets, the corresponding building's data file is rewritten.
- Navigation
    - The navigation module uses the data loaded on startup from the stored text file to compare to current Wi-Fi scans. The process with which this data is compared is where the majority of time devoted to developing the backend software was spent. Since the signals retrieved from nearby access points varies even when stationary, this method of comparison relies on finding which stored location is most similar to the current scan. This module also directly relates to the "Point to Point" and "Tour Functionality" aspects of the systems diagram in Figure 3.2. Point to point navigation is the base function of our software and uses the device's current location and a map of the building to calculate the direction the user must move while following the shortest path to their destination. Tour functionality merely takes this navigation between two points and guides the user to numerous points throughout the building.

4. Module and Interface Descriptions

By separating our system into a front and backend module the user is able to operate the software without needing extensive knowledge of the backend module. These modules are explored in greater detail in the following sections.

4.1 Frontend Software

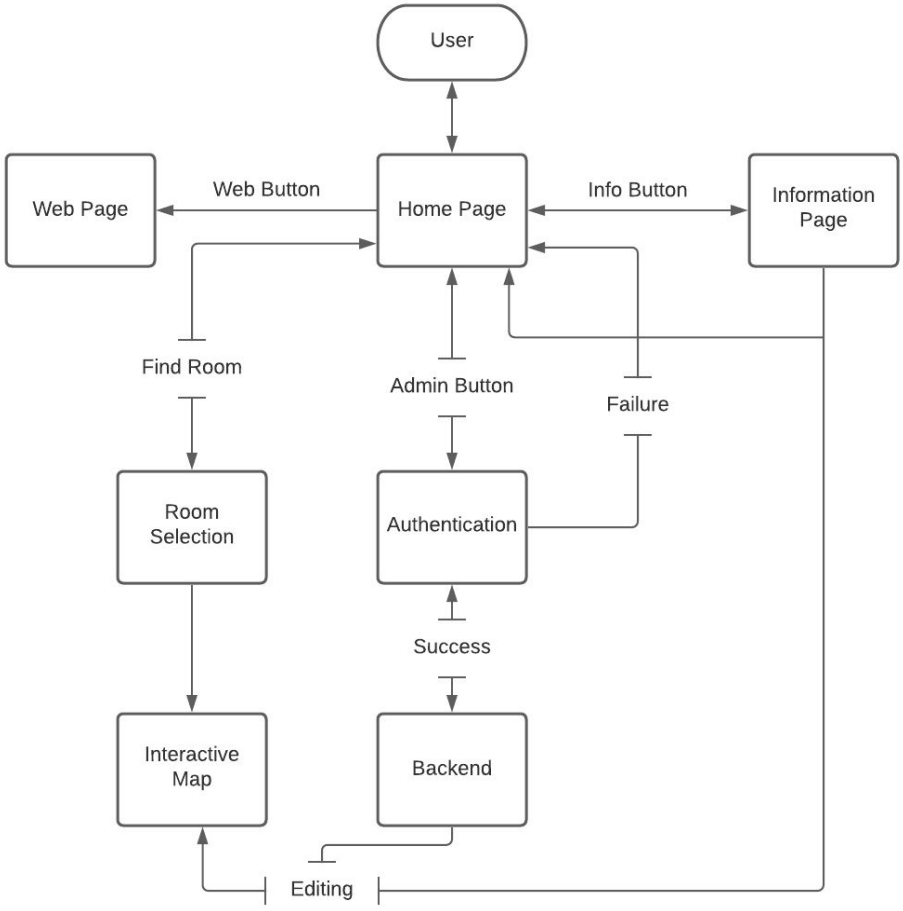

Figure 4.1: Detailed Diagram of Frontend Operation

4.1.1 Graphical User Interface

The software used to build our GUI is the *tkinter* library. The GUI's home page is the hub of the frontend operation and links the administration page, information page, team website, and navigation display together. Figures 4.2 and 4.3 break down the widgets and functions from *tkinter* used to construct our GUI.

Widgets

Option Menu	A dropdown menu that the user can select from
Entry	Used to take user input (for login feature)
Button	buttons can be linked with functions via the command feature
Canvas	A container for images and other widgets allowing for grouping and simpler resizing.

