Shape

Description automatically generated with low confidence

**MightyMover Market/Product Requirements**

CPE 4800 - Senior Design Proposal

Spring 2023

Kennesaw State University

Faculty: Dr. Jeffrey L Yiin

Team Members:

Grant Burke

Hanson Chaney

Cooper Newlin

Chris Turner

Rev. Number: Draft

March 25, 2023

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Grant Burke | 04-02-2023 | Initial Release | Rev. - Draft |

Table 1. Revision History

# Table of Contents

Contents

[Revision History 2](#_Toc131364098)

[Table of Contents 3](#_Toc131364099)

[List of Figures 4](#_Toc131364100)

[List of Tables 4](#_Toc131364101)

[Introduction 5](#_Toc131364102)

[MightyMover Top-Level Product Description: 5](#_Toc131364103)

[Value Proposition for Our Customers: 6](#_Toc131364104)

[Target Market Analysis: 6](#_Toc131364105)

[Visual Representation of MightyMover: 7](#_Toc131364106)

[Marketing Requirements 7](#_Toc131364107)

[System Architecture 9](#_Toc131364108)

[System Block Diagram of MightyMover 9](#_Toc131364109)

[Block Diagrams of Subsystems 10](#_Toc131364110)

[Phone-to-MightyMover System 10](#_Toc131364111)

[MightyMover Motor Control System 11](#_Toc131364112)

[MightyMover LED Control System 11](#_Toc131364113)

[LiDAR Subsystem 12](#_Toc131364114)

[Software Hub Diagram 12](#_Toc131364115)

[Battery Management System 13](#_Toc131364116)

[Use-Case Analysis 14](#_Toc131364117)

[Case 1 - Turn on MightyMover 14](#_Toc131364118)

[Case 2 - Initiate Following Through Use of Mobile App 15](#_Toc131364119)

[Case 3 – MightyMover is too close to an obstacle and must stop 16](#_Toc131364120)

[Bill of Materials 17](#_Toc131364121)

[Product Requirements 18](#_Toc131364122)

[Bibliography 21](#_Toc131364123)

# List of Figures

[Figure 1. Visual Representation of MightyMover 7](#_Toc131364131)

[Figure 2. Top-Level Block Diagram of MightyMover System 9](#_Toc131364132)

[Figure 3. Basic Flow of Phone-to-MightyMover Communication 10](#_Toc131364133)

[Figure 4. Motor Control System on MightyMover 11](#_Toc131364134)

[Figure 5. LED Control System on MightyMover 11](#_Toc131364135)

[Figure 6. LIDAR Subsystem 12](#_Toc131364136)

[Figure 7. Software Hub Block Diagram 12](#_Toc131364137)

[Figure 8. Battery Management System 13](#_Toc131364138)

[Figure 9. Case 1 - Turn on MightyMover 14](#_Toc131364139)

[Figure 10. Case 2 - Initiate Following with Mobile App 15](#_Toc131364140)

[Figure 11. Case 3 - MightyMover is too close to an obstacle 16](#_Toc131364141)

# List of Tables

[Table 1. Revision History 2](#_Toc131364142)

[Table 2. Marketing Requirements 8](#_Toc131364143)

[Table 3. Bill of Materials 17](#_Toc131364144)

[Table 4. Product Requirements 20](#_Toc131364145)

# Introduction

The MightyMover is an autonomously following cart with GPS location control that eases the burden of carrying heavy objects. The top 3 problems that we found our customers facing were carrying their gardening items around, getting old enough where extra help is needed, and having physical limitations where carrying items seems impossible. We know our product solves their problem by carrying items autonomously or by waypoint easing physical strain.

The feature set of MightyMover includes a mobile app, a pathway recording system, emergency brake system, full suspension, and LIDAR obstacle avoidance. The mobile app consists of mode selection, battery information, and location monitoring. The mobile application is meant to have an easy to learn layout, so our targeted customers do not have trouble using our products. This easy to learn layout will include concepts such as a minimum number of buttons to click, so it does not look crowded and is easy to work with. In the mobile application, there will be a button to record pathways, so that the MightyMover can navigate that path without needing the user. With the emergency brake, users will not worry about the MightyMover moving while the user inserts items or if parked on an incline. The MightyMover’s suspension will provide good steering and mobility; this will make items less likely to fall out due to absorbing and dampening shock.

## MightyMover Top-Level Product Description:

MightyMover leaves standby and moves after checking initial conditions for emergency stop and object detection. The User can stop the mover during anytime using the app’s stop function, once the mover stops, it will then enter standby to await user input.

Under Location Mode, the mover will be following a set GPS course created by the user which will send updated coordinates via the app once the mover begins. If the mover leaves Bluetooth range of the user, a command file will run sending the mover to its home destination after a set amount of time.

