

Vehicle Mileage Tracker

Project Document

by
MingChu Chiu
James Ewing
Caeleb Lacey

12 October, 2022

|

9 June, 2023

Table of Contents

Vehicle Mileage Tracker	1
Project Document	1
1 Overview	2
1.1 Executive Summary	2
1.2 Team Contacts and Protocols	3
1.3 Gap Analysis	3
1.4 Timeline/Proposed Timeline	4
1.5 References and File Links	4
1.6 Revision Table	4
2 Impact and Risks	4
2.1 Design Impact Statement	4
2.2 Risks	7
2.3 References and File Links	7
2.3.1 References	7
2.3.2 File Links	9
2.4 Revision Table	9
3 Top-Level Architecture	9
3.1 Block Diagram	9
3.2 Block Descriptions	9
3.3 Interface Definitions	14
3.4 References and File Links	15
3.4.1 References	15
3.4.2 File Links	15
3.5 Revision Table	15
4 Block Validations	15
4.1 Bluetooth Module	15
4.1.1 Description	15
4.1.2 Design	16
4.1.3 General Validation	16
4.1.4 Interface Validation	17
4.1.5 Verification Process	21
4.1.6 References and File Links	21
4.1.7 Revision Table	21
4.2 Bluetooth Handshake	21
4.2.1 Description	21

4.2.2 Design	21
4.2.3 General Validation	22
4.2.4 Interface Validation	22
4.2.5 Verification Process	24
4.2.6 References and File Links	24
4.2.7 Revision Table	24
4.3 Mobile App	24
4.3.1 Description	24
4.3.2 Design	25
4.3.3 General Validation	26
4.3.4 Interface Validation	27
4.3.5 Verification Process	31
4.3.6 References and File Links	32
4.3.7 Revision Table	32
4.4 File Management	33
4.4.1 Description	33
4.4.2 Design	33
4.4.3 General Validation	33
4.4.4 Interface Validation	33
4.4.5 Verification Process	33
4.4.6 References and File Links	33
4.4.7 Revision Table	33
4.5 External Storage Hub	33
4.5.1 Description	33
4.5.2 Design	33
4.5.3 General Validation	33
4.5.4 Interface Validation	33
4.5.5 Verification Process	33
4.5.6 References and File Links	33
4.5.7 Revision Table	33
4.6 Microcontroller	33
4.6.1 Description	33
4.6.2 Design	33
4.6.3 General Validation	34
4.6.4 Interface Validation	34
4.6.5 Verification Process	34
4.6.6 References and File Links	34
4.6.7 Revision Table	34
4.7 Distance Processing	34
4.7.1 Description	34
4.7.2 Design	34

4.7.3 General Validation	34
4.7.4 Interface Validation	36
4.7.5 Verification Process	39
4.7.6 References and File Links	42
4.7.7 Revision Table	42
4.8 Enclosure	42
4.8.1 Description	42
4.8.2 Design	43
4.8.3 General Validation	44
4.8.4 Interface Validation	46
4.8.5 Verification Process	47
4.8.6 References and File Links	47
4.8.7 Revision Table	47
4.9 HMI	47
4.9.1 Description	47
4.9.2 Design	48
4.9.3 General Validation	49
4.9.4 Interface Validation	51
4.9.5 Verification Process	53
4.9.6 References and File Links	53
4.9.7 Revision Table	53
4.10 GPS Module	53
4.10.1 Description	53
4.10.2 Design	54
4.10.3 General Validation	55
4.10.4 Interface Validation	57
4.10.5 Verification Process	59
2. Turn the device on.	59
3. Using Business Mode of the device, record a trip that accords to the created route.	59
4. Compare the recorded GPS distance with the mileage shown on Google Maps.	59
4.10.6 References and File Links	59
4.10.7 Revision Table	59
4.11 Power Stepper	59
4.11.1 Description	59
4.11.2 Design	59
4.11.3 General Validation	61
4.11.4 Interface Validation	64
4.11.5 Verification Process	66
4.11.6 References and File Links	69
4.11.7 Revision Table	70

4.12 Trip ID	70
4.12.1 Description	70
4.12.2 Design	70
4.12.3 General Validation	70
4.12.4 Interface Validation	70
4.12.5 Verification Process	70
4.12.6 References and File Links	70
4.12.7 Revision Table	70
4.13 Main Code	70
4.13.1 Description	70
4.13.2 Design	70
4.13.3 General Validation	70
4.13.4 Interface Validation	70
4.13.5 Verification Process	70
4.13.6 References and File Links	70
4.13.7 Revision Table	70
5 System Verification Evidence	70
5.1 Universal Constraints	70
5.1.1 The system may not include a breadboard	70
5.1.2 The final system must contain a student designed PCB with greater than 30 pads.	70
5.1.3 All connections to PCBs must use connectors.	70
5.1.4 All power supplies in the system must be at least 65% efficient.	70
5.1.5 The system may be no more than 50% built from purchased 'modules.'	71
5.2 Project Requirements	71
5.2.1. Bluetooth Configurable	71
5.2.1.1. Project Partner Requirement:	71
5.2.1.2. Engineering Requirement:	71
5.2.1.3. Testing Method:	71
5.2.1.4. Verification Process:	71
5.2.1.5. Pass Condition:	71
5.2.1.6. Testing Evidence:	71
5.2.2. Compact Packaging	71
5.2.2.1 Project Partner Requirement:	71
5.2.2.2 Engineering Requirement:	72
5.2.2.3 Testing Method:	72
5.2.2.4 Verification Process:	72
5.2.2.5 Pass Condition:	72
5.2.2.6 Testing Evidence:	72
5.2.3. Driver Safety	74

5.2.3.1 Project Partner Requirement:	74
5.2.3.2 Engineering Requirement:	74
5.2.3.3 Testing Method:	75
5.2.3.4 Verification Process:	75
5.2.3.5 Pass Condition:	77
5.2.3.6 Testing Evidence:	77
5.2.4. End User Documentation	77
5.2.4.1 Project Partner Requirement:	77
5.2.4.2 Engineering Requirement:	77
5.2.4.3 Testing Method:	77
5.2.4.4 Verification Process:	77
5.2.4.5 Pass Condition:	79
5.2.4.6 Testing Evidence:	79
5.2.5. Neatly Presented Data	79
5.2.5.1 Project Partner Requirement:	79
5.2.5.2 Engineering Requirement:	79
5.2.5.3 Testing Method:	79
5.2.5.4 Verification Process:	79
5.2.5.5 Pass Condition:	79
5.2.5.6 Testing Evidence:	80
5.2.6. Tracking Vehicle Mileage	80
5.2.6.1 Project Partner Requirement:	80
5.2.6.2 Engineering Requirement:	80
5.2.6.3 Testing Method:	80
5.2.6.4 Verification Process:	80
5.2.6.5 Pass Condition:	80
5.2.6.6 Testing Evidence:	81
5.2.7. Trip Identification	82
5.2.7.1 Project Partner Requirement:	82
5.2.7.2 Engineering Requirement:	82
5.2.7.3 Testing Method:	82
5.2.7.4 Verification Process:	82
5.2.7.5 Pass Condition:	82
5.2.7.6 Testing Evidence:	82
5.2.8. Vehicle Powered	83
5.2.8.1 Project Partner Requirement:	83
5.2.8.2 Engineering Requirement:	83
5.2.8.3 Testing Method:	83
5.2.8.4 Verification Process:	83
5.2.8.5 Pass Condition:	83
5.2.8.6 Testing Evidence:	84

6 Project Closing	84
6.1 Future Recommendations	84
6.1.1 Technical Recommendations	84
6.1.2 Global Impact Recommendations	85
6.1.3 Teamwork Recommendations	86
6.2 Project Artifact Summary with Links	86
6.3 Presentation Materials	87
6.4 References and File Links	87
6.5 Revision Table	88

1 Overview

1.1 Executive Summary

The purpose of the vehicle mileage tracker is to present trip information neatly to taxpayers in a way they can easily transfer the information to their tax form. The current solutions in the market require users to pay additional subscription fees for collected data. The goal of this device is to provide users with a one-time charge solution. This device will generate a list of business mileage and destinations, with necessary summary information. It will be a small unit capable of recording location information, such as start and end GPS coordinates, and documenting mileage of vehicle trips. The additional functionality of this device includes the ability to identify when a trip starts or stops, collect and store vehicle trip information in local storage, and to identify trips as business or other via mobile app. While this project is still in a conceptual phase, the intent is to begin system testing in January and finalize a prototype by the end of April.

1.2 Team Contacts and Protocols

TABLE I
TEAM CONTACTS

Name	Role	Contact Info
Caeleb Lacey	Hardware & Software	laceyca@oregonstate.edu
Ming Chu Chiu	Software	chiumi@oregonstate.edu
James Ewing	Hardware	ewingj@oregonstate.edu

TABLE II
TEAM PROTOCOLS

Topics	Expectation
Meeting time	Weekly: Monday 2:30 pm Back up time: Thursday 9 am
Documentation	Shared on Google Drive
Discord Etiquette	<ul style="list-style-type: none">• Pin important links• At least check it once a day

Communication	<ul style="list-style-type: none"> Bring ideas forward, other members take the time to listen Update roadblocks to team
Responsibility	<ul style="list-style-type: none"> Nail down feature specs Have individual block responsibility
Coding	Make clear comments for functions and confusing lines

1.3 Gap Analysis

Small business owners and contractors can report their business-related mileage to the IRS and receive partial reimbursements on the mile. The intent of this product is to fulfill the need for an inexpensive mileage tracker system which does not require a subscription service to utilize. Most competitive trackers on the market require a subscription service to access the data, on top of a sometimes expensive hardware purchase [1]. This product grants the user free access to their tracked mileage information via a one-time, inexpensive hardware purchase.

This product can provide small business owners and contractors greater freedom and flexibility within their operations and lessen stress on tax processing.

1.4 Timeline/Proposed Timeline

TABLE III
PROJECT TIMELINE

TASK TITLE	START WEEK	DUE WEEK	DURATION	PCT OF TASK COMPLETE	441 Design											442 Build											443 Present										
					Fall Term										Winter Break		Winter Term										SB	Spring Term									
					2	3	4	5	6	7	8	9	10	F			11	12	13	14	15	16	17	18	19	20	F	21	22	23	24	25	26	27	28	29	30
Project Conception and Initiation																																					
Design Conceptualization	2	3	1	80%																																	
Market Research	2	3	1	100%																																	
Budget	3	5	2	90%																																	
Project Requirements	3	5	2	90%																																	
Project Document - Sections 1&2	3	8	5	100%																																	
Design Impact Assessment	5	9	4	70%																																	
Risk Management	5	9	4	60%																																	
Block Definition																																					
Block Diagram Definition	8	9	2	22%																																	
Interface Definition	8	9	2	16%																																	
Block Implementation			0	0%																																	
Block Testing			0	0%																																	
System Integration																																					
TBD			0	0%																																	
System Level Testing			0	0%																																	
Project Presentation																																					
TBD			0	0%																																	

1.5 References and File Links

[1] E. G. Ruiz, "5 best mileage tracker apps for small businesses in 2022," *Fit Small Business*, 03-Oct-2022. [Online]. Available: <https://fitsmallbusiness.com/best-mileage-tracker-app/>.

1.6 Revision Table

10/12/2022	Chiu, Ewing, Lacey - Initial Document Creation
11/4/2022	Added IEEE compliant table labeling, added gap analysis reference
11/14/2022	Made edits to executive summary; added Gantt timeline chart

2 Impact and Risks

2.1 Design Impact Statement

In this section, we will explore potential harms that the Vehicle Mileage Tracker may bring and come up with preventative solutions and mitigating actions.

Public Health, Safety, and Welfare Impacts: Driver Safety

A critical concern that our project posts is potentially being a distraction to the driver. In the research paper “Do in-car devices affect experienced users' driving performance?” [1], the provided data shows that drivers on average glance off the road for around 0.5 seconds during regular driving. On the other hand, drivers tend to glance off the road for 0.962 seconds when they are texting during their drive. And when drivers enter their destination in the navigation system, they tend to glance off the road for 1.337 seconds. Therefore our goal for our design is to not engage our end users for any time longer than 0.5 seconds during the drive.

Cultural and Social Impacts: Factory Conditions

Our design will be utilizing components that are cut and manufactured in semiconductor factories. We must take the potential negative working conditions of these factories into consideration when choosing our components. In China, for example, water fabrication workers often work 12 hour days in near isolation while being paid well below the average pay for workers of similar skill levels in other factories [2]. Due to the high expectations of production, the stress and penalties of failure cause these factories to have high turnover rates, with one worker quoted as saying “three are recruited, and three leave.” As such, our design will focus on using components that are manufactured in countries with a much higher priority on maintaining ethical labor regulations and supporting workers’ rights, with an emphasis toward U.S. based companies.

Environmental Impacts: PCB Manufacturing

Our design will include a PCB. Traditional PCB manufacturing relies on “energy intensive and high-emission processes that involve copper, epoxy resin, glass fiber, and water.” [3] Furthermore, after a product life cycle ends, PCBs become a waste product. They pile up and become an environmental issue. Handling abandoned PCBs properly could help mitigate this issue. One way to properly handle abandoned PCBs is to recycle it. To make a PCB recyclable, we want to make our device easy to fix. If we design the PCB with reusability in mind, we can allow issues to be detected easily and components to be replaced without too much effort.

Economic Impacts: Influence to Market

The second impact is on currently established companies that produce competitor products to our vehicle mileage tracker. Our design will undercut the profits of those companies because our solution is much more economical for the average small business owner. [4]

2.2 Risks

Table IV
Risk Assessment and Action Plans

Risk ID	Risk Description	Risk category	Risk probability	Risk impact	Performance indicator	Action Plan
R1	Incompatible interface	Technical	H	H	Interface Properties are being met for a block	Adjust Interface Properties along the way
R2	Vendor delay	Timeline	H	M	Parts do not arrive at expected time	Work extra hours to meet project deadline
R3	Go beyond budget	Cost	L	L	Price change	Consider low cost options from the beginning
R4	Unavailable Parts	Timeline	M	H	Unable to order parts	Prepare a backup parts list
R5	Erasing Collected Data by accident	Technical	M	H	Corrupted files on SD Card	Store backup files on user's mobile device via app
R6	Unix Clock unsupported in system peripherals after 2038 [5]	Technical	L	H	Error in Timestamp	Get time stamp from user's mobile device via Bluetooth

2.3 References and File Links

2.3.1 References

[1] A. S. Knapper, M. P. Hagenzieker, and K. A. Brookhuis, IATSS Research, tech., 2015. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0386111214000338>

- [2] China Labour, "China's pristine chip processing plants disguise harsh reality of the work," *China Labour Bulletin*, 30-Mar-2021. [Online]. Available: <https://clb.org.hk/content/china%E2%80%99s-pristine-chip-processing-plants-disguise-harsh-reality-work>.
- [3] "Eco-Friendly Printed Circuit Boards: Present and Future Manufacturability," *resources.pcb.cadence.com*. [Online]. Available: <https://resources.pcb.cadence.com/blog/2020-eco-friendly-printed-circuit-boards-present-and-future-manufacturability>
- [4] Driversnote, "IRS & Employer Mileage Log Requirements: See What Records You Need," *Driversnote*, 04-Oct-2022. [Online]. Available: <https://www.driversnote.com/irs-mileage-guide/mileage-log-requirements> [Accessed: 04-Nov-2022].
- [5] S. Gibbs, "Is the year 2038 problem the new Y2K Bug?," *The Guardian*, 17-Dec-2014. [Online]. Available: <https://www.theguardian.com/technology/2014/dec/17/is-the-year-2038-problem-the-new-y2k-bug>. [Accessed: 16-Nov-2022].

2.3.2 File Links

2.4 Revision Table

11/4/2022	Chiu, Ewing, Lacey - Initial Risk Table Creation
11/16/2022	Added needed Unix Clock reference
04/25/2023	Compiled the Design Impact Statement from Design Impact Assessment assignment.