Figure 4.2: GUI Widgets

Functions

PhotoImage	Using an image file link, this file will return an image object
Resize	Used to resize objects in the GUI
Pack	Places objects within the GUI anchoring them with commands like bottom or nw(north west)
Place	Places objects using the *xy*-coordinate plane

Figure 4.3: GUI Functions

4.1.2 Home Page

The home page allows for easy navigation throughout our system by connecting all parts of it together using Buttons. So far, there are 4 buttons on the homepage where a backend user with the appropriate credentials has access to all of them however a public user will not be able to access the administration page and features. Every page except for the website will also have a button leading back to the home page.

4.1.3 Administration Page

The administration page will have a login feature that will allow editing of building information and protect the system from unauthorized changes. During development, the password will be hard coded into the system as a way to show proof of concept and ease of use during construction. Once complete, after a successful login the system will allow the user to edit or create maps and nodes for the navigation system.

4.1.4 Information Page

The information page will display general information about our system. It will include a section giving thanks and credits for any software we used in building our system and a help section containing FAQs. These systems will be displayed with canvas widgets.

4.1.5 Team Website

The team website button will open an exterior web browser and initially take the user to a landing page. This landing page reminds the user to exit their browser once finished looking over the website in order to return to the program.

4.1.6 Navigation Display

The navigation display will be set up with a map and node system that communicates with the backend. The nodes will act as key points on the map and will be linked to specific Wi-Fi data ranges gathered using our Wi-Fi scanning module (section 4.2.1). The number of nodes will be dependent on each map and will be easily edited using the administration feature of the GUI. Every map will have a distinguishable node (different in shape and color) to represent the current position of the robot. As the robot moves and the system takes scans this node will also move around the map. Each mapping page will have a search bar that will auto fill from a list of rooms as you type, a drop down menu, and a goTo button. Once the user selects a destination and clicks the goTo button the shortest path to their destination will be given. These directions will also be sent to the robot. The directions given to the robot will be pulled from the backend point to point system (section 4.2.4) allowing the robot to confirm its current location and receive simple directions i.e (forward, backward, turn left, turn right) as it drives to its destination. Finally, the navigation display will have a tour option which will give the user a tour of the current building using the touring functionality (section 4.2.5).

4.2 Backend Software

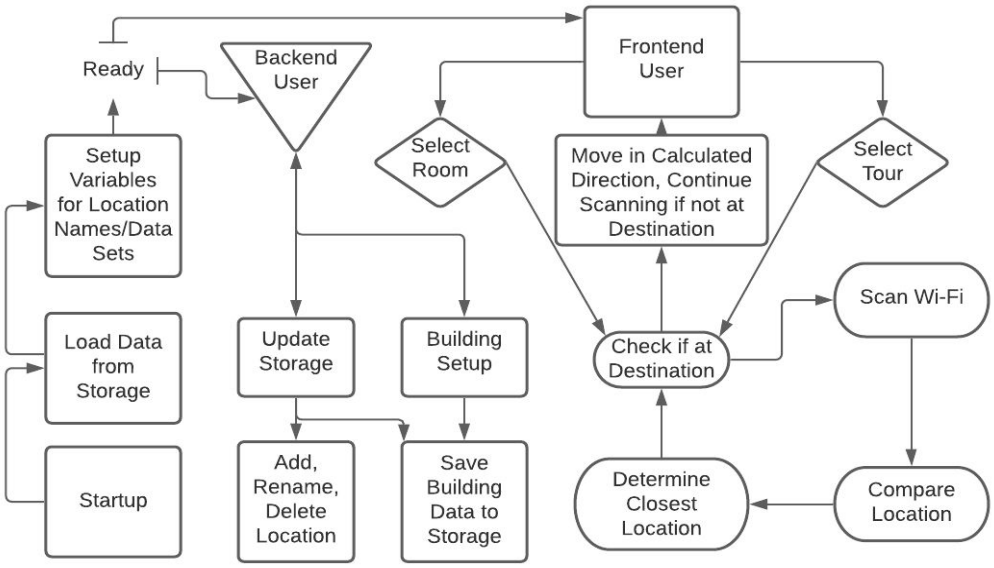

Figure 4.4: Detailed Diagram of Backend Operation

4.2.1 Wi-Fi Scanning

This module utilizes a wireless tool for Linux, *iwlist*, to scan and output valuable information about each nearby wireless access point. To streamline this process, our system imports a library called RSSI which will execute an *iwlist* call and parse the output to return the relevant pieces of information. Specifically, this process will output current signal strength and a unique identifier for each access point. This data will be used for two purposes, setting up the device in a new building and determining its current location. For setup, numerous scans are made in the same area to create a new set of data which reflects the possible ranges of signals for each available access point. The data set for each registered location is stored for future use. When testing nearby networks to determine location, the software executes a single scan and then compares the output to the stored data sets created through the setup process (method of comparison is discussed further in 4.2.3). The program then indicates where it most likely is in the building.