Under Following Mode, the mover follows behind the user a set distance. During the Following Mode, the user can record the journey via waypoints under approval by the user through the app. This recording will be used to create paths to set locations for Location Mode. While the mover follows, multiple checks from our LiDAR and GPS inputs influence the mover’s behavior; as an example, if an object comes within a set distance to the mover, the mover will stop and wait for the object to move before setting off to catch up to the user. This LIDAR system will allow the mover to determine if it is able to fit into small spaces or doorways, if not, the user will be notified via the app.

## Value Proposition for Our Customers:

The time and stress saved from not having to move your equipment around your property will allow you to increase your efficiency and your productivity. The MightyMover allows you to work without outside help, those who may be physically challenged will no longer need a second hand to assist them in doing the things they love. With just two modes, our MightyMover can be used by all ages and for almost any outdoor work activity.

## Target Market Analysis:

American adults spend $47.8 billion (average household spent $503) on gardening each year and that number has only increased since the start of the pandemic. The highest spenders are between the ages of 35-44 years old; however, this age group is a perfect candidate for our MightyMover. The population size of those in America who are 65 or older has reached 54.1 million, and those in this group are spending more on gardening each year.

Roughly 44 million Americans suffer from a disability which is 13.4% of the total population. Within this demographic, many are farmers, gardeners and landscapers. Those that think of a disability gravitate towards physical limitations, but many disabilities are unseen such as cognitive, vision and hearing impairment.

Even if we had below 1% of this market, there is still a lot of profit to be made. After our surveying, we found a market for commercial use in the textile industry which would open more opportunities, but as for now, we are focusing on a personal use, general-purpose wagon.

Competition would consist of wagons, push carts, and dollies. These all require pushing or pulling which can be strenuous on the user. There are also electric pallet jacks and similar products, but these are mostly used in commercial settings which are not our focused market. Other competitors closely related are the food delivery robot and the auto-following suitcase, but these are a niche market for food service and personal use. From what we have researched, our product is the first of its kind to target a consumer market dedicated to outside use.