3 Top-Level Architecture

3.1 Block Diagram

Below are the top-level diagrams of the system which demonstrate the inputs and outputs involved. The first in Fig. 1 is a black box diagram which strictly shows the inputs that the system takes in, as well as the outputs that it provides. In Fig. 2 the more detailed block diagram is displayed, demonstrating the internal blocks and how they interact with one another.

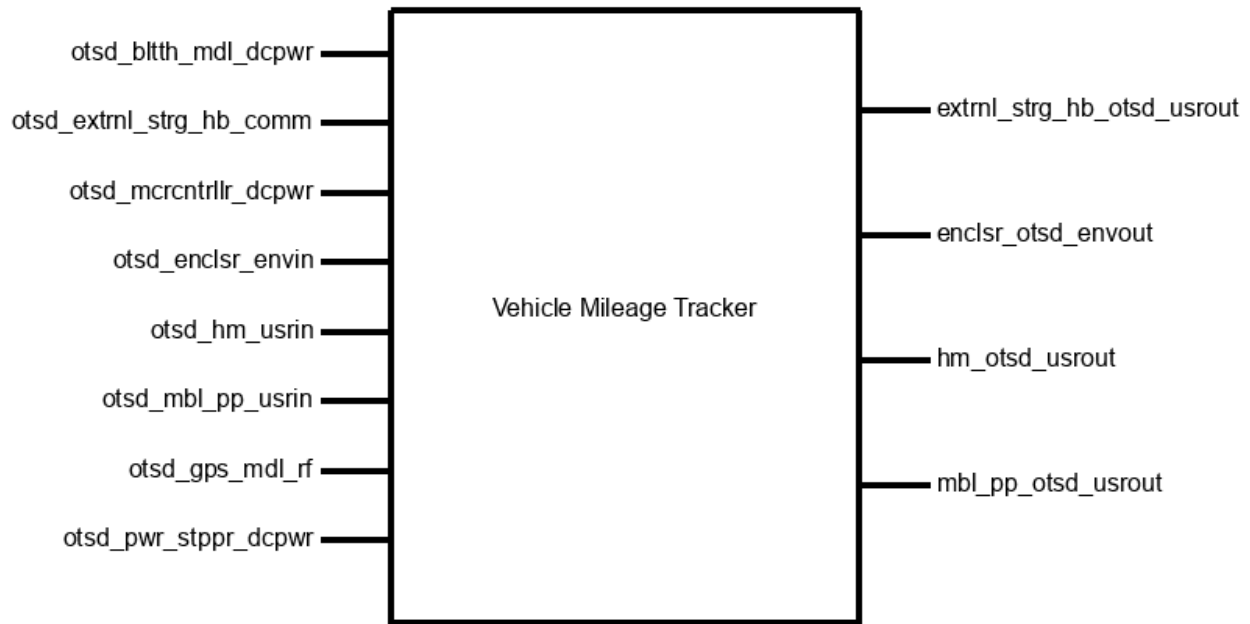


Fig. 1 Black Box Diagram

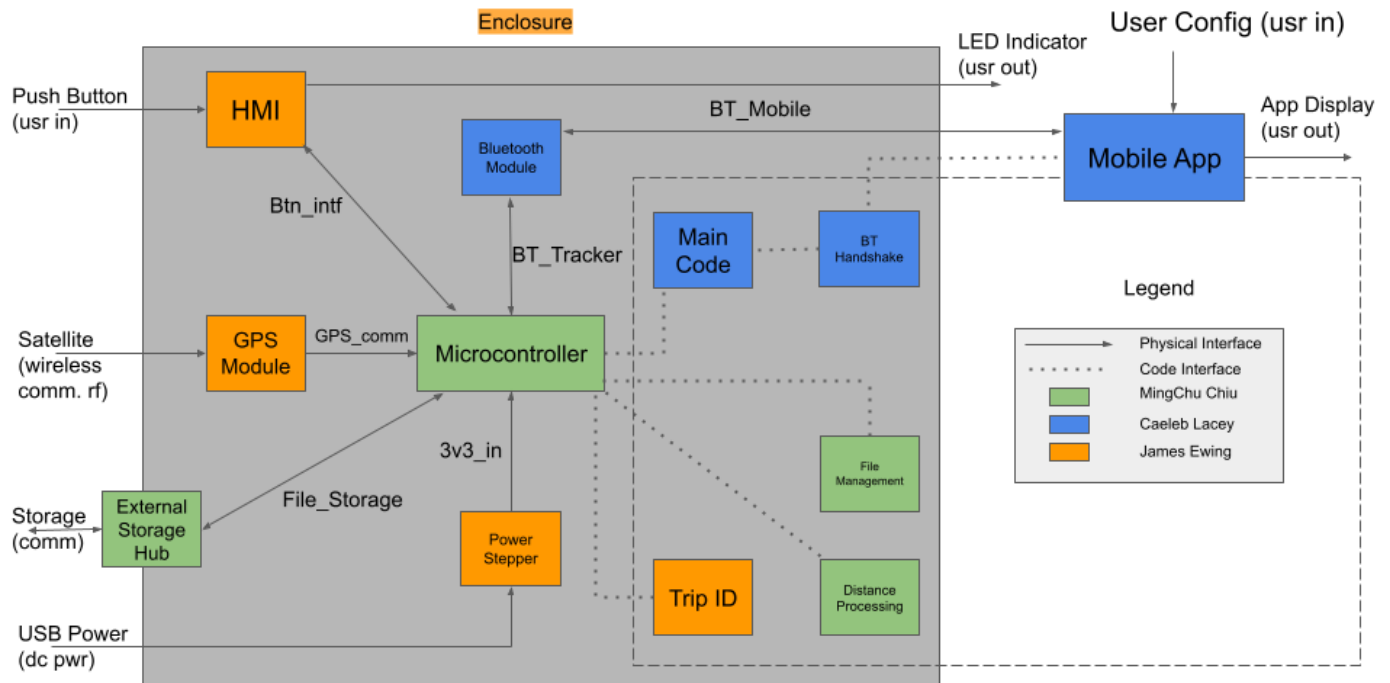


Fig. 2 System Block Diagram

3.2 Block Descriptions

Below is a list of the system blocks, along with a short detailed description of each block and its overall function within the system, as well as the team member who is championing the production of that block.

Name	Description
Bluetooth Module Champion: Caeleb Lacey	The Bluetooth module handles the physical processing of Bluetooth signals which facilitates the connection between our device and a mobile device running our mobile app. It will communicate with the mobile app through an approximately 2.4 GHz signal, and communicate with the microcontroller via some digital signal medium(I2C, USART, etc.).
Bluetooth Handshake Champion: Caeleb Lacey	The Bluetooth handshake is a necessary step in establishing and maintaining Bluetooth connections. This process will sync the device up with a mobile device via our mobile application and maintain a secure connection.

Mobile App Champion: Caeleb Lacey	The Mobile App block is the primary user interface for the mileage tracker system. It is a visual medium that allows users to easily interact with their data, as well as configure settings for the tracker device. The application includes several features that make use of this system simple for end users, and plays an integral role into several of the engineering requirements for the system: • Bluetooth Configurable: The mobile application allows smooth transfer of data for read/write purposes over a paired Bluetooth connection between the user's mobile device and the mileage tracker device. • Driver Safety: The application locks itself during a trip to prevent the user from being distracted by the application while actively driving. • End User Documentation: The application includes a guided walkthrough for the user upon setting up the device, with the option for the user to always return to the walkthrough at any point to refresh their knowledge. This documentation will provide the user with a full use guide for the system and allow modification of system settings. • Neatly Presented Data: The application will neatly present tracked data to the user through well-organized and easy to read tables.
File Management Champion: MingChu Chiu	The File Management block of our code will handle the creation, deletion, and filling of data for the CSV files meant to store trip data. It will handle the creation of a new folder on the first trip of a new year, creation of a file on the first trip of a new month, and will fill the necessary trip data at the beginning and end of each trip. As well, this code block will handle any edits made by the user via the mobile app as the signal are communicated to the microcontroller via Bluetooth.
External Storage Hub Champion: MingChu Chiu	The block is the physical local storage block. It is where an SD card would be inserted, and it is where the main code will store data to. When trip data is accessed through the mobile app, the main code will read the stored data from the SD card. Another way to access the stored data is to physically pull out the SD card and read it on a PC.
Microcontroller Champion: MingChu Chiu	The Microcontroller block is our central processing unit. Hardware modules are accessed through the I/O pins of the microcontroller. It will process data and output it to the local storage to save. It will also provide the output pins for user interface such as an LED. It will host our main code that interacts with the mobile app as well.
Distance Processing Champion: MingChu Chiu	This code block calculates the distance from collected GPS data. And then it will pass it back to the main code on the microcontroller to go through the process for storage.

Enclosure Champion: James Ewing	This block takes care of the outside mechanical enclosure that packages all of the electronics.
HMI Champion: James Ewing	Push button with status LED (indicating whether there is an error, the trip is being recorded, or the trip is not being recorded. The push button/LED combo will be mounted on the dash for ease of interaction.
GPS Module Champion: James Ewing	GPS chip that will provide the location of the device to the microcontroller for purposes of tracking the mileage accumulated on a specified trip along with providing the route that was taken during the trip.
Power Stepper Champion: James Ewing	Power conversion circuitry to supply our logic circuits with the appropriate voltage. This block steps a 5V DC input down to DC 3.3V. The 5V DC input is from a USB car charger as typically seen plugged into a car's cigarette lighter. The specific logic circuits being supplied by the 3.3V DC output of the power stepper are the SD card, GPS module, and Bluetooth module. The purpose of this block is specifically to supply enough current for the three logic circuits. The daughtered microcontroller used for our system is capable of supplying 3.3V at 100mA peak. This is short of the 350mA peak current draw calculated for our system which created the need for a specific power conversion circuit.
Trip ID Champion: James Ewing	Code to identify whether the trip is a business trip or a non-business trip. It will poll the HMI push button for trip identity changes.
Main Code Champion: Caeleb Lacey	This block contains the code that will interact with the user as well as the microcontroller of our system.

3.3 Interface Definitions

Below are the interfaces which define the interactions between blocks. Detailed are the properties for each interface to satisfy sufficient operation.

Name	Properties
otsd_extrnl_strg_hb_comm	<ul style="list-style-type: none"> • Other: Size: 16GB • Other: File System: FAT32 • Other: Type: Micro SD

otsd_mrcntrllr_dcpwr	<ul style="list-style-type: none"> • Inominal: 100 mA • Ipeak: 300 mA • Vnominal: 3.3 V
otsd_enclsr_envin	<ul style="list-style-type: none"> • Other: USB Connector • Other: Main PCB • Other: HMI Switch
otsd_hm_usrin	<ul style="list-style-type: none"> • Other: When the button is actuated, it triggers only once (debouncing check). • Other: Button can be actuated 10 times and 9 of the 10 times the signal has to be received. . • Timing: GPIO logic pin on microcontroller changed on button release.
otsd_mbl_pp_usrin	<ul style="list-style-type: none"> • Timing: Odometer entries and document access will take less than 10 seconds in either direction. • Type: Interaction with mobile touch screen. • Usability: 9 out of 10 users find app intuitive and simple to navigate.
otsd_gps_mdl_rf	<ul style="list-style-type: none"> • Other: Receive Positional data • Other: Operating frequency of 1575MHz on the L1 band. • Protocol: GPS Network Operation
otsd_pwr_stppr_dcpwr	<ul style="list-style-type: none"> • Inominal: 115mA • Ipeak: 400mA • Vmax: 5.5V • Vmin: 2.7V
bltth_mdl_mrcntrllr_dsig	<ul style="list-style-type: none"> • Logic-Level: Active High • Vnominal: 3.3V

bltth_mdI_mbl_pp_rf	<ul style="list-style-type: none"> • Messages: Odometer Entries: Integer value • Messages: Mileage Documents: CSV files • Other: Bidirectional Connection • Protocol: Bluetooth
bltth_hndshk_mn_cd_data	<ul style="list-style-type: none"> • Messages: Bluetooth Key Pairing • Other: Bidirectional Connection • Protocol: Bluetooth - Server
extrnl_strg_hb_otsd_usrout	<ul style="list-style-type: none"> • Type: IRS relevant data fields: Date, Odometer Readings (beginning and ending), Trip Mileage, with optional data fields such as Destination, Business Purpose • Type: Chronological Organization for Trip Records • Type: CSV data
mcrctrllr_bltth_mdI_dsig	<ul style="list-style-type: none"> • Logic-Level: Active High • Vnominal: 3.3V
mcrctrllr_fl_mngmnt_data	<ul style="list-style-type: none"> • Other: Write to a file • Other: Create a file in directory • Other: Read from a file • Other: This is a bidirectional interface
mcrctrllr_extrnl_strg_hb_comm	<ul style="list-style-type: none"> • Datarate: Baud Rate: 40M • Other: Bidirectional Interface • Protocol: SPI
mcrctrllr_dstnc_prssng_data	<ul style="list-style-type: none"> • Messages: a newly collected GPS ping • Messages: ISR_2sec(): the timer overflow Interrupt Service Routine • Other: Frequency: every 2 seconds (0.5 Hz)

mrcntrlr_hm_asig	<ul style="list-style-type: none"> • Other: Red LED - Error Report • Other: Blue LED - Personal Mode • Other: Green LED - Business Mode • Vmax: 3.3 V
dstnc_prcssng_mrcntrlr_data	<ul style="list-style-type: none"> • Messages: Calculated distance between 2 Pings • Other: calcTrip(): Accepts a list of distances and returns a total distance traveled • Other: Update estimated odometer reading after trip ended • Other: This interface is bidirectional • Other: Add a trip record into CSV
enclsr_otsd_envout	<ul style="list-style-type: none"> • Other: Width less than 80mm • Other: Height less than 40mm • Other: Length less than 108mm
hm_otsd_usrout	<ul style="list-style-type: none"> • Other: Blue LED is visible from for Personal Trip • Other: Green LED is visible for Business Trip • Other: Light is visible from 3 feet away.
hm_mrcntrlr_asig	<ul style="list-style-type: none"> • Other: Push button connected to the internal pull-up resistor of Rpi (active low signal) • Vmax: 3.3
mbl_pp_otsd_usrout	<ul style="list-style-type: none"> • Type: Numerical data recorded by the tracker device. • Type: Table of recorded information on document accessed. • Usability: 9 out of 10 users find information well-organized and easy to read.

mbl_pp_blth_hndshk_data	<ul style="list-style-type: none"> • Messages: Bluetooth Key Pairing • Other: Bidirectional Connection • Protocol: Bluetooth - Client
gps_md1_mcrctrllr_comm	<ul style="list-style-type: none"> • Datarate: 19200 • Protocol: UART • Protocol: SPI
pwr_stppr_blth_md1_dcpwr	<ul style="list-style-type: none"> • Inominal: 50mA • Ipeak: 100mA • Vnominal: 3.3V
pwr_stppr_fl_mngmnt_dcpwr	<ul style="list-style-type: none"> • Inominal: 15mA • Ipeak: 170mA • Vnominal: 3.3V
pwr_stppr_gps_md1_dcpwr	<ul style="list-style-type: none"> • Inominal: 50mA • Ipeak: 100mA • Vnominal: 3.3V
trp_d_mcrctrllr_data	<ul style="list-style-type: none"> • Other: The current mode of operation recorded in the microcontroller should match user input • Other: Bi-directional Interface

3.4 References and File Links

3.4.1 References

3.4.2 File Links

3.5 Revision Table

3/12/2023	Imported Our Content from Capstone Student Portal -Chiu
4/25/2023	Filled in interface properties for some GPS interfaces -Team

4 Block Validations

4.1 Bluetooth Module

4.1.1 Description

The Bluetooth module handles the physical processing of Bluetooth signals which facilitates the connection between our device and a mobile device running our mobile app. It will communicate with the mobile app through an approximately 2.4 GHz signal, and communicate with the microcontroller via a UART serial connection. The Bluetooth Module block is integral to the first of our engineering requirements:

- **Bluetooth Configurable:** The Bluetooth Module block handles the processing of the data that is passed between the main device and the mobile application.