4.2.2 Data Storage

Once the setup process is complete, the data set related to each visited point is stored in a text file. On startup, the program parses through the text file associated with the current building (selected by a backend user) and initializes two lists. The first of these lists holds the registered names of every stored location. The second holds a list of lists where each element stores all the access points and associated signal ranges. These two lists are related by index, the first location name corresponds to the first data set, the second name to the second data set, and so on. This data can be updated by a backend user during execution either by adding a new area, removing an existing one, or simply renaming a location. On shutdown, if the data has been updated the text file will also be rewritten to reflect all changes made.

4.2.3 Navigational System

With the data storage and Wi-Fi scanning modules complete, our team could move onto the process of navigating through a building which is the entire purpose of our software. Consequently, the navigation module has received a substantial amount of time being designed and thoroughly tested. This module relies on one main idea: when a current Wi-Fi scan is compared to all stored locations, the majority of the time the new scan will be most similar to the data set for the location where the new scan is closest to. Wi-Fi signal strength is not absolutely consistent and so there are instances where the program does not output the location that is actually closest. When a Wi-Fi scan and a location are compared the location is assigned a rating, once every location is tested against the area that has been given the highest rating is returned as the place that the device is most likely closest to. A location's rating can be between -100 and 100, with 100 being the best possible score. Ratings are calculated by directly comparing an access point in a new scan with the same access point in the location's stored data set. The closer to the average value the new scan's value is, the larger the increase to the location's rating. Similarly, signal values outside the average range decrease the rating.

There are two types of stored locations that a building will have: a single point and an edge. A single point will be used when defining hallway intersections or specific rooms and will be acquired by standing in the same spot with the software while it repeatedly scans nearby Wi-Fi signals. An edge will represent a length of hallway that is acquired by running the same repeated scanning procedure as a single point, but the backend user will walk back and forth in the designated stretch of the hallway.

4.2.4 Point to Point

This software uses these edges and single points in order to navigate from one registered point in the building to another. When testing a current scan against an edge, that location's rating is highest when closer to the middle of the edge. To know when the device has arrived at a location, this steady decline in the edge's rating is used in conjunction with the sudden spike of a single point's rating. To move between any two points, A and B, their shortest path is calculated and the previous comparison process is used for each registered point between A and B.

4.2.5 Tour Functionality

This system's tour functionality is just an extension of the point to point traversal. The tour for a given building may, for a simple example, guide users to lab rooms on the first floor, professors' offices on the second floor, and then a study room on the third floor all while also showing the user to the bathrooms on each floor. This tour route would be designed by a backend user. Wi-Fi scanning, data storage, general navigation, point to point navigation, and tour functionality comprise the backend module of this software. When completed, the backend will communicate with the frontend to display direction information to the user. When this software is eventually integrated with the robot, this information is used to direct the robot through any building.

5. Implementation Plan

Our plan is to have all modules up and running by early April. This will allow us to have sufficient time to test these modules, and the system as a whole for bugs or errors. We have broken up the tasks into three categories, frontend, backend, and general. The front and backend tasks are concerned with their respective modules, while the general tasks deal with physically mapping out the buildings to be used by the former two modules. This will allow us to create smaller teams to focus on one side of the project at a time. Most of our modules also build on one another meaning once one module is complete work on the next one will begin. Also, some frontend tasks require that backend tasks be complete thus they are spaced accordingly. Below is a Gantt chart showing the timeline we expect to follow for this project.