## Visual Representation of MightyMover:

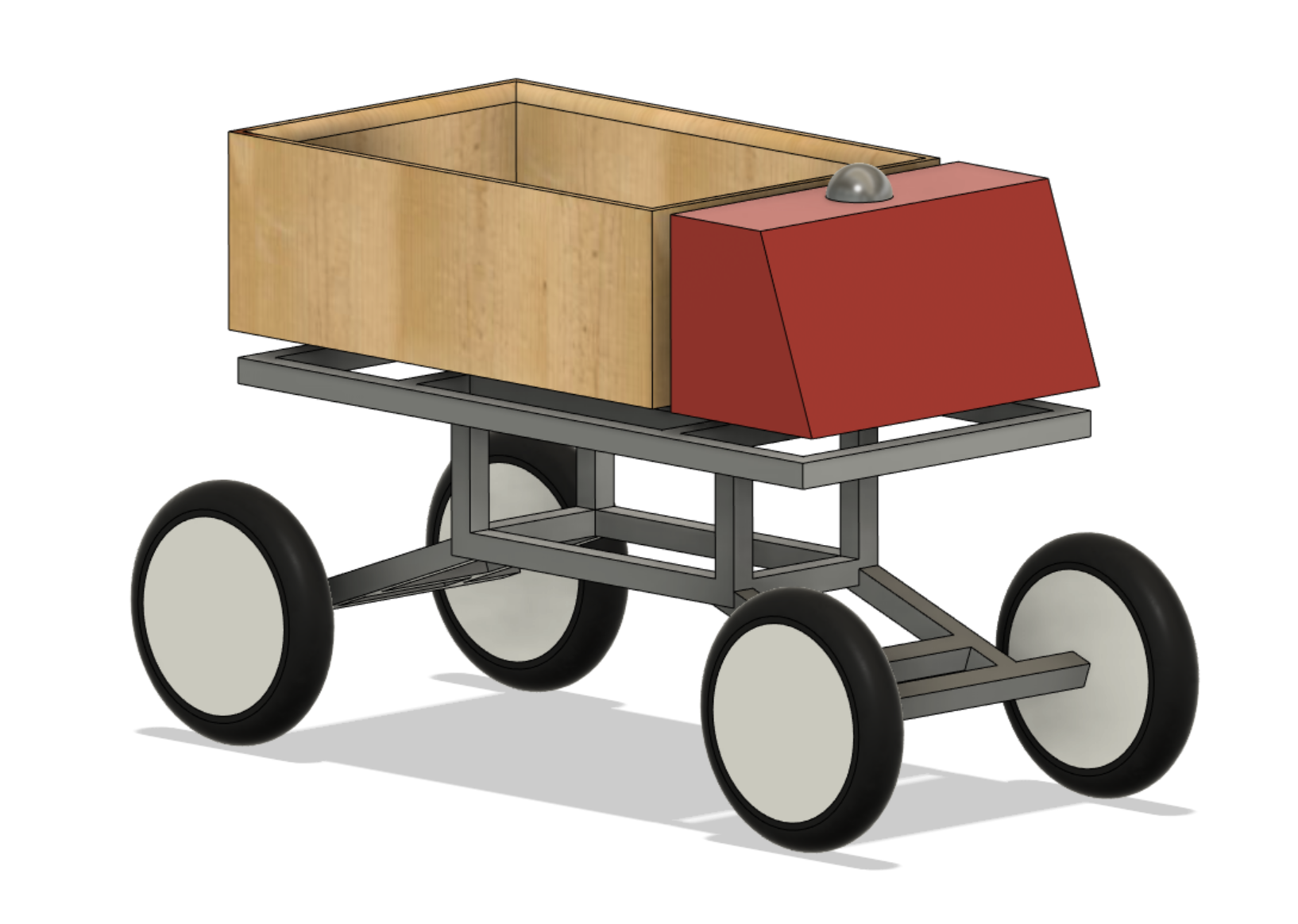


Figure 1. Visual Representation of MightyMover

# Marketing Requirements

*Table 2. Marketing Requirements*

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID:** | **Requirement** | **Type** | **Criticality** |
| MR-FNC-01 | The MightyMover shall follow the user which shall allow the user to get to their destination without having to push or pull on the cart. | Function | Must Have |
| MR-FNC-02 | The Mover shall navigate to set waypoints instead of following the user. | Function | Must Have |
| MR-FNC-03 | The Mover shall include an emergency stop system to prevent movement during loading or operation. | Function | Must Have |
| MR-FNC-04 | The Mover shall possess the ability to detect objects in its way. | Function | Must Have |
| MR-FNC-05 | The Mover shall have an indicator of the status of the battery. | Function | Nice to Have |
| MR-PER-01 | The Mover shall have a suspension system suitable for driving on uneven terrain. | Performance | Must Have |
| MR-PER-02 | The Mover shall move efficiently to where the mover will never come to a stop while navigating through off- road terrain. | Performance | Must Have |
| MR-PER-03 | The connection between the phone to the Mover shall send signals in less than 15 seconds. | Performance | Must Have |
| MR-PER-04 | The Mover’s emergency stop system must stop the Mover under 2 seconds. | Performance | Must Have |
| MR-PER-05 | The Mover shall provide a battery level of at least 4 consecutive hours of MightyMover moving. | Performance | Must Have |
| MR-FIT-01 | The system shall provide a mobile application to control the operation of the cart. | Fit and Form | Must Have |
| MR-FIT-02 | The electronics of the cart shall be housed in a 3D-printed water-resistant enclosure. | Fit and Form | Must Have |
| MR-IFC-01 | All cart controls shall be accessed remotely via Bluetooth. | Interface | Must Have |
| MR-IFC-02 | Emergency stop shall be accessed in the app. | Interface | Nice to Have |
| MR-IFC-03 | Emergency stop shall be accessed physically on the Mover. | Interface | Must Have |
| MR-CON-01 | The user shall be able to wirelessly control the cart within a 10-meter range. | Constraint | Must Have |
| MR-CON-02 | The market price of the Mover must not exceed $2000. | Constraint | Nice to Have |
| MR-CON-03 | The Mover’s dimensions shall be able to move potted plants. | Constraint | Must Have |