Below, further information can be found in regards to the design of the block, a deeper dive into the features of the application, the interfaces with which the block interacts with the overall system, and the steps for verifying the function of the block as intended.

4.1.2 Design

The design of the Bluetooth Module block consists of physical components. The block is interfaced with the power stepper as an input of power, the mobile app along the bidirectional Bluetooth connection, and the microcontroller via a bidirectional UART serial connection. Below, the black box diagram of the block and the circuit level design of the block can be found.

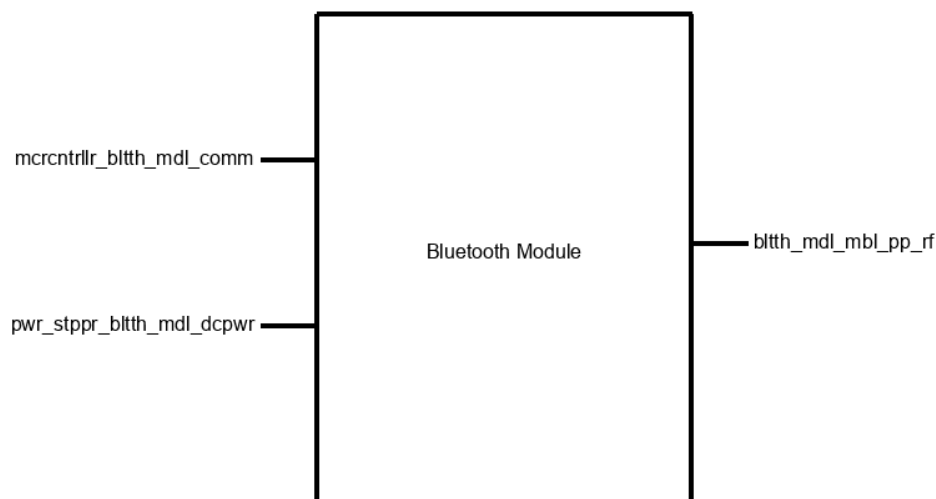


Figure 4.1.2.1: Bluetooth Module Block Black Box

4.1.3 General Validation

The Bluetooth Module block handles the communication between the main device and the mobile application. The system includes a Microchip BM78 Bluetooth module which

can be configured by the microcontroller via AT commands on the UART communication line.

The device's name, UUID, baud rate, and other settings are all set in a one-time setup loop enacted by the microcontroller. Once the settings are established, then the device is set to broadcast as a Bluetooth Low-Energy (BLE) peripheral. Upon a connection request from the mobile device, the connection is established and the module begins listening.

Upon receiving information from the microcontroller, the device will communicate to the mobile application that data is coming, and then will send the data upon receiving the acknowledgement from the app.

When the mobile application is attempting to send data to the main device, the bluetooth module will communicate an acknowledgement and then wait for the data. Upon data being received, it will be communicated to the microcontroller along the UART line for further processing.

4.1.4 Interface Validation

The Bluetooth Module block is connected to the Power Stepper block as its source of power, the Microcontroller block over UART for processing of data, and the Mobile App block over Bluetooth Low-Energy for communication of data. Below is a table of the interfaces and their properties, the purpose of these properties, and how the operation of the Bluetooth module meets these properties.

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?
--------------------	-----------------------------------	--

bltth_mdI_mbl_pp_rf : Output

Messages: Odometer Entries: Integer value	The user needs to be able to communicate odometer values to the tracker device from the mobile app for data recording.	Bluetooth connections can easily carry a value as small as an integer in a single swift message.
Messages: Mileage Documents: CSV files	The user needs to be able to access the recorded documents stored on the tracker device through the	Bluetooth can be utilized to transmit large scale files such as HD videos, any CSV file that is only

	mobile app and submit changes to the documents back to the device.	recording data for a year will never be able to come close to that kind of file size, and therefore the connection will do more than well enough to quickly move documents between the tracker device and mobile device.
Other: Bidirectional Connection	Bluetooth connection by its nature is a bidirectional connection. The module needs to be able to both send information to and receive information from the mobile app.	Information Only
Protocol: Bluetooth	The tracker device needs to be able to communicate with the mobile app.	We chose Bluetooth over Wifi as a quick communication method with a mobile device as a Wifi connection can overcome any internet connection on some devices. Bluetooth allows for a user to have multiple devices connected and still maintain a vital connection to the internet for other purposes such as navigation.

mcrntrlr_bltth_mdl_comm : Input

Datarate: Baud Rate: 115,200	The baud rate needs to be in a band which allows the Bluetooth module to clear out its cache as quickly as possible so as to not cause data flow backups.	The BM78 module can be configured for any baud rate up to 2 Mbps. 115,200 is a commonly used baud rate and allows the device to quickly clear its onboard cache in either direction of communication.
Messages: Mileage Documents: CSV files	The user needs to be able to access the recorded	The BM78 model we are utilizing has 320 KB of

	documents stored on the tracker device on the mobile app and submit changes to the documents back to the device.	onboard ROM which acts as a data buffer[1]. Considering the size of a single character value is 1 byte (B), and our output format averages around 50 characters per line. This would take a CSV file that is over 6,500 trips long to overload at a transfer rate of just 1 byte per second. As we are transmitting data both through the UART and over Bluetooth at far faster rates than that, then our choice for this module fits well within our needed parameters.
Messages: Odometer Entries: Integer value	The user needs to be able to communicate odometer values to the tracker device from the mobile app for data recording.	Similar to the CSV file explanation above, Integer values are only 8 bytes long, which transfers incredibly fast over the Bluetooth and UART connections.
Other: Bidirectional Connection	UART by nature is a bidirectional connection, with a transmitter (TX) and receiver (RX) line on either end of the connection. The TX line of one device connects to the RX line of the other, and vice versa.	Information Only
Protocol: UART Serial	The Bluetooth module requires some avenue of communicating data directly with the microcontroller.	The BM78 Bluetooth module utilizes UART to communicate with a microcontroller. This is a pretty standard route for serial communication with Bluetooth modules across the industry.

pwr_stppr_blth_mdI_dcpwr : Input

Inominal: 50mA	This nominal current was chosen based on the expected current needs of the system overall.	For the BM78 bluetooth module: <ul style="list-style-type: none"> • 43mA for continuous TX condition, rounded up to 50mA. [1]
Ipeak: 100mA	In this example, we don't expect the current draw of the whole system to ever spike above this number. The value was selected by adding up the maximum current of all parts and multiplying by 2	For the BM78 bluetooth module: <ul style="list-style-type: none"> • Peak current draw 100mA from on board LDO. [1]
Vnominal: 3.3V	In this example, this property was chosen based on the design we plan to use in the power supply block.	For the BM78 bluetooth module: <ul style="list-style-type: none"> • Nominal operating voltage listed as 3.3V. [1]

4.1.5 Verification Process

Below is the verification plan for confirming the proper operation of the Bluetooth Module block.

1. Apply power to the tracking device. The onboard LED connected to the Bluetooth device should confirm that it is operational and awaiting a connection.
2. If device connection has not yet been established, connect the tracker device to the mobile application via the verification steps under the Bluetooth Handshake block (Section 4.2.5 below).
3. Test the communication by initiating a trip on the mobile application using your vehicle's odometer readings, and then go on a short trip.
4. Upon ending the trip, access the tracking device's onboard storage via the mobile application and confirm the successful recording of your trip.
5. If the trip file is able to be opened on the mobile device, then this confirms the successful communication of a CSV file over the Bluetooth connection.
6. If the most recent trip correctly includes the odometer values you utilized on the test trip, then this confirms the correct communication of integer values over the Bluetooth connection.

That concludes the operation verification of the Bluetooth Module block.

4.1.6 References and File Links

[1] Microchip technology,
<https://ww1.microchip.com/downloads/en/DeviceDoc/BM78-Data-Sheet-DS60001380D.pdf> (accessed May 15, 2023).

4.1.7 Revision Table

Date	Revision
5/14/23	Caeleb Lacey: Initial Block Validation Submitted

4.2 Bluetooth Handshake

4.2.1 Description

The Bluetooth Handshake block is a necessary step in establishing and maintaining Bluetooth connections. This process will sync the device up with a mobile device via our mobile application and maintain a secure connection.

4.2.2 Design

The Bluetooth Handshake block is a slightly automated code process which acts as a “hello, my name is” protocol between the tracker device and the mobile application. Handled over Bluetooth Low-Energy, the process is quite standardized according to the protocols of BLE. Below is a simple diagram illustrating the interfaces with other code blocks.

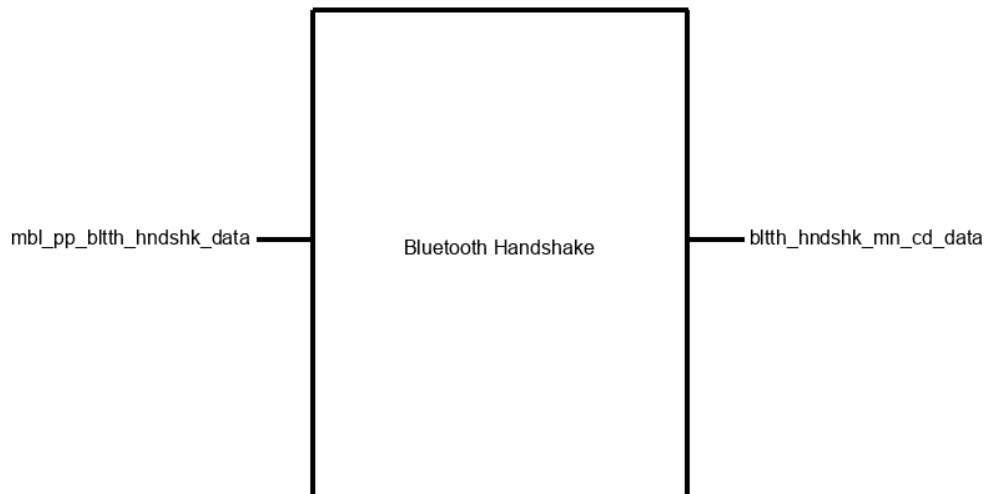


Figure 4.2.2.1: Bluetooth Handshake Block Black Box

4.2.3 General Validation

The Bluetooth Handshake block is a simple, but necessary portion of our code that allows the connection between the tracker device and a mobile device through our mobile app to be established.

Through its own code, the mobile application broadcasts the mobile device as a BLE central device, having it act as the command initiating device. The Bluetooth module on the tracker device broadcasts itself instead as a peripheral device. This allows the mobile app to seek out and “see” the tracker device, and then initiate the connection.

The handshake code is setup in such a way that when the mobile app specifically initiates a connection request, the tracker device will accept the pairing and share its pairing key. This then allows for a “remembered” connection, which should minimize the number of times a user potentially has to engage the connection again.

4.2.4 Interface Validation

The Bluetooth Handshake block is a portion of the main code on the microcontroller as well as the code on the mobile application. Below is a table of the two primary interfaces and their properties, the purpose of these properties, and how the operation of the Bluetooth handshake protocol meets these properties.

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?
--------------------	-----------------------------------	--

bltth_hndshk_mn_cd_data : Output

Messages: Bluetooth Key Pairing	Bluetooth, be it Classic or Low-Energy, requires an agreed upon sharing of access keys by both devices involved.	Upon receiving a connection request from the mobile app, the Bluetooth handshake code shares the tracking device's key with the mobile device, prompting the mobile device to share its back and establish the connection.
Other: Bidirectional Connection	Bluetooth connection by its nature is a bidirectional connection. The module needs to be able to both send information to and	Information Only

	receive information from the mobile app.	
Protocol: Bluetooth - Peripheral	The tracking device needs to advertise as a peripheral. This is due to not having a GUI on the tracker device itself, meaning there would be no way to initiate a connection to a specific mobile device from it if it were acting as central.	Our code establishes the tracking device as the peripheral device, which allows it to be “seen” by the mobile app where a user can initiate the connection protocol manually.

mbl_pp_bltth_hndshk_data : Input

Messages: Bluetooth Key Pairing	Bluetooth, be it Classic or Low-Energy, requires an agreed upon sharing of access keys by both devices involved.	Upon receiving a connection request from the mobile app, the Bluetooth handshake code shares the tracking device’s key with the mobile device, prompting the mobile device to share its back and establish the connection.
Other: Bidirectional Connection	Bluetooth connection by its nature is a bidirectional connection. The module needs to be able to both send information to and receive information from the mobile app.	Information Only
Protocol: Bluetooth - Central	The mobile application needs to be set as a central manager. This allows us to provide the user with an interface that allows them to select a peripheral device, in this case our tracking device, to connect with.	Our code utilizes the mobile device as a central device through Apple’s CoreBluetooth protocols, and allows the user to “see” advertising peripheral devices and select our tracking device to establish a connection with.

4.2.5 Verification Process

The Bluetooth Handshake block has a fairly simple verification process which is outlined below.

1. Power on the mileage tracker device and open the mobile mileage tracker application.
2. Within the application, confirm that the mileage tracker device has been successfully connected to automatically. If this has not occurred, initiate a search by pressing the button that indicates so on the main menu.
3. The device should automatically connect to the application, and at this point the handshake has commenced and been acknowledged, and therefore the process is complete.

This concludes the verification process for the Bluetooth Handshake block.

4.2.6 References and File Links

4.2.7 Revision Table

Date	Revision
5/14/23	Caeleb Lacey: Initial Block Validation Submitted

4.3 Mobile App

4.3.1 Description

The Mobile App block is the primary user interface for the mileage tracker system. It is a visual medium that allows users to easily interact with their data, as well as configure settings for the tracker device. The application includes several features that make use of this system simple for end users, and plays an integral role into several of the engineering requirements for the system:

- Bluetooth Configurable: The mobile application allows smooth transfer of data for read/write purposes over a paired Bluetooth connection between the user's mobile device and the mileage tracker device.
- Driver Safety: The application locks itself during a trip to prevent the user from being distracted by the application while actively driving.
- End User Documentation: The application includes a guided walkthrough for the user upon setting up the device, with the option for the user to always return to the walkthrough at any point to refresh their knowledge. This documentation will provide the user with a full use guide for the system and allow modification of system settings.

- Below, further information can be found in regards to the design of the block, a deeper dive into the features of the application, the interfaces with which the block interacts with the overall system, and the steps for verifying the function of the block as intended.

The design of the Mobile App block is primarily code-based. The block is interfaced by both user input and output, as well as a hardware Bluetooth connection which supports the software transfer of data. Below, the black box diagram of the block and the operation workflow of the application's code can be found.



Figure 4.3.2.1 above is a black box diagram of the Mobile App block and its connected interfaces.