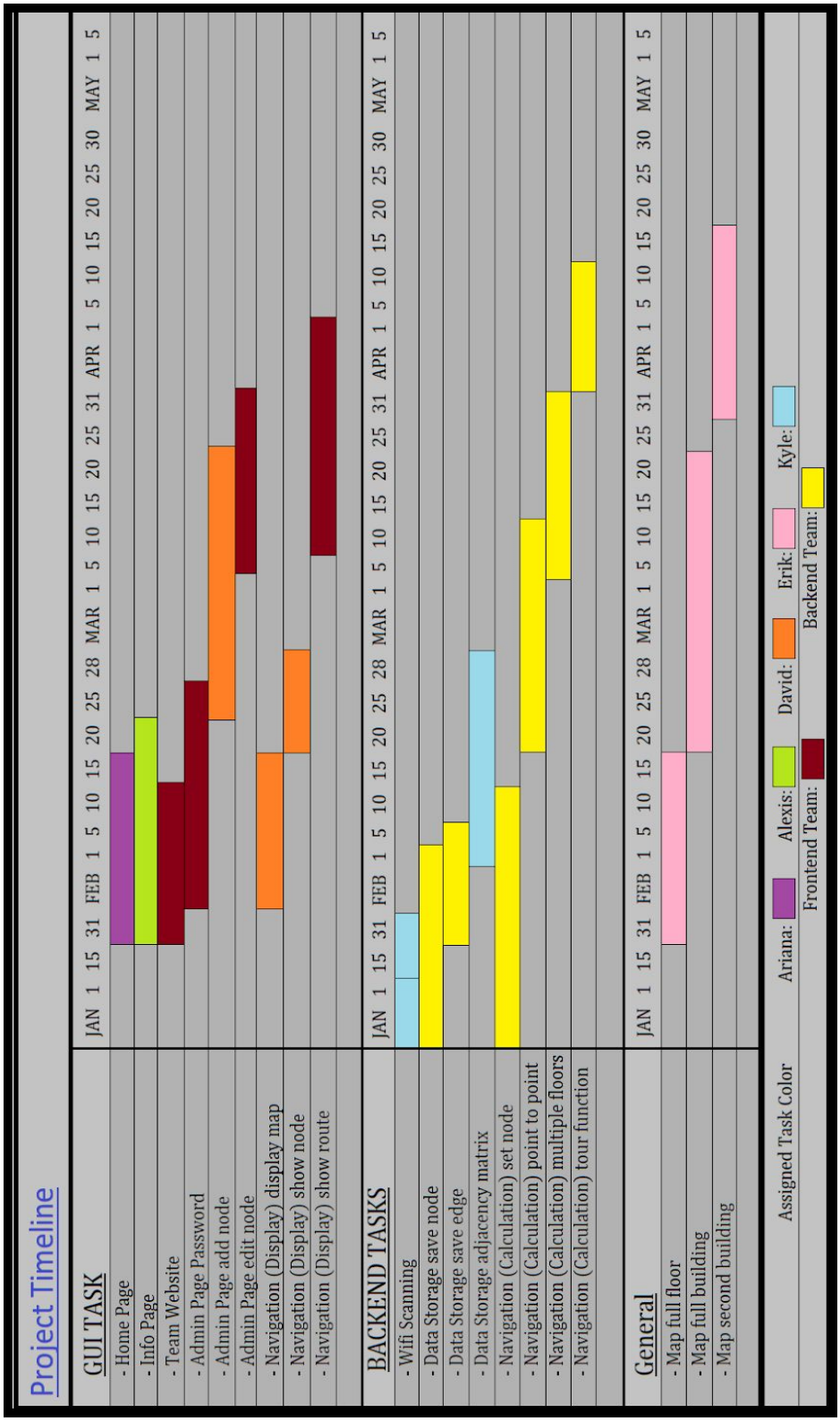

Figure 5.1: Gantt Chart Detailing the Remaining Schedule for this Project

6. Conclusion

With the expansion of mobile robotics capabilities and the growth of problems that can be solved by them. Mobile robotics is an increasingly valuable field and the more people who can get involved early on will help expand the subject area both further and faster. Our project, “The Thirty Gallon Robot”, seeks to demonstrate affordable ways that educational institutions can get students involved with robotics. By contributing our software to a robot that is as inexpensive as possible, our team (and previous teams) believe that we can inspire other instructors and institutes that getting their students involved with robotics is not only possible, but truly rewarding.

The specific problem our team is working to solve is designing, developing and delivering a functioning software product that utilizes the signal strengths of nearby routers, without knowing the location of those routers, to navigate a building. This problem exists because our team sponsor, Dr. Leverington, has a robot in its third year of development and it does not currently have a way to autonomously navigate the engineering building at Northern Arizona University as desired.

To solve this problem, the design of our software is split up into a front and backend module. The frontend module is responsible for communicating directly with the user via a Graphical User Interface (GUI). This GUI will be able to calculate the shortest path to a user selected destination and allows the software to be set up in a new building. While the backend module will perform all of the Wi-Fi scanning necessary to determine the device’s position and send directions to the GUI. Once the software is eventually integrated into the robot, it will use our system’s directions to give automated tours of any building that has been mapped.