Table 2. Marketing Requirements

# System Architecture

## System Block Diagram of MightyMover

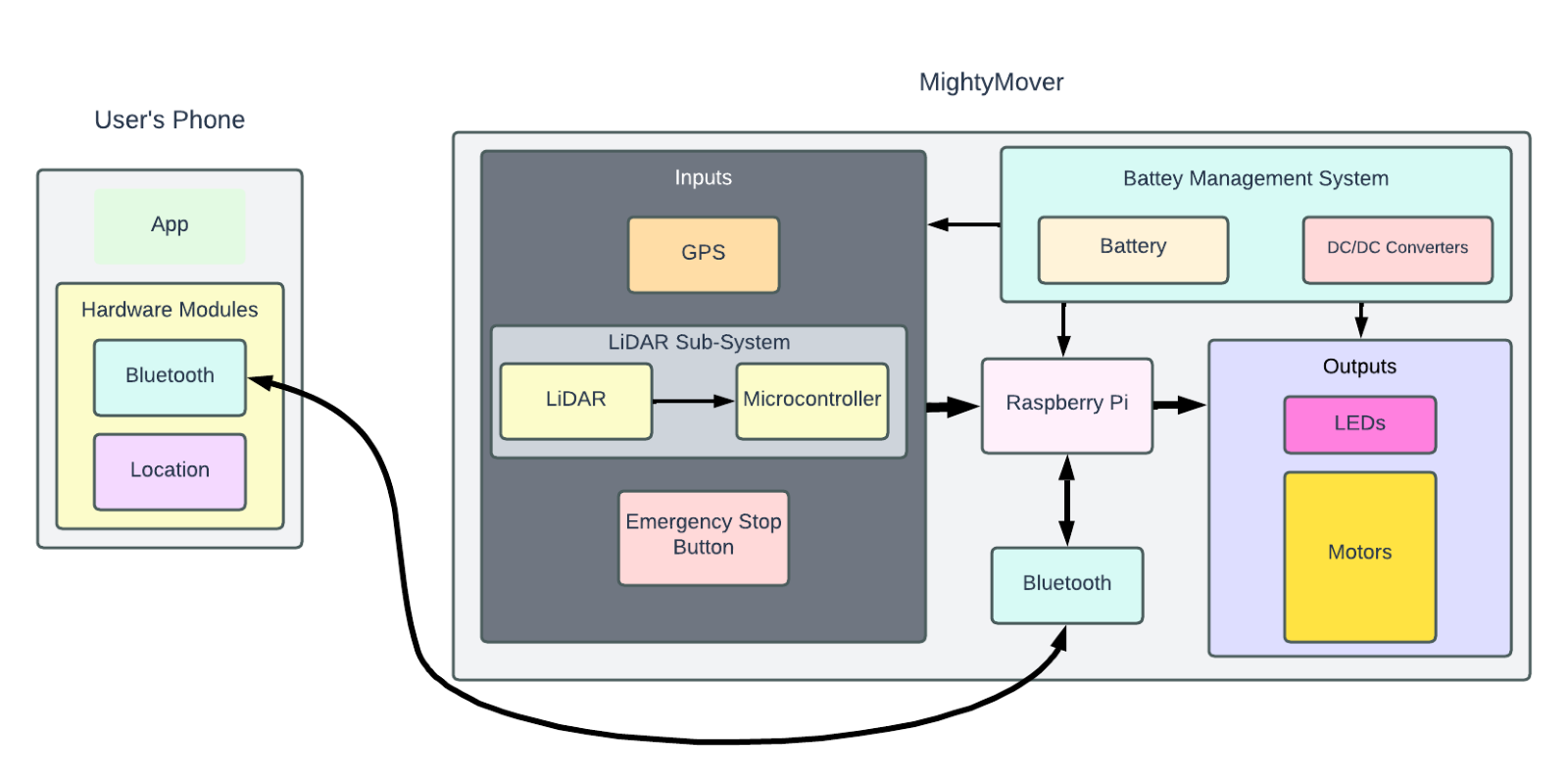


Figure 2. Top-Level Block Diagram of MightyMover System

MightyMover system is a two-component system including the user’s phone and the MightyMover itself. The user’s phone will have an app compatible to MightyMover. The user will send instructions and get feedback over a Bluetooth connection. The raspberry pi will process Bluetooth communication, object detection code, and PWM output code (for the motors). MightyMover includes subsystems such as phone-to-MightyMover communications, MightyMover motor control system, MightyMover LED control system, LiDAR sub-system, software hub design, and battery management.