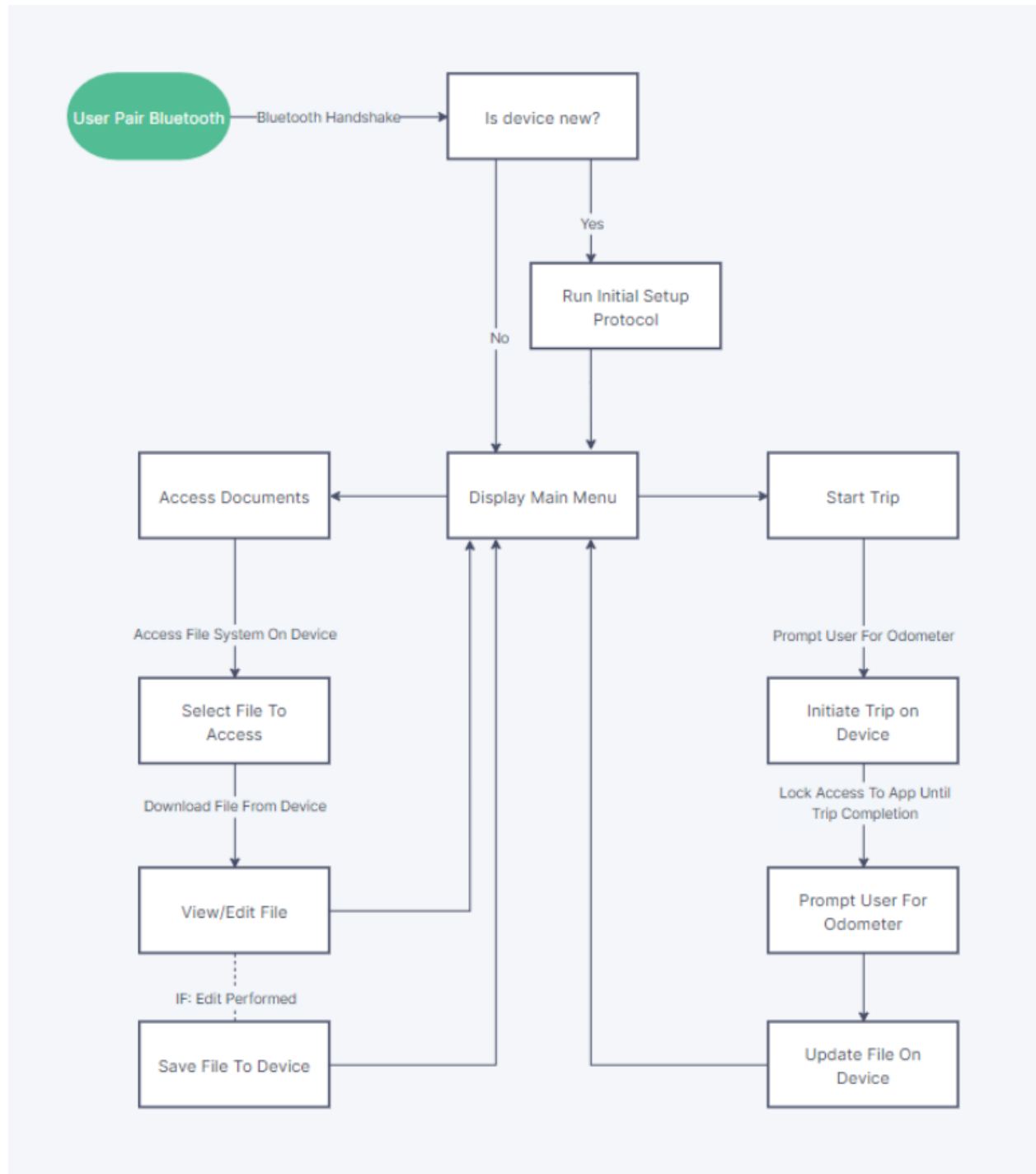


Figure 4.3.2.2: Mobile App Operation Flowchart

Figure 4.3.2.2 above is a demonstration of the workflow of the application as the user interacts with its features.

4.3.3 General Validation

The Mobile App block represents the main code through which the user will interact with the system. Upon initialization of the application, a paired Bluetooth connection is sought. If the tracker device has already been paired with the mobile device utilizing the application, then the application proceeds to the main menu. If no device is paired, then a setup guide is instead initiated.

In the setup guide, the user is walked through establishing the Bluetooth pairing, deciding upon settings for operation of the system, and guided on a walkthrough of how to utilize the system and its features. Upon completing this setup, the Bluetooth pairing between the tracker device and the mobile device has been established and the user may proceed to utilizing the system.

On the main menu, the user is provided with three options: start a trip, document access, and settings. Under the settings, the user may reconfigure any settings they had decided upon in the initial setup, or review the guided walkthrough for how to properly use the system.

If the user chooses to start a trip, then they are prompted to input the current odometer reading. The user is then prompted to lock their phone to prevent distracted driving. If the tracker device is found to be in motion, the application will lock itself and prevent the user from interacting with it while driving. Upon completion of a trip and coming to a complete stop, the application will open up once more and allow the user to end the trip. Upon ending the trip, the user is once more prompted to enter the current odometer readout, and both the starting and final value are sent to the tracker device for storage and the user is returned to the application's main menu.

If the user chooses to access documents, then the application communicates with the tracker device for file information, presenting the user with the file system. The user will first be presented with file directories by year, and within each directory, CSV files will be arranged by month. When the user selects a month, the CSV file is then transferred to the mobile application and parsed into an easily readable table. At this table, the user can choose to simply view the information to find the values they are looking for, or they can choose to edit entries if they find there to be information that needs to be corrected. If an edit is performed, the file is saved and then sent back to the tracker device in order to update the record when the user is done, and the user is returned to the main menu.

Ultimately, the application is designed in this way to grant the user ease of access to their data and fulfill the desires of our project partner to have the data easily accessed, read, and modified. An alternate design to this access was considered, and it is also being implemented into the system. The alternate design is in case that the user does not have a compatible mobile device, they can simply remove the SD storage card connected to the device and access the information on any computer that can read it.

4.3.4 Interface Validation

The Mobile App block communicates primarily with the Bluetooth Module block on a hardware scale, and through the Bluetooth Handshake block as a communication go-between with the main code of the tracker device. Below is a table of the interfaces and their properties, the purpose of these properties, and how the operation of the application meets these properties.

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?
otsd_mbl_pp_usrin : Input		
Timing: Odometer entries and document access will take less than 10 seconds in either direction.	Data needs to be accessed or communicated between the mobile device and the tracker device swiftly for the sake of ease of use.	This property on average will be met purely by the transfer rate of Bluetooth, which averages around 1 Mbps[1]. Seeing as the only data we will be transferring back and forth from the mobile device to the tracker device are integer values and CSV files, the messages will be transferred nearly in an instant.
Type: Interaction with mobile touch screen.	The user needs to interface with the application through their mobile device's touch screen.	The application is being developed for iOS, the operating system of iPhones. iPhones utilize a touch screen for interaction with applications.
Usability: 9 out of 10 users find app intuitive and simple to navigate.	Our users need to find the system, as a whole, simple and easy to use.	The guided setup portion of the application walks the user through how to properly use both the tracker device and the mobile application itself.

bltth_mdI_mbl_pp_rf : Input

Messages: Odometer Entries: Integer value	The user needs to be able to communicate odometer values to the tracker device for data recording.	At the start of a trip, the user will be prompted to input the starting odometer reading. At the end of a trip, the user will once more be prompted to input the ending odometer reading for the trip. Upon entering the final reading, both values will be sent to the tracker device for recording.
Messages: Mileage Documents: CSV files	The user needs to be able to access the recorded documents stored on the tracker device.	The user will be able to access documents stored on the tracking device via the Bluetooth connection. Upon selecting a specific file, the entire CSV will be sent to the application for further navigation.
Other: Bidirectional Connection	Bluetooth connection by its nature is a bidirectional connection. The application needs to be able to both send information to and receive information from the tracker device.	Information Only
Protocol: Bluetooth	The mobile application needs to be able to communicate with the tracker device.	Bluetooth offers a simple connection between our two-device system.

mbl_pp_otsd_usrout : Output

Type: Table of recorded information on document accessed.	Information stored in the CSV file needs to be presented to the user in an understandable fashion.	CSV files by their design offer an easy method of aligning data into a table.
Type: Numerical data recorded by the tracker device.	The data populating the CSV needs to be accurate and representative of the recorded data needed for	The numerical values stored in the CSV file will be presented to the user in an easy to read fashion

	IRS reporting.	which makes the numbers clear as to their value and purpose.
Usability: 9 out of 10 users find information well-organized and easy to read.	Our users need to find the system, as a whole, simple and easy to use.	The table will be arranged in a simple to understand format. The first column will denote the date and time of the start of the trip, with entries descending in chronological order. Columns at the top of the table will remain at the top of the screen so that even if a user has to scroll far for the information they're seeking, they'll still be able to quickly reference what each column's data means.

mbl_pp_bltth_hndshk_data : Output

Messages: Bluetooth Key Pairing	The application needs to establish a connection with the tracker device for quick and easy communication of data.	If the mobile device is not paired with the tracker device already, the initial setup portion of the application will guide the user through establishing the connection. If the mobile device is already paired with the tracker device, then opening the application will simply include a quick handshake to confirm that the two are connected and ready to communicate.
Other: Bidirectional Connection	Bluetooth connection by its nature is a bidirectional connection. The application needs to be able to both send information to and receive information from the tracker device.	Information Only

Protocol: Bluetooth	The mobile application needs to be able to communicate with the tracker device.	Bluetooth offers a simple connection between our two-device system.
---------------------	---	---

4.3.5 Verification Process

Below is the verification plan for confirming proper operation of the Mobile App block. This process is laid out in a manner that will allow any end user to confirm that the mobile application is functioning as designed.

1. The user will open the application (otsd_mbl_pp_usrin) on their mobile device while the tracker device is on. If the mobile device and tracker have already been paired, then skip to step 4. If the two are not paired yet, proceed to step 2.
2. The user will be prompted with the initial setup process of the mobile application. First, the setup will guide the user through establishing the Bluetooth connection (mbl_pp_blth_hndshk_data, blth_mdl_mbl_pp_rf). Once the connection is established, they will set the device settings and those settings will be communicated with the tracker device.
3. The user will move through the guided walkthrough on use of the system, one page at a time. Upon completion of the walkthrough guide, the user will be asked if they understand how to utilize the system, and if confirmed, will proceed to step 4.
4. Upon landing on the main menu, the user will initiate a trip. The application will prompt the user to enter the odometer reading.
5. The user will then end the trip, and will once more be prompted to input the final odometer reading. Upon entering these values, both values will be communicated with the tracker device for storage in a new trip entry.
6. The user will now proceed to document access. Upon accessing the file system, the user will access one of the CSV files and confirm that all information is present and clear to understand (mbl_pp_otsd_usrout).
7. The user will select a cell in the table to modify, and upon successful editing of the block, confirm and save the file. Upon saving the file, the file will be sent back to the tracker device for the most up to date recording. The user will then be returned to the main menu.
8. The user will once more select the file to confirm that the edit was properly saved.
9. Finally, the user will return to the main menu and access the settings. They will confirm that their settings for the tracker device can be modified and that the user guide can be accessed.

That concludes the operation verification for the Mobile App block.

4.3.6 References and File Links

[1] "Bluetooth technology overview," Bluetooth® Technology Website, 2022. [Online]. Available: <https://www.bluetooth.com/learn-about-bluetooth/techoverview/>. [Accessed: 20-Jan-2023].

4.3.7 Revision Table

Date	Revision
1/20/23	Caeleb Lacey: Initial Block Validation Draft Submitted
2/11/23	Caeleb Lacey: Revisions applied to every section, updated and clarified information in regards to design and details of operation.

4.4 File Management

4.4.1 Description

4.4.2 Design

4.4.3 General Validation

4.4.4 Interface Validation

4.4.5 Verification Process

4.4.6 References and File Links

4.4.7 Revision Table

4.5 External Storage Hub

4.5.1 Description

4.5.2 Design

4.5.3 General Validation

4.5.4 Interface Validation

4.5.5 Verification Process

4.5.6 References and File Links

4.5.7 Revision Table

4.6 Microcontroller

4.6.1 Description

The Microcontroller block is our central processing unit. Hardware modules are accessed through the I/O pins of the microcontroller. It will process data and output it to the local storage to save. It will also provide the output pins for user interface such as an LED. It will host our main code that interacts with the mobile app as well.

4.6.2 Design

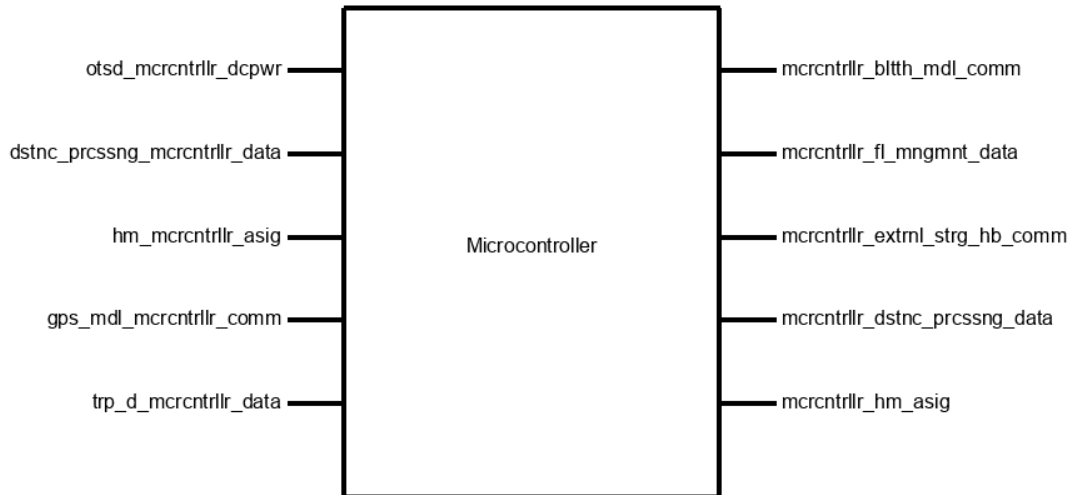


Figure 1: Black Box Diagram of Microcontroller Block

To achieve the engineering requirements, the main code that runs on the microcontroller should:

- Constantly read in GPS messages once powered on (such as a 2-second timer interrupt service routine)
- Constantly check on HMI (push button, as well as mobile app trip_start request) input
- Constantly determine which state the microcontroller should be in, depending on the values above.

Since there are three main things that the microcontroller should constantly check on, I used three different timer interrupts on the Raspberry Pi Pico to accomplish these checks.

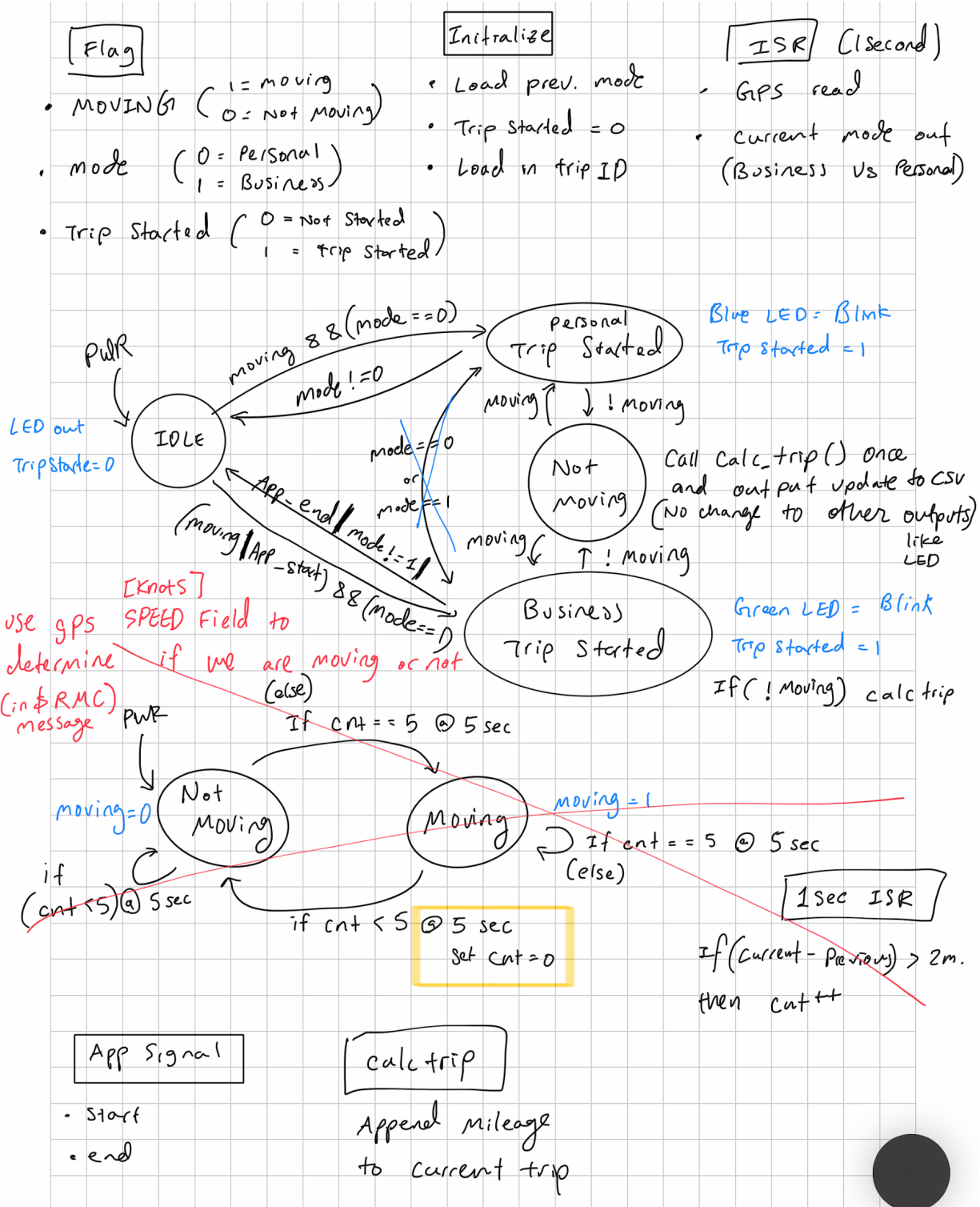


Figure 2: Design of State Machine on Microcontroller

Notes: [name in blackbox] ⇔ [name in state machine diagram]

otsd_mcrntrlr_dcpwr ⇔ PWR, dstnc_prCSSng_mcrntrlr_data ⇔ Calc_trip(),

Hm_mcrntrlr_asig ⇔ mode, gps_mdI_mcrntrlr_comm ⇔ GPSread(),

trp_d_mcrntrlr_data ⇔ moving

4.6.3 General Validation

I used 3 different timer interrupts on the Raspberry Pi Pico, of different frequencies. For checking the GPS messages, I used a 2-second period timer interrupt. This is because