## Block Diagrams of Subsystems

### Phone-to-MightyMover System

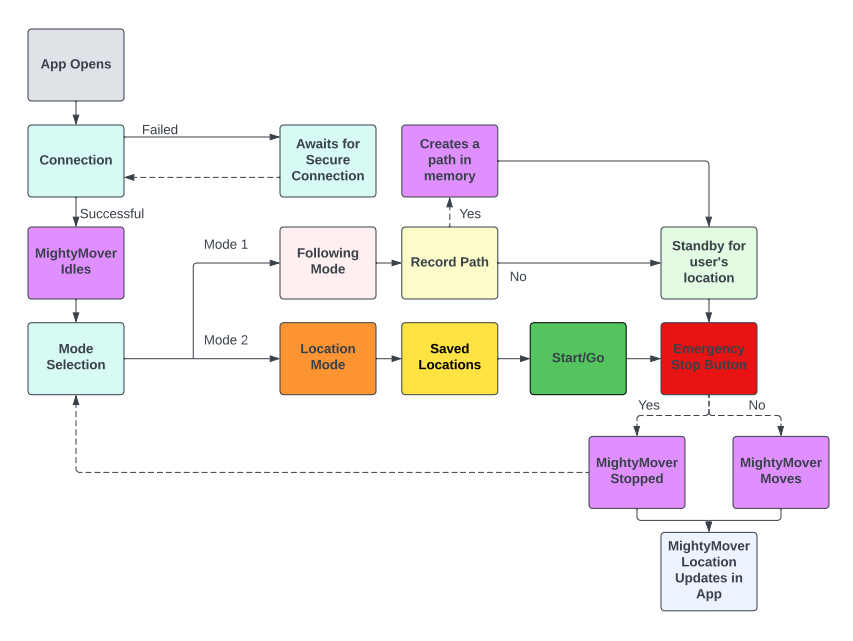


Figure 3. Basic Flow of Phone-to-MightyMover Communication

The first sub-system demonstrates the process in which the mobile app communicates to MightyMover based on user input. First, the app must make a successful Bluetooth connection, then the MightyMover idles while getting inputs from user. Depending on the user’s inputs (different colors), the MightyMover will respond (in purple). In this case, the red emergency stop button is both in the app and physical.

### MightyMover Motor Control System

Diagram

Description automatically generated

Figure 4. Motor Control System on MightyMover

The motors determine where the MightyMover moves, and this is partially the main function of the cart. The motor control system is specifically designed by the code on the raspberry pi. Before this code is executed a lot of parameters are being weighed. This includes previous codes and inputs both from MightyMover and the user’s phone. Overall, the diagram above shows what the system controlling the motors is.

### MightyMover LED Control System

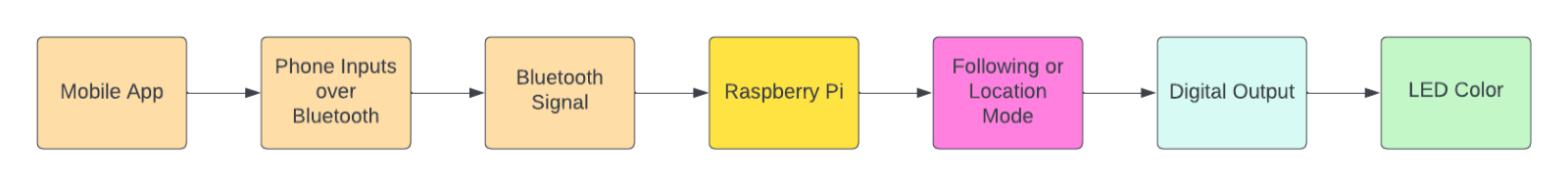


Figure 5. LED Control System on MightyMover

The LEDs show which mode the cart is currently in and if your system is on. This will help the user understand what the cart is doing and will help the team in debugging.

### LiDAR Subsystem

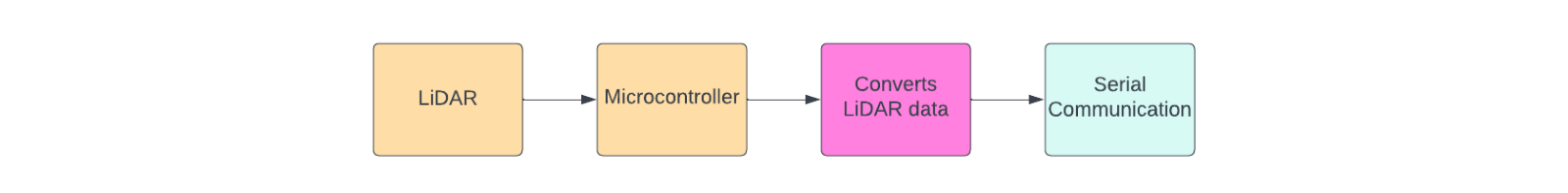


Figure 6. LIDAR Subsystem

The LiDAR system will include a microcontroller to process the LiDAR data to convert into serial communication. This will help determine objects around the path of MightyMover. We are thinking that the microcontroller may also be able to run the object detection code that we initially have on the Raspberry Pi. For now, we plan to have the raspberry pi running the code for simplicity.

### Software Hub Diagram

Diagram

Description automatically generated

Figure 7. Software Hub Block Diagram

The Raspberry Pi will take in information from the mobile app via Bluetooth signal, GPS location, and LIDAR input from its microcontroller. These inputs will feed our multiple command files when they are executed. As the command files run, they will then continuously pull data from each sensor and produce a PWM output to the motors. The emergency stop, lost signal, and object detection command files will hold higher precedence over the location and following command file. If any condition within the red colored blocks (command files) are met, they will run thus stopping the following or location command files from running.

### Battery Management System

Diagram

Description automatically generated

Figure 8. Battery Management System

The battery management system is simple to understand. We are still in the process of picking parts out, so most of these materials are still being determined, but this is the idea so far. If things change to a low voltage system, we may need a different DC to DC converter.