4.6.4 Interface Validation

4.6.5 Verification Process

4.6.6 References and File Links

4.6.7 Revision Table

4.7 Distance Processing

4.7.1 Description

This code block calculates the distance between 2 GPS pings. The main code on the microcontroller will pass a new GPS ping every 2 seconds to this distance processing code block, using a Timer Overflow Interrupt. The calculated distance will then be passed back to the main code on the microcontroller and then stored as Comma Separated Values (CSV) on the external storage hub. When the trip is determined to be at the end, this block will sum up all the calculated distances in this trip, and append this totalled amount of mileage to the current odometer reading. That way, we will get an estimated value of the odometer reading when a trip ends. The ending odometer reading is a relevant information to attach in the report to the Internal Revenue Service (IRS) regarding business trips. Therefore we will save the odometer reading at the beginning and the end of a trip into a separate CSV file for the purpose of creating a presentable set of data.

4.7.2 Design

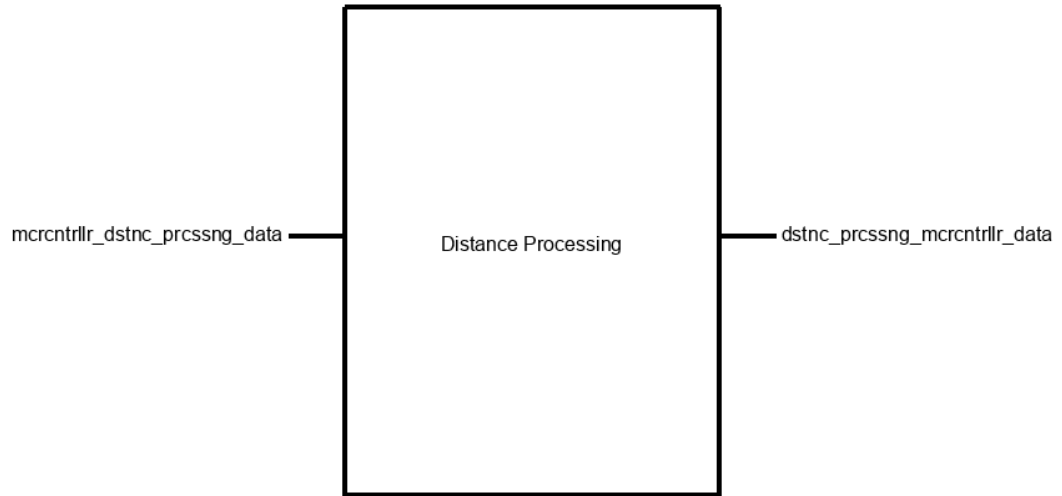


Figure 1: Black Box Diagram of Distance Processing Block

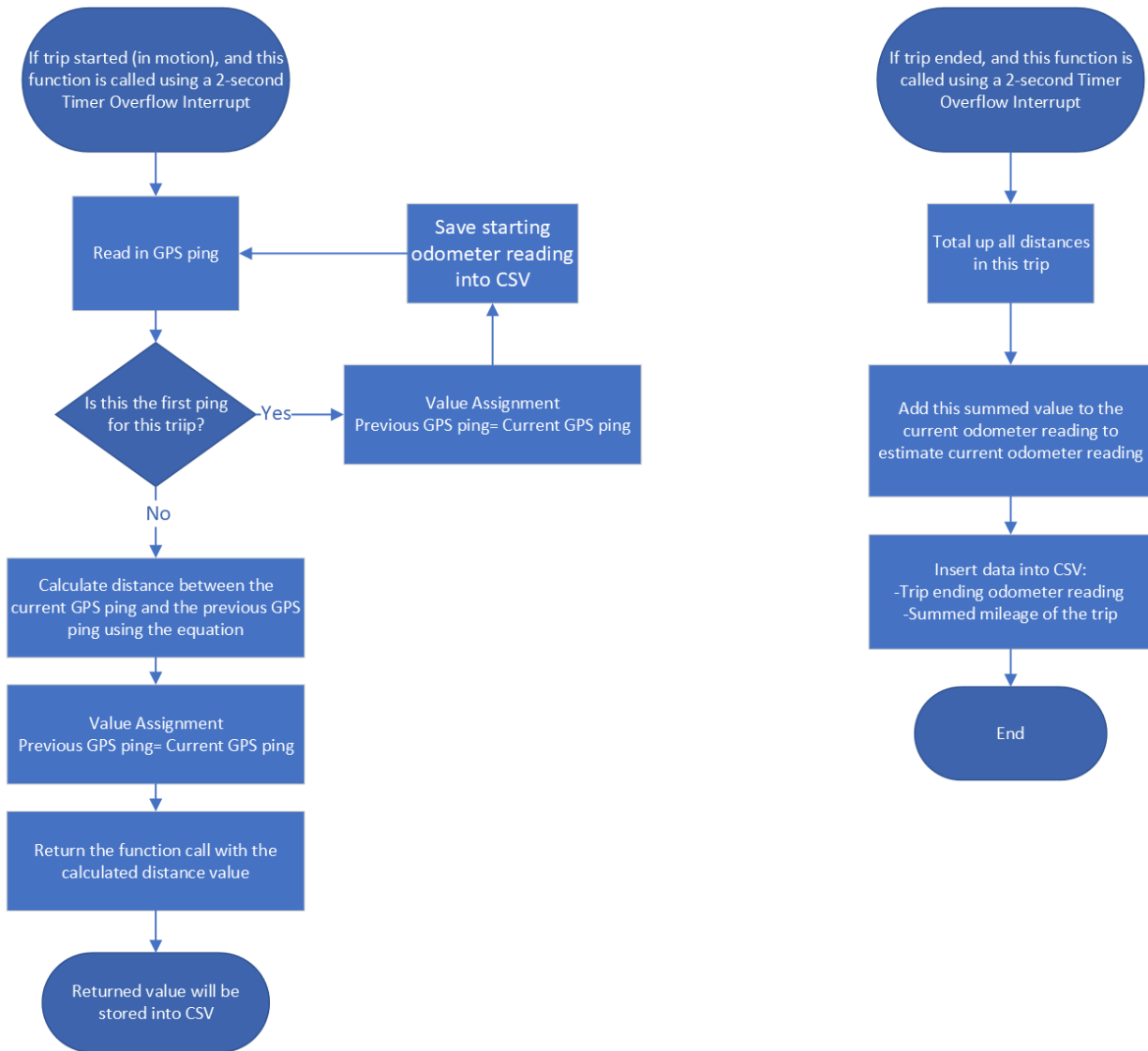
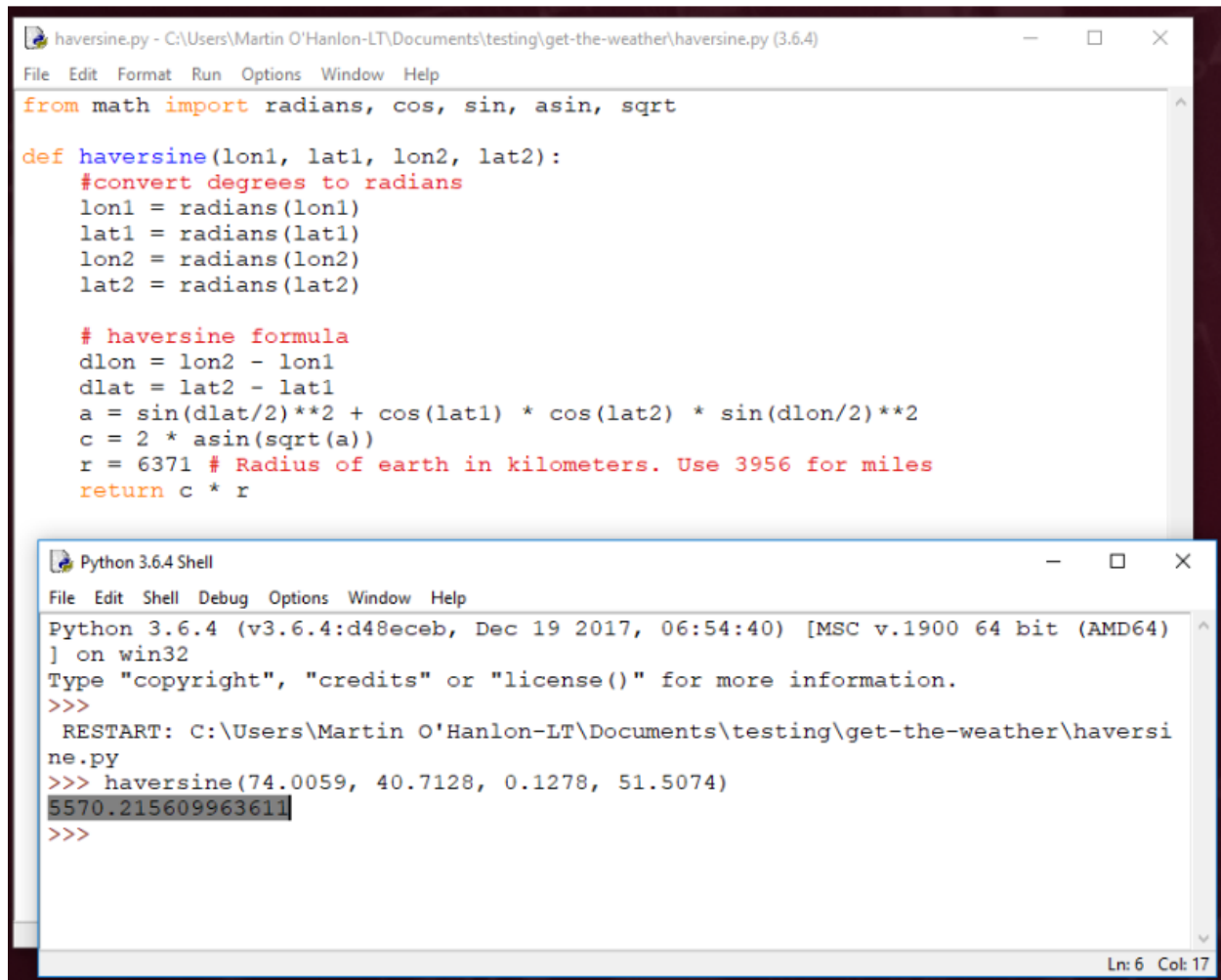


Figure 2: Distance Processing Flow Chart

Notes:

GPS Ping is the input of this block, **mcrctrllr_dstnc_prcssng_data**.

Calculated distance, totalled distance, and estimated odometer reading are the output of this block, **dstnc_prcssng_mcrctrllr_data**.



```
haversine.py - C:\Users\Martin O'Hanlon-LT\Documents\testing\get-the-weather\haversine.py (3.6.4)
File Edit Format Run Options Window Help
from math import radians, cos, sin, asin, sqrt

def haversine(lon1, lat1, lon2, lat2):
    #convert degrees to radians
    lon1 = radians(lon1)
    lat1 = radians(lat1)
    lon2 = radians(lon2)
    lat2 = radians(lat2)

    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 6371 # Radius of earth in kilometers. Use 3956 for miles
    return c * r

Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\Martin O'Hanlon-LT\Documents\testing\get-the-weather\haversi
ne.py
>>> haversine(74.0059, 40.7128, 0.1278, 51.5074)
5570.215609963611
>>>
```

Figure 3: Distance Calculation Formula for GPS pings

4.7.3 General Validation

This is the Distance Processing code block for the Vehicle Mileage Tracking system to process the received GPS pings and turn them into mileages. The idea is to sum up all distances calculated from the pings of the same trip to get our trip mileage.

I chose to sample GPS pings every 2 seconds because that seems to be a reasonable logging distance considering when a vehicle is traveling at 65 mph. $65 \text{ mph} \left(\frac{1609 \text{ m}}{1 \text{ mile}} \right) \left(\frac{1 \text{ hr}}{3600 \text{ s}} \right) = 29.05 \frac{\text{m}}{\text{s}}$ which makes 2 seconds cover a distance of $29.05 \frac{\text{m}}{\text{s}} (2 \text{ s}) \approx 60 \text{ m}$. With 60 meters being under 100 meters, the short distance in my mind, I find it appropriate to log the GPS pings with a 2-second interval. And I chose to use interrupt on the microcontroller because logging GPS coordinates is a time critical event.

The circular flow in the flowchart came from waiting for a second GPS ping to arrive before attempting distance calculation. Notice that there are two exits in this flowchart, with the exit on

the left happening every two seconds, and the exit on the right happening on the exit of the entire trip.

Figure 3 includes the calculation formula that I would apply to the GPS coordinates to obtain the distance. I chose this formula because it seems implementable on a Raspberry Pi Pico.

4.7.4 Interface Validation

Interface Property	Why is this interface of this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?
mcrctrllr_dstnc_prcssng_data : Input		
Messages: Read GPS Ping every 2 seconds	I chose to pull in GPS pings every 2 seconds because during that time interval the distance covered matches my definition of short distance, of it being 100 meters.	I plan to use a timer overflow interrupt every 2 seconds which would pause any procedures at hand and execute the interrupt routine of getting the current GPS coordinate.
dstnc_prcssng_mcrctrllr_data : Output		
Messages: Calculated distance between 2 Pings	Being able to determine the distance between any two GPS pings allows us to calculate individual distances within the trip, which is crucial for calculating the entire trip distance.	I have looked for resources online and found some equations for calculating distances between two GPS coordinates.
Messages: Totalled distance in one trip	The summed distances will be our trip mileage, which we will use to estimate the current odometer reading as well.	This property could be met by summing up all the individual distances of the same trip.
Other: Store individual distances within the trip into CSV	This property is needed because it will log down individual distances, which will be used to calculate trip mileage at the end of the trip.	This property could be met by recording down the values returned by the calculation functions.
Other: Add a trip record into CSV	Trip records will be kept in a different CSV file than the individual distances, to make it	IRS relevant data (starting and ending odometer readings, trip mileage) about this trip will be

	easier to read for the user.	added into a record in CSV.
Other: Update estimated odometer reading after trip ended	This property is needed so that at the end of each trip, we will update the odometer reading by the total trip mileage.	This could be done by appending the total trip mileage to the current odometer reading.