# Use-Case Analysis

## Case 1 - Turn on MightyMover

* *Description:* 
  + Turn on MightyMover
* *Actors:* 
  + User, power switch, power system, phone, CPU, operating system, Bluetooth module
* *Basic flow:*
  + The user flips the power switch which allows the SoC to receive power. This then requires the operating system and modules to boot up. The Bluetooth module awaits signal from the user’s phone. Now that everything in the process has successfully happened, the MightyMover is ready for the user’s instruction.
* *Alternative/exception flows:* 
  + Battery is not charged, System cannot start, Error in boot process
* *Outcome:* 
  + The MightyMover is powered on and ready to start following.
* *Swim Lane Diagram:*

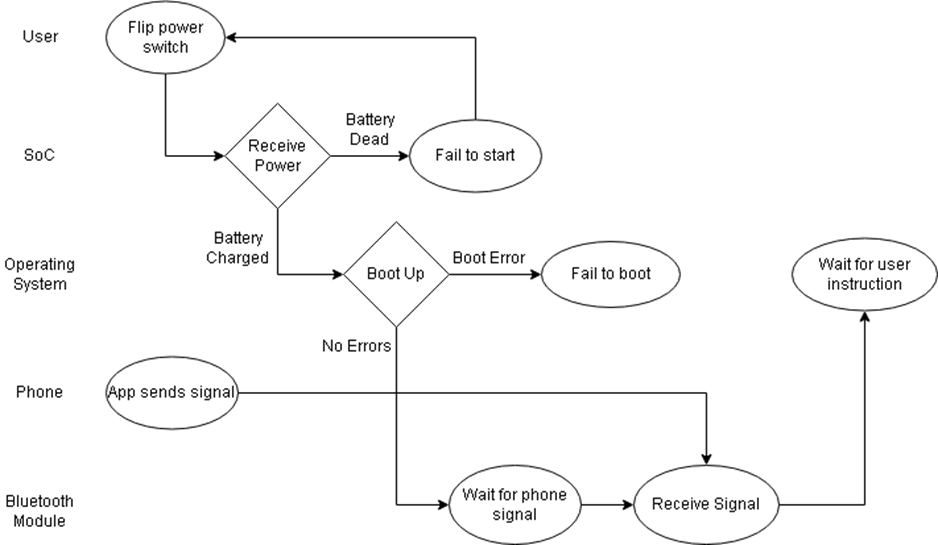


Figure 9. Case 1 - Turn on MightyMover

## Case 2 - Initiate Following Through Use of Mobile App

* *Description:* 
  + The user can activate the following mode from an app on their phone.
* *Actors:* 
  + User, cell phone, mobile app, CPU, Bluetooth module, motors
* *Basic flow:*
  + The user opens the app. The app then waits for the user to press the following mode button. Once this is done, the phone sends a Bluetooth signal to the Bluetooth module. This signal is processed by the SoC, and the SoC sends the appropriate electrical signal to the motors so the mover can travel. If the Bluetooth module cannot receive a signal, it will wait until the user sends another signal from the app.
* *Alternative/exception flows:* 
  + App can’t establish connection to MightyMover.
  + App establishes connection with MightyMover, but MightyMover can’t start following the user.
* *Outcome:*
  + MightyMover begins following user.
* *Swim Lane Diagram:*

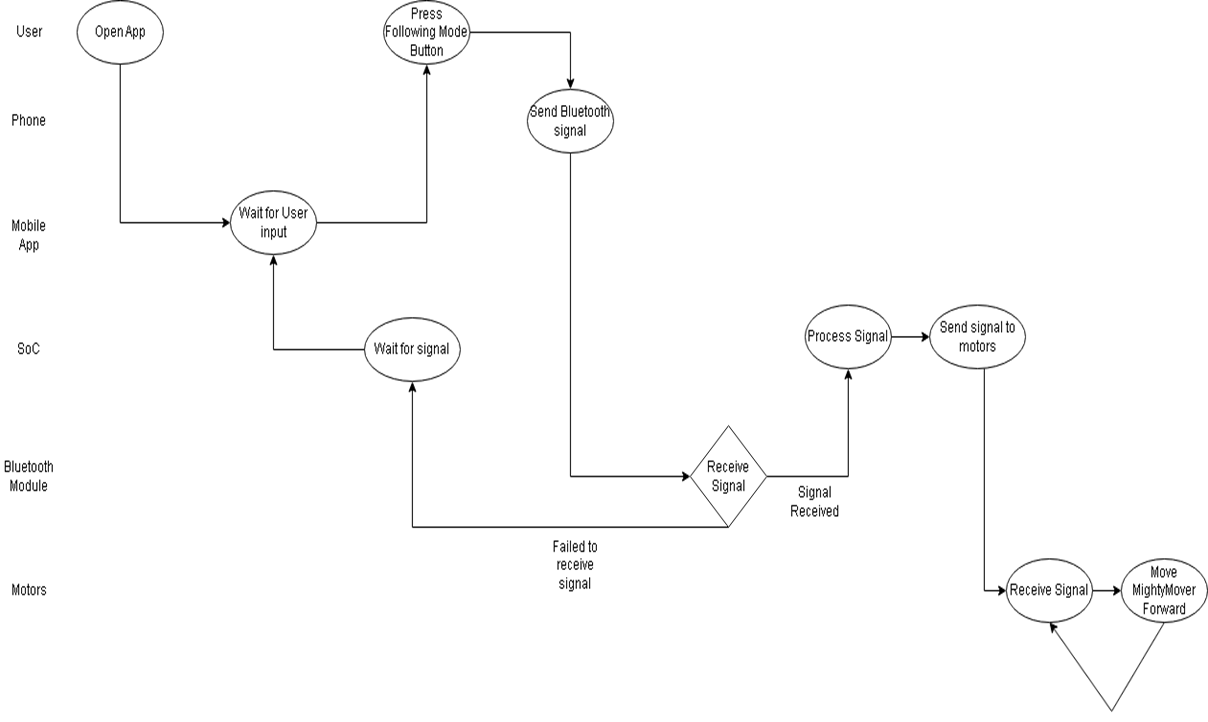


Figure 10. Case 2 - Initiate Following with Mobile App

## Case 3 – MightyMover is too close to an obstacle and must stop

* *Description:* 
  + The MightyMover detects when it is too close to an object and must stop or travel around it to avoid collision and proceed on its path.
* *Actors:* 
  + User, Mobile App, Cellphone, CPU, Bluetooth module, motors, Distance Sensor
* *Basic Flow:* 
  + The Mover detects an object in its path and stops to avoid collision. It then detects an alternative path and travels that path to proceed with its set trajectory.
* *Alternative/Exception Flows:*
  + The Mover is far enough away from any obstacle and can proceed to travel on its path.
* *Swim Lane Diagram:*

Icon

Description automatically generated with medium confidence

Figure 11. Case 3 - MightyMover is too close to an obstacle

# Bill of Materials

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part | Model Number | Quantity | Price | Provided |
| Raspberry Pi 4 (2GB) | SC15184 | 1 | $139.99 | Yes |
| Slamtec RPLIDAR A1M8 2D 360 Degree 12 Meters  Scanning Radius LIDAR  Sensor Scanner | RPLIDAR A1M8 | 1 | $99.99 | No |
| 2020 T-slot Aluminum Extrusion | EXT-2020-COMBO | 1 | $99.95 | No |
| Steel Solid Sheet Metal | GVL0108 | 1 | $14.48 | No |
| Overture PETG filament - 1 kg | OVPETG175 | 1 | $20.99 | No |
| SAMSUNG EVO Plus w/SD Adaptor 128GB Micro SDXC | MB-MC128KA/AM | 1 | $13.81 | Yes |
| 8 Inch Wheels | B08SJJL8KB | 4 | $57.58 | No |
| Shock Absorbers | W15128190153 | 4 | $67.16 | No |
| Motors | XD-3420 | 2 | $31.97 | No |
| Bluetooth Module | 8541554474 | 1 | $10.39 | No |
| Arduino Mega 2560 | Mega 2560 Rev3 | 1 | $48.40 | Yes |
| GPS Module | 3-01-1541 | 1 | $11.39 | No |
| Lithium-Ion Battery | NM-12V12Ah | 1 | $49.99 | No |
|  |  | Total: | $666.09 | $463.89 |