4.7.5 Verification Process

For validation, I will be testing the interfaces to say that the block is functional.

mcrctrllr_dstnc_prssng_data : Input

Messages: Read GPS Ping every 2 seconds	<p>The code will feed in a GPS coordinate every 2 seconds.</p> <ol style="list-style-type: none"> 1. Output the GPS coordinate to the serial monitor before feeding it into the distance calculating function. 2. Look at the timestamp on the serial monitor and determine if the timestamp intervals are 2-second long.
---	---

dstnc_prssng_mcrctrllr_data : Output

Messages: Totalled distance in one trip	<p>This code will sum up all the individual distances between GPS pings of the trip.</p> <ol style="list-style-type: none"> 1. If the output of this code equals all the individual distances in the CSV added together, then this code is validated.
Messages: Calculated distance between 2 Pings	<p>This code will calculate the distance between 2 GPS pings.</p> <ol style="list-style-type: none"> 1. If the output of this code is the same distance as what is calculated on Google Maps, then this code is validated.C
Other: Add a trip record into CSV	<p>This code will add trip records such as total trip mileage, beginning and ending odometer readings to a CSV for the user to read.</p> <ol style="list-style-type: none"> 1. This code is validated if the correct data values are added into the correct columns in the CSV file.
Other: Update estimated odometer reading after trip ended	<p>This code will modify the odometer reading value by the amount traveled in this trip, after the trip ends.</p> <ol style="list-style-type: none"> 1. This code is validated if it modifies the odometer reading by the correct amount when the trip-ending is flagged.

4.7.6 References and File Links

“Finding the distance between two points on the Earth,” *Projects.raspberrypi.org*. [Online]. Available: <https://projects.raspberrypi.org/en/projects/fetching-the-weather/6>. [Accessed: 21-Jan-2023].

4.7.7 Revision Table

1/21/2023	Initial Document Creation -Chiu

4.8 Enclosure

4.8.1 Description

The enclosure for our project is a 3D printed two-part structure designed to house the main PCB and the SD Card PCB. The bottom half of the enclosure features heat set inserts that allow the main PCB to be securely mounted using M3 screws. Furthermore, the bottom half of the enclosure includes holes for both an RF coaxial connector that connects to the GPS module and a micro-USB connector that connects to the Raspberry PI Pico on the main PCB. The top half of the enclosure includes two holes, one for an LED and the other for a toggle switch that together compose of the HMI. The SD Card PCB is connected to the top half using heat set inserts secured with M3 screws. Overall, the enclosure provides a durable and secure housing for the necessary components of the system, while allowing easy access to all of the external connections to the system.

4.8.2 Design

The design process for the enclosure began with a thorough analysis of our project sponsors requirements and the final PCB dimensions. Our team then carefully considered the placement of the components on the PCB and figured out what would be the most ideal location for the connectors and switches that would be externally interfaced with. After a few rough draft sketches with pencil and paper I then began modeling the enclosure with the 3D modeling software Autodesk Inventor 2023 Professional. The initial design was then reviewed and then adjusted for fitment and functionality before being sent to be 3D printed.

During the prototyping phase, my team and I conducted several tests to ensure the enclosure met all of the requirements for the final product, including ease of assembly, durability, and proper ventilation. The heat set inerts used to secure the PCBs in place were chosen for their ease of assembly and for their strong and reliable connection. Our team also tested the

placements of the external connectors and switches to ensure they were accessible and easy to use.

After successfully testing the prototype 3D print, the final design was adjusted for any necessary changes. The necessary changes after the prototype design review was that the micro-USB connector hole was moved up to better match the height of the PCB mounted micro-USB connector, enlarging of the LED and toggle switch holes, and slight repositioning of the mounting holes for the main PCB. The enclosure was then sent for the final 3D print in which we made sure to use a high infill for the print to make the final result as smooth as possible. Summing things up, the design process for the enclosure involved careful consideration of all of the requirements for our project, thorough prototyping and testing, and close attention to detail to ensure a functional and reliable final product.

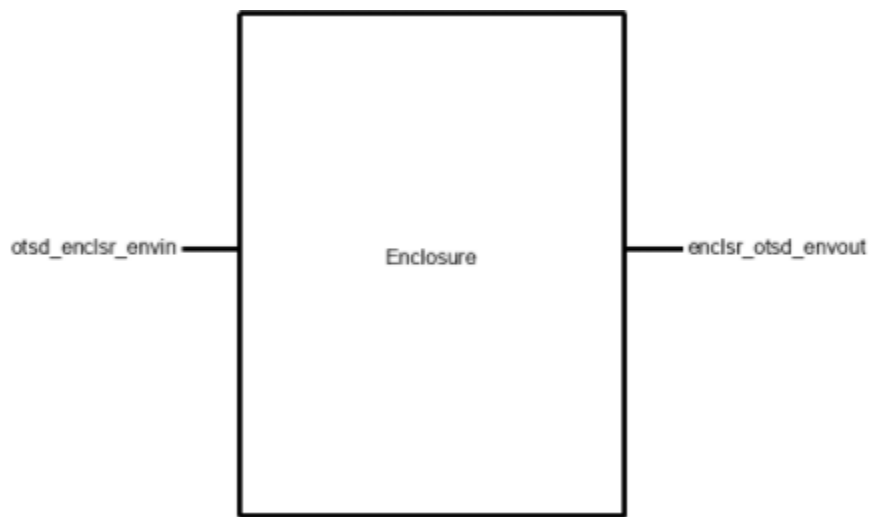


Fig. 1: Figure showing the inputs and outputs environmentally to the enclosure.



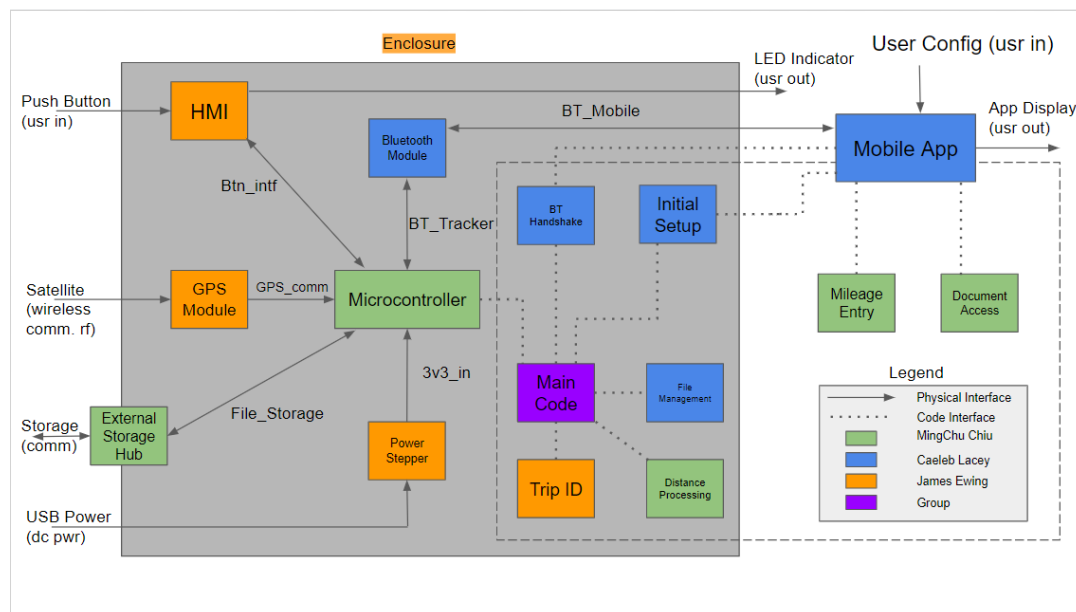


Fig. 2: This is the high level block diagram for the overall system that the Enclosure is apart of.

4.8.3 General Validation

The design of our enclosure was justified based on several factors, one of which was the ability for rapid prototyping. With 3D printing, we were able to iterate quickly and make modifications to the design without having to wait for traditional manufacturing processes. This allowed us to test and refine our design until we were satisfied with the final product.

Another key factor in our decision to use 3D printing was the low cost of filament. This meant that we were able to produce multiple iterations of the enclosure without significantly impacting the overall budget of the project. Additionally, the flexibility of 3D printing allowed us to create a customized design that fit our specific needs.

The enclosure itself consists of two pieces, a bottom piece and a top piece, which are easily taken apart to access the main PCB. The small form factor of the enclosure was dictated by the size of the main PCB and the placement of the connectors, especially the RF connector. Our project sponsor, Stephen Covrig, had requested that the overall package be small enough to fit in the palm of your hand, and we were able to achieve this goal with our design. Overall, the 3D printed enclosure allowed us to create a customized, low-cost, and easily accessible solution for our project needs.

4.8.4 Interface Validation

otsd_enclsr_envin: Input

HMI Switch	Must Exist	
Main PCB	Must Exist	
USB Connector	Must Exist	

enclsr_otsd_envout: Output

Width less than 80mm	Measure width with Calipers to be less than listed value.	
Height less than 40mm	Measure width with Calipers to be less than listed value.	
Length less than 108mm	Measure width with Calipers to be less than listed value.	

4.8.5 Verification Process

- Step 1: Turn on digital calipers and set to inch units by pressing mode button.
- Step 2: Measure length of the enclosure with the digital calipers.
- Step 3: Measure width of the enclosure with the digital calipers.
- Step 4: Measure height of the enclosure with the digital calipers.

4.8.6 References and File Links

[1] "Carr," McMaster, <https://www.mcmaster.com/94180A331/> (accessed May 15, 2023).

4.8.7 Revision Table

Date	Revision
04/23/2023	First revision.
04/30/2023	Added more detailed references for interface definitions.
05/10/2023	Updated document based on feedback.
05/11/2023	Added more figures in Section 3.

4.9 HMI

4.9.1 Description

The HMI, or Human Machine Interface, is an essential component of the system that provides a user-friendly way to interact with the system and understand its status. It consists of a toggle button and an RGB LED, which displays blue for personal trips and green for business trips, indicating when the vehicle mileage is being recorded. The HMI is mounted on top of the 3D printed enclosure, providing a convenient location for easy access and visibility. The toggle button allows the user to switch between personal and business modes, while the LED provides clear and visible feedback on the current mode. Overall, the HMI enhances the system's usability and provides a simple and intuitive way for users to interact with the system and understand its status.

4.9.2 Design

The design process for the HMI involved careful consideration of the user's needs and preferences. The toggle button and LED were chosen for their simplicity and effectiveness in conveying information. To ensure ease of use, the toggle button was placed in a prominent location on top of the enclosure, making it easily accessible and visible to the user. Similarly, the RGB LED was placed in a visible location on the top of the enclosure, ensuring that the user can easily see the current mode of operation. The color choice for the LED, blue for personal trips and green for business trips, was selected for its clarity and visual appeal. Overall, the design of the HMI was informed by the principles of usability and simplicity, with a focus on providing an intuitive and user-friendly interface.

In addition to its functional aspects, the HMI was also designed with aesthetics in mind. The toggle button and RGB LED were selected for their sleek and modern appearance, which

complements the overall design of the system. The placement of the HMI on top of the 3D printed enclosure not only provides functional benefits but also contributes to the overall visual appeal of the system. The clean lines and minimalistic design of the HMI and enclosure create a cohesive and stylish look that is both functional and visually pleasing.

Finally, the HMI was tested extensively to ensure that it met the required specifications and provided a seamless user experience. The toggle button and LED were tested for their responsiveness and accuracy, and adjustments were made to ensure that they functioned smoothly and reliably especially with debouncing tuning. The HMI was also tested in various lighting conditions to ensure that the LED was clearly visible and the button was easily distinguishable. Through testing and refinement, the HMI was designed to provide the best possible user experience and meet the requirements of the system.

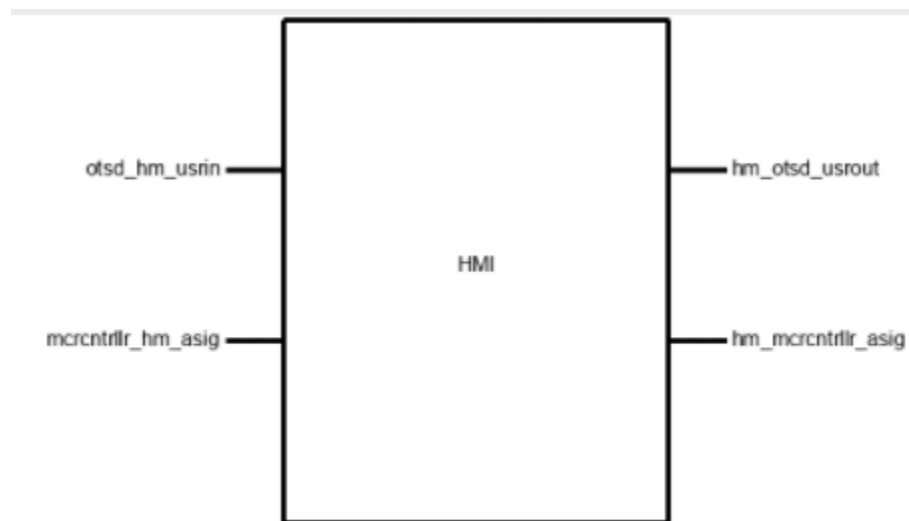


Fig. 1: This figure shows the inputs and output interfaces to the HMI block.



Fig. 2: This is a picture of the HMI in action.

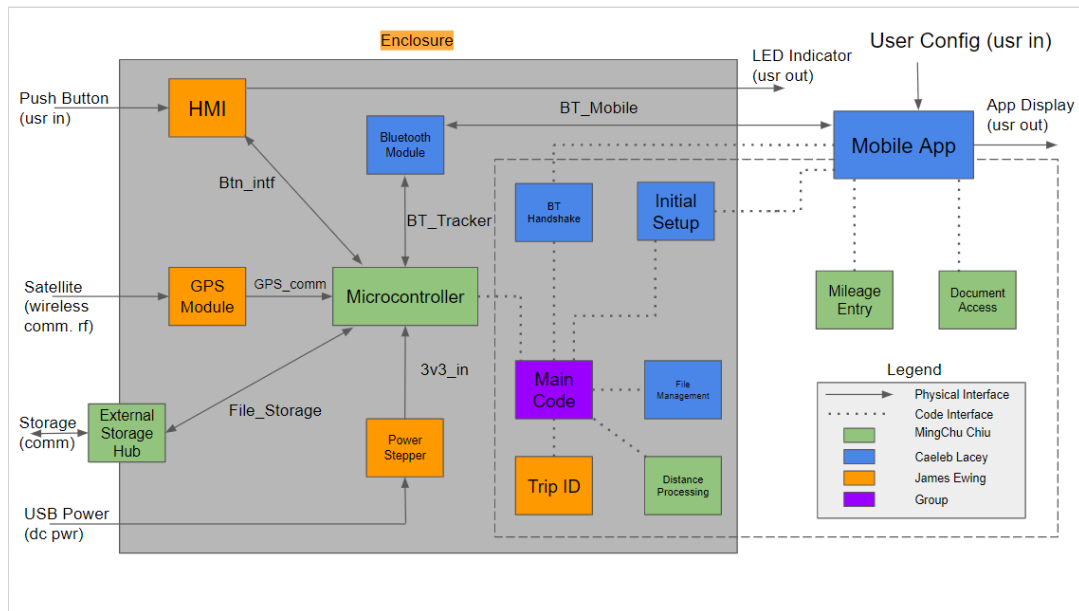


Fig. 3: This is the high level block diagram for the overall system that the HMI is apart of.

4.9.3 General Validation

Throughout the design process of the Human-Machine Interface (HMI), careful consideration was given to the user's needs and preferences. The toggle button and RGB LED were selected for their simplicity and effectiveness in conveying information to the user. The RGB LED is visible from a distance and its green color is clear, indicating a business trip while the blue color is less confrontational and not an annoyance to the user. The toggle button was placed in a prominent location on top of the enclosure, making it easily accessible and preventing drivers from taking their attention away from the road for too long. The design of the HMI was informed by the principles of usability and simplicity, with a focus on providing an intuitive and user-friendly interface.

The aesthetics of the HMI were also taken into account during the design process. The toggle button and RGB LED were selected for their sleek and modern appearance, which complemented the overall design of the system. The placement of the HMI on top of the 3D printed enclosure not only provided functional benefits but also contributed to the overall visual appeal of the system. The clean lines and minimalistic design of the HMI and enclosure created a cohesive and stylish look that was both functional and visually pleasing.

Finally, the HMI underwent extensive testing to ensure that it met the required specifications and provided a seamless user experience. The toggle button and RGB LED were tested for their responsiveness and accuracy, and adjustments were made to ensure that they functioned smoothly and reliably, including debouncing tuning. The HMI was tested in various lighting conditions to ensure that the LED was clearly visible, and the button was easily distinguishable. Through testing and refinement, the HMI was designed to provide the best possible user experience and meet the requirements of the system. The result was an intuitive and user-friendly HMI that complemented the overall design of the system, providing both functionality and style.

4.9.4 Interface Validation

Otsd_hm_usrin: Input

Other: When the button is actuated, it triggers only once (debouncing check).	Inspection	
Other: Button can be actuated 10 times and 9 of the 10 times the signal has	Inspection	

to be received.		
Timing: GPIO logic pin on microcontroller changed on button release.	Inspection	

Mrcntrlr_hm_asig: Output

Other: Green LED - Business Mode	Inspection	
Other: Red LED - Error Report	Inspection	
Other: Blue LED - Personal Mode	Inspection	
Vmax: 3.3 V	Measurement	

Hm_otsd_usrout: Output

Other: Green LED is visible for Business Trip	Inspection	
Other: Light is visible from 3 feet away.	Inspection	
Other: Blue LED is visible from for Personal Trip	Inspection	

hm_mrcntrlr_asig: Output

Other: Push	Measurement	
-------------	-------------	--

button connected to the internal pull-up resistor of Rpi (active low signal)		
Vmax: 3.3	Measurement	

4.9.5 Verification Process

- Test 1:**
1. Provide power to the system at 5V.
 2. Observe onboard LED. Confirm state: Blue = Personal, Green = Business.
 3. Press the onboard button once.
 4. Observe the onboard LED once more. The LED will have swapped colors, indicating the shift to the other trip mode.
 5. Record a trip, then check the trip CSV.
 6. Confirm that the trip state displayed by the LED aligns with the trip type listed in the most recent trip entry.

4.9.6 References and File Links

[1] Metal thin film chip resistors - 進工業株式会社(susumu) | 薄膜 ..., https://www.susumu.co.jp/common/pdf/n_catalog_partition01_en.pdf (accessed May 16, 2023).

[2] WP154A4SUREQBFZGC T-1 3/4 (5mm) full color led Lamp - Kingbright USA, <https://www.kingbrightusa.com/images/catalog/spec/WP154A4SUREQBFZGC.pdf> (accessed May 16, 2023).

4.9.7 Revision Table

Date	Revision
02/23/2023	Initial Creation.
02/30/2023	Added more detailed references for interface

	definitions.
03/10/2023	Updated document based on feedback.
03/11/2023	Added more figures in Section 3.

4.10 GPS Module

4.10.1 Description

The NEO M8N GPS module is an integral part of the main PCB in the system. The module operates on the L1 band at 1575MHz and is equipped with a transceiver that connects to an external active antenna with 26dB amplification. The antenna node has an RLC filter implemented on it, ensuring high-accuracy impedance matching. The GPS module can achieve a high accuracy of 2.5m in high accuracy mode, making it suitable for various applications that require precise location data. To communicate with the Raspberry PI Pico microcontroller, the NEO M8N GPS module uses UART communication, specifically the UART version without CTS or RTS. This allows for a reliable and efficient exchange of data between the GPS module and the microcontroller. Additionally, the GPS module is integrated as part of the main PCB, allowing for a compact and streamlined design. The NEO M8N GPS module is a highly capable and accurate component that plays a crucial role in the system's overall functionality.

4.10.2 Design

The design process for the NEO M8N GPS module involved careful consideration of the required specifications and performance requirements for the system. The selection of the L1 band at 1575MHz was based on its availability and compatibility with the system's other components. The decision to use an external active antenna with 26dB amplification was made to ensure reliable and accurate reception of GPS signals, while the RLC filter implementation on the antenna node was necessary to achieve high-accuracy impedance matching. These choices ultimately led to the NEO M8N GPS module's ability to achieve a high accuracy of 2.5m in high accuracy mode, making it suitable for various applications that require precise location data.

The selection of UART communication for the NEO M8N GPS module was based on its reliability and efficiency for exchanging data between the GPS module and the microcontroller. Specifically, the UART version without CTS or RTS was chosen to simplify the design and reduce complexity. The integration of the GPS module as part of the main PCB allowed for a compact and streamlined design, making it easier to manufacture and assemble.

Throughout the design process, careful consideration was given to the system's overall functionality and performance requirements. The NEO M8N GPS module's highly capable and accurate performance was a crucial factor in ensuring the system's overall success. The design process involved a balance between selecting the necessary components, optimizing their performance, and integrating them into the system's design to achieve the desired outcome.

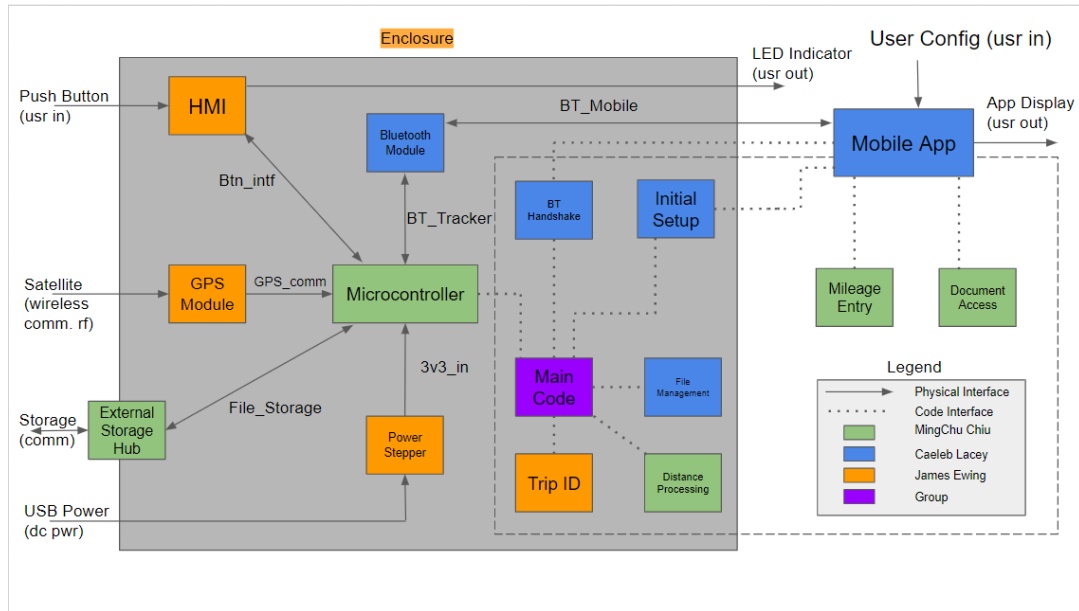


Fig. 3: This is the high level block diagram for the overall system that the GPS module is apart of.

4.10.3 General Validation

The design of the NEO M8N GPS module went through a thorough validation process to ensure its accuracy and functionality met the requirements of the system. The module's high-accuracy mode provides a location accuracy of 2.5m, making it highly suitable for various applications requiring precise location data. The built-in functions of the NEO M8N, including odometer and speed calculations, made it a useful choice for mileage tracking duties, and its compatibility with the system's other components allowed for streamlined integration.

The selection of an active antenna with 26dB amplification and an RLC filter implementation on the antenna node was necessary to achieve high-accuracy impedance matching. The active antenna's ability to pick up signals in noisy environments commonly found in cars was critical for reliable GPS reception. The transceiver was simple to tune with the RLC filter, and this combination allowed for optimal signal strength and reception.

During the design process, considerations were made for the system's overall functionality and performance requirements. The NEO M8N GPS module's high accuracy and reliable performance were integral to the system's overall success. The integration of the GPS module into the main PCB enabled a compact and streamlined design that was easier to manufacture and assemble. Overall, the NEO M8N GPS module met the system's requirements and validated the design choices made during its selection and integration.

4.10.4 Interface Validation

otsd_gps_md1_rf: Input

Other: Receive Positional data	Communication established.	
Other: Operating frequency of 1575MHz on the L1 band.	Communication established.	
Protocol: GPS Network Operation	Communication established.	

gps_md1_mrcntrlr_comm: Output

Datarate: 19200	Communication established.	
Protocol: UART	Communication established.	
Protocol: SPI	Communication established.	

pwr_stppr_gps_md1_dcpwr: Input

Inominal: 50mA	This nominal current was chosen based on the expected current needs of the system overall. The	For the BM78 bluetooth module: <ul style="list-style-type: none">• 43mA for continuous TX condition, rounded up to 50mA. [2]
Ipeak: 100mA	In this example, we don't expect the current draw of the whole system to ever spike above this number. The value was selected by adding up the maximum current of all parts and multiplying by 2	For the BM78 bluetooth module: <ul style="list-style-type: none">• Peak current draw 100mA from on board LDO. [2]
Vnominal: 3.3V	In this example, this property was	For the BM78 bluetooth module:

	chosen based on the design we plan to use in the power supply block.	<ul style="list-style-type: none"> Nominal operating voltage listed as 3.3V. [2]
--	--	---

Pwr_stppr_gps_mdI_dcpwr: Output

Inominal: 30mA	This nominal current was chosen based on the expected current needs of the system overall. The	For the NEO-M8N: <ul style="list-style-type: none"> 27mA typical current listed. Rounded up to 30mA. [4] (pg. 19)
Ipeak: 70mA	In this example, we don't expect the current draw of the whole system to ever spike above this number. The value was selected by adding up the maximum current of all parts and multiplying by 2	For the NEO-M8N: <ul style="list-style-type: none"> 67mA max current rating listed. Rounded up to 70mA [4] (pg 19.)
Vnominal: 3.3V	In this example, this property was chosen based on the design we plan to use in the power supply block.	For the NEO-M8N: <ul style="list-style-type: none"> 3.0V typical voltage listed for M8N [4] (pg. 18)

4.10.5 Verification Process

- Test 1:**
1. Create a route on Google Maps.
 2. Turn the device on.
 3. Using Business Mode of the device, record a trip that accords to the created route.
 4. Compare the recorded GPS distance with the mileage shown on Google Maps.

4.10.6 References and File Links

[1] "NEO-M8 series," u, <https://www.u-blox.com/en/product/neo-m8-series> (accessed May 15, 2023).

[2] "NEO-M8 series," u, <https://www.u-blox.com/en/product/neo-m8-series> (accessed May 15, 2023).

[3] "NEO-M8 series," u, <https://www.u-blox.com/en/product/neo-m8-series> (accessed May 15, 2023).

4.10.7 Revision Table

Date	Revision
04/23/2023	Initial Creation.
04/30/2023	Added more detailed references for interface definitions.
05/10/2023	Updated document based on feedback.
05/11/2023	Added more figures in Section 3.

4.11 Power Stepper

4.11.1 Description

Power conversion circuitry to supply our logic circuits with the appropriate voltage. This block steps a 5V DC input down to DC 3.3V. The 5V DC input is from a USB car charger as typically seen plugged into a cars cigarette lighter. The specific logic circuits being supplied by the 3.3V DC output of the power stepper are the SD card, GPS module, and Bluetooth module. The purpose for this block is specifically to supply enough current for the three logic circuits. The daughtered microcontroller used for our system is capable of supplying 3.3V at 300mA peak. This is short of the 400mA peak current draw for our system which created the need for a specific power conversion circuit.

4.11.2 Design

The microcontroller we are using is the Raspberry Pi Pico which can be powered through the micro usb connector or Vin pin on the board with 5V. The Pico has an on board buck-boost converter in which it then steps down the 5V input to 3v3 to power the logic of the board and also supply the GPIO pins and more specifically the 3v3 Out pin. The max current that can be supplied by the 3v3 Out pin is 300mA. Our total max draw for powering the microcontroller, GPS module, Bluetooth module, and SD card module is 400mA. This being the case, we were unable to power the whole circuit off of the 3v3 Out pin on the Raspberry Pi Pico since our worst case exceeded the rating of our 3v3 pin.

Additional power load may be found in testing with the antenna for the GPS module. From my research and experience antennas can have extreme load conditions for a very short amount of time that can cause significant issues for the system. Further validation of the design will need to be done in a variety of conditions to ensure that the antenna load conditions are met in addition to the rest of the systems components.

The solution that I came up with is to power the logic circuits external to the Raspberry Pi Pico with a separate power path. This power path is from the 5V DC input node through the NCP59748 linear regulator to supply 3v3 to the SD card, Bluetooth module, and GPS module with the capability of a max current supply of 1.5A. The supporting components for the linear regulator include C1, C3, and C4 10uF capacitors for filtering on the input and output nodes. C2 10nF capacitor is responsible for setting the soft start time for the linear regulator. Resistors R1 and R2 are responsible for setting the output voltage regulation to 3v3. R3 is providing high impedance as the Power Good functionality is not utilized.

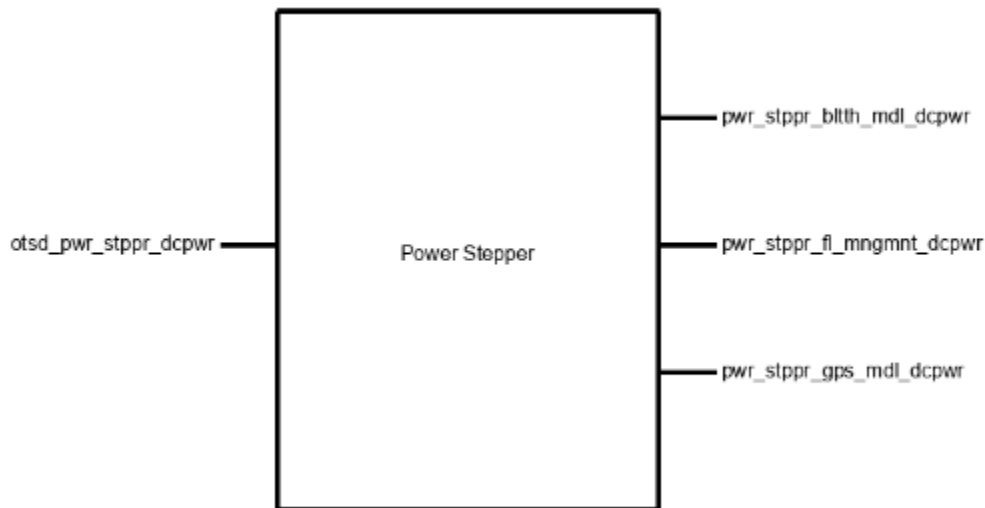


Fig. 1: The output interfaces of the Power Stepper Block are all grouped together and connected to the LDO_Output node in the schematic below.

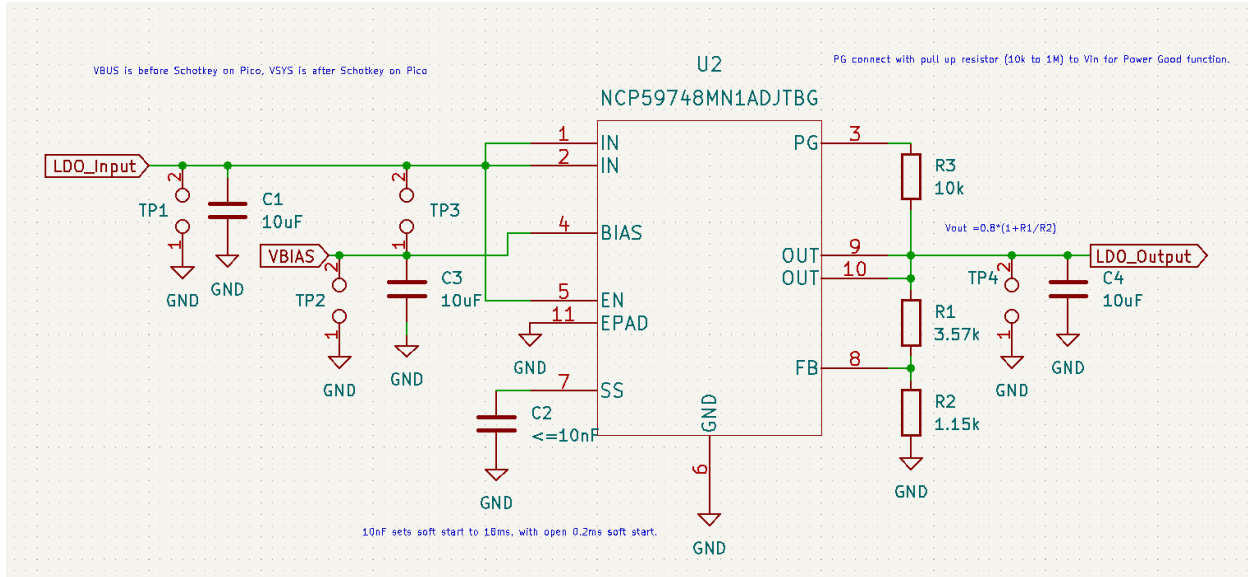


Fig. 2: This is the schematic of the NCP59748MN1ADJTBG linear regulator circuit.