Table 3. Bill of Materials

# Product Requirements

|  |  |
| --- | --- |
| ID-Name | MOVER-SYS-001: Phone to MightyMover |
| Description | This sub system is the process of the phone and the MightyMover communicating. The user will have feedback from MightyMover showing the battery percentage, location monitoring, and the current mode of the Mover. The app will start with a mode selection which would be following or location mode. Then prompts will display with the user putting in further information to help MightyMover get instructions. The app shall always show an emergency stop button, so when the user sees a situation, they can wirelessly shut the Mover down. |
| Key Components | Mobile App, Bluetooth Module, Raspberry Pi |
| Interfaces | Xcode, Android Studio, Linux OS |
| Requirements | PRD-SYS-001 – Must have Bluetooth 5.0 Connectivity.  PRD-SYS-002 – Must Receive Bluetooth Signal up to 50 meters away from module.  PRD-SYS-003 – Must transfer data at a speed of at least 2 Mbit/s. |
| ID-Name | MOVER-SYS-002: MightyMover Motor Control System |
| Description | The motors determine where the MightyMover moves. The motors will be controlled by a Raspberry Pi PWM GPIO port. The Raspberry Pi will make decisions based on the user’s phone location and all the sensors on the Mover. |
| Key Components | Raspberry Pi, LiDAR sensor, GPS module, mobile app, Bluetooth Module. Motors, |
| Interfaces | Linux OS |
| Requirements | PRD-SYS-004 – Motor must consume no more than 12V.  PRD-SYS-005 – Motor must possess 30 Watts of horsepower.  PRD-SYS-006 – Motor must be smaller than 2x2x3 inches. |
| ID-Name | MOVER-SYS-003: MightyMover LED Control System |
| Description | The LEDs show which mode the cart is currently in along with the power on LED. These LEDs will be controlled by the Raspberry Pi which will get the mode it’s currently in from the user’s phone. This will help the user understand what the cart is doing and will help the team in debugging. |
| Key Components | LEDs, Raspberry Pi, Mobile App |
| Interfaces | Linux OS, Xcode, Android Studio |
| Requirements | PRD-SYS-007- The LEDs must operate in accordance with the RGB color model.  PRD-SYS-008- The LEDs shall be always on, even during motion.  PRD-SYS-009- The LEDs shall use no more than 20mA of current. |
| ID-Name | MOVER-SYS-004: LiDAR Subsystem |
| Description | The LiDAR system will include a microcontroller to process the LiDAR data to convert into serial communication for the Raspberry Pi. We are currently looking into ways that we can use the LiDAR system to get more accurate location than a GPS module. This system is very important because it will help determine object detection, help create a path in memory, and make sure we are following the user. We are thinking that the microcontroller may also be able to run the object detection code that we initially have on the Raspberry Pi. For now, we plan to have the raspberry pi running the code for simplicity. |
| Key Components | LiDAR sensor, Raspberry Pi |
| Interfaces | Linux OS |
| Requirements | PRD-SYS-010- The LiDAR sensor must be capable of sensing objects within 12 meters of the device.  PRD-SYS-011- The LiDAR sensor must have a sample rate of 8000 Times.  PRD-SYS-012- The LiDAR sensor must weigh no more than 1 pound. |
| ID-Name | MOVER-SYS-005: Software Hub |
| Description | The Raspberry Pi is the decision maker of the Mover. The RPi shall take in information from the mobile app via Bluetooth signal, GPS location, and LIDAR input from its microcontroller. The app data will be used to determine which mode that the Mover will be in as well as determine if the Mover needs to stop (emergency stop). The GPS location will be used to determine the next point that the Mover needs to move to. Finally, the LiDAR sub-system will run object detection and help with PWM outputs to the motor. These inputs will feed into many files when they are executed. The emergency stop, lost signal, and object detection command files will hold higher precedence over the location and following command file because these are the safety features to the Mover. |
| Key Components | Raspberry Pi, GPS Module, LiDAR Module, Bluetooth Module |
| Interfaces | Linux OS |
| Requirements | PRD-SYS-013- System must start all applications within one minute of turning on.  PRD-SYS-014- System must accept all data sent from sensors.  PRD-SYS-015- System must determine mode from app signal. |
| ID-Name | MOVER-SYS-006: Battery Management System |
| Description | The battery management system sends the power to mostly every component of the Mover. We are still in the process of picking parts out, so most of these materials are still being determined, but this is the idea so far. If things change to a low voltage system, we may need a different DC to DC converter. |
| Key Components | Battery, DC-DC Converter |
| Interfaces | None |
| Requirements | PRD-SYS-016- Battery must have a capacity of 12 Ah.  PRD-SYS-017- Battery must weigh less than 4 pounds.  PRD-SYS-018- The battery must be 5.94 x 3.81 x 3.71 inches in size. |

Table 4. Product Requirements

# **Bibliography**

1. Chu, Hung-Chi.”Design and implementation of an auto-following robot-car system for the elderly.” 2016 International Conference on System Science and Engineering (ICSSE), July 2016.
2. Wang, Y., Mulvaney, D., Sillitoe, I. et al. Robot Navigation by Waypoints. J Intell Robot Syst 52, 175–207 (2008).
3. Monarca, D.; Rossi, P.; Alemanno, R.; Cossio, F.; Nepa, P.; Motroni, A.; Gabbrielli, R.; Pirozzi, M.; Console, C.; Cecchini, M. Autonomous Vehicles Management in Agriculture with Bluetooth Low Energy (BLE) and Passive Radio Frequency Identification (RFID) for Obstacle Avoidance. Sustainability 2022, 14, 9393.