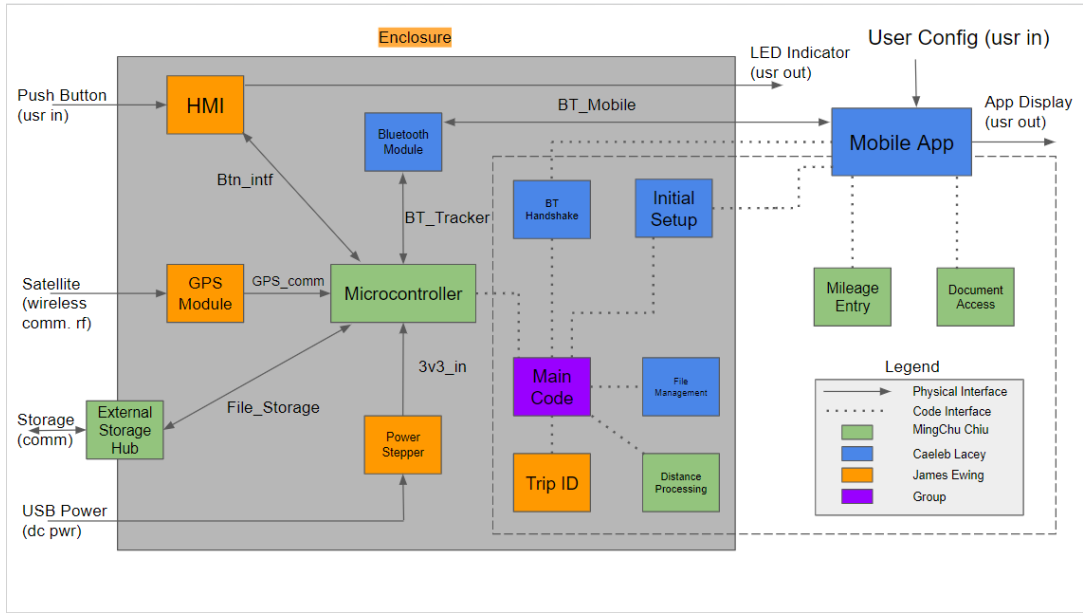


Fig. 3: This is the high level block diagram for the overall system that the power stepper is apart of.

4.11.3 General Validation

The pcb design for the Power Stepper block is appropriate when considering the bill of material costs. The Power Stepper design utilizes the NCP59748 linear regulator in the DFN10 package. Linear regulators of similar specifications hover between the \$1.50 to \$3.00 range with the more expensive examples correlating to smaller package size regulators. The NCP59748

offered one of the smallest package sizes with the DFN10 coming in at 3mm x 3mm while still costing under \$2.00 at \$1.92. The other expensive item for the board is the PCB itself which came in at \$2.82 shipped from OSHPark. The per board price will drop significantly when ordered in greater quantities than the 3 for prototyping and from larger board houses such as JLCPCB. The total cost for the board including the aforementioned items and passive components is \$5.38. Price comparison to other premade modules for 5V to 3.3V conversion bodes well for the design as the least expensive premade module that could be purchased came in at \$8.00

The components chosen for the design all had in stock quantities on Digi-Key exceeding 3,600 pieces. The lowest stock component was the 10nF capacitor responsible for setting the soft start time for the linear regulator. There were many alternative options for the 10nF on Digi-Key with similar stock listed. Furthermore the soft start feature in which the 10nF is populated for is not necessary for our application. These reasons justified the selection of the C1206C103K6RAC7800 10nF capacitor.

The selection of the NCP59748 linear regulator simplified the Power Stepper block for the system greatly. The alternative design for the Power Stepper block included current steering due to the initial source combination of the buck-boost converter on the Raspberry Pi Pico and an external buck converter to rectify the current waveform of the 3.3V node. Current steering can be tricky and most importantly time intensive to get working correctly especially during peak conditions. The selection of a linear regulator allowed the circuit to be greatly simplified and the engineering time reduced when compared to the current steering design. Furthermore the design is much more modular as it isn't dependent on the specific buck-boost converter characteristics of the Raspberry Pi Pico.

Technical performance is the primary reason for the selection of the NCP59748 linear regulator as it performed the best in thermal, max power loss, and peak current draw conditions. In 40 deg C conditions the junction temperature saw a thermal max of 55.8 deg C which is well in range of what the DFN10 package can handle. The peak power loss condition saw 691mW which was reasonable considering the load conditions of 405mA peak current draw.

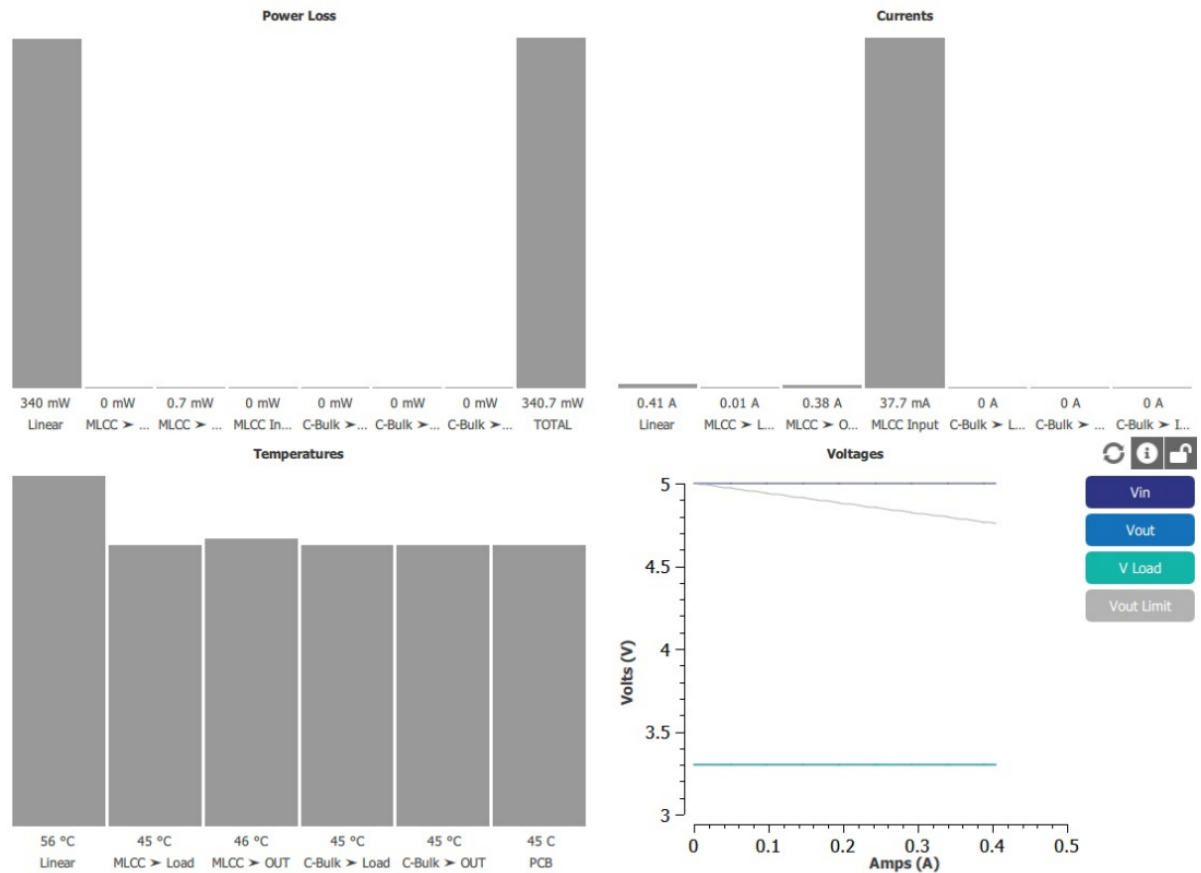


Fig. 4: This shows the simulation results for the NCP59748 linear regulator in the expected conditions of the system.

The Linear Regulator DFN10 package size will allow the final result to be very small and fit well within our projected max dimension sizes of the enclosure for the system. The passive components currently chosen for the Power Stepper Board could be reduced in size by 2 package sizes. Currently the components are all Standard 1206 package size to allow for ease of hand soldering for the prototype. 0603 package size passive components can be selected and the overall footprint of the circuit could be reduced significantly if it is required.

The layout design for the Power Stepper PCB was executed with the main focus of ease of fabrication and testing ahead of the actual implementation within the larger system. Test points TP1 and TP4 are placed on the Voltage input and Voltage output nodes respectively of the linear regulator to allow the board to be compatible with a variety of connection styles for testing. TP3 is placed between the Vin node and the Vbias node of the linear regulator to allow for the test case of whether the Vbias can be connected to the same node as Vin. TP2 is placed to allow a Vbias to be introduced to the linear regulator incase Vbias can't be jumped to Vin. The Vin and Vout nodes are pairs of equal width to the test point pad outer diameters.

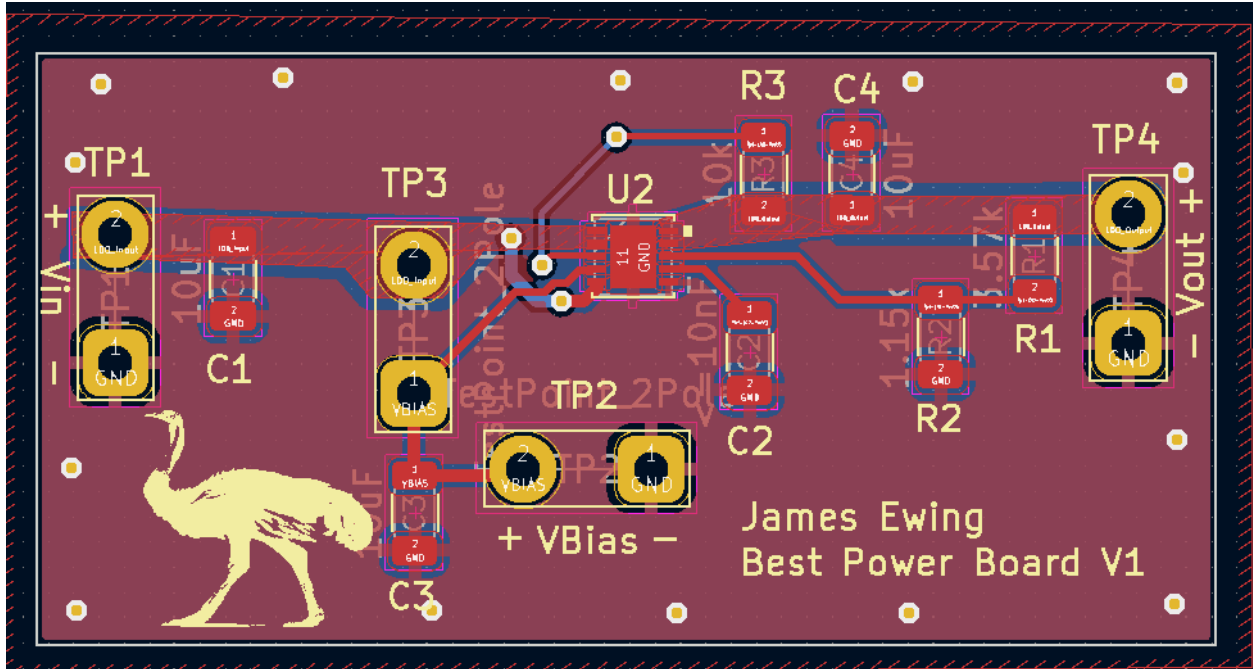


Fig. 5: This is the layout of the Power Stepper board as described by the schematic and block diagram shown in Fig. 2 and Fig. 1 respectively.

4.11.4 Interface Validation

Otsd_pwr_stppr_dcpwr: Input

Inominal: 100mA	This nominal current was chosen based on the expected current needs of the system overall. The	For the NCP59748 in a DFN10 package: <ul style="list-style-type: none"> Component is rated to +1.5A at our target junction temperature of 70 degrees C. Nominal current is well within range. [1] (Absolute Max Ratings, pg. 3) In 40C air, the component can handle 9W of dissipation. This current would be ~.7W of dissipation. [1] (pg.7)
Ipeak: 400mA	In this example, we don't expect the current draw of the whole system to ever spike above this number. The value was selected by adding up the maximum current of all parts and multiplying	For the NCP59748 in a DFN10 package: <ul style="list-style-type: none"> At 3.3V output, maximum of +3A can be supplied. [1] (Typical Characteristics, pg. 7) In 125C Junction Temperature,

	by 2	the component can handle 9W of dissipation. 9W dissipation is the worst case. [1] (Typical Characteristics table, pg. 7) Since 3A current is well beyond our worst case scenario we foresee no issues (Judgment Call)
Vmax: 5.5V	In this example, this property was chosen based on the design we plan to use in the power supply block.	For the NCP59748 in a DFN10 package: <ul style="list-style-type: none"> • Nominal voltage input is 5.5V. [1] (Max ratings table, pg. 4) • Voltage dropout from input to output is 165mV at 5.5V. [1] (Absolute max ratings, pg. 3)

Pwr_stppr_blthh_mdl_dcpwr: Output

Inominal: 50mA	This nominal current was chosen based on the expected current needs of the system overall. The	For the BM78 bluetooth module: <ul style="list-style-type: none"> • 43mA for continuous TX condition, rounded up to 50mA. [2]
Ipeak: 100mA	In this example, we don't expect the current draw of the whole system to ever spike above this number. The value was selected by adding up the maximum current of all parts and multiplying by 2	For the BM78 bluetooth module: <ul style="list-style-type: none"> • Peak current draw 100mA from on board LDO. [2]
Vnominal: 3.3V	In this example, this property was chosen based on the design we plan to use in the power supply block.	For the BM78 bluetooth module: <ul style="list-style-type: none"> • Nominal operating voltage listed as 3.3V. [2]

Pwr_stppr_fl_mngmnt_dcpwr: Output

Inominal: 15mA	This nominal current was chosen based on the expected current needs of the system overall. The	For the SD storage in a microSD package: <ul style="list-style-type: none"> Nominal current listed for write at 3.3V is 15mA [3]
Ipeak: 170mA	In this example, we don't expect the current draw of the whole system to ever spike above this number. The value was selected by adding up the maximum current of all parts and multiplying by 2	For the SD storage in a microSD package: <ul style="list-style-type: none"> Peak current listed as 163mA for 3.3V, rounded up to 170mA for spec. [3]
Vnominal: 3.3V	In this example, this property was chosen based on the design we plan to use in the power supply block.	For the SD storage in a microSD package: <ul style="list-style-type: none"> Nominal voltage listed as 3.3V. [3].

Pwr_stppr_gps_mdl_dcpwr: Output

Inominal: 30mA	This nominal current was chosen based on the expected current needs of the system overall. The	For the NEO-M8N: <ul style="list-style-type: none"> 27mA typical current listed. Rounded up to 30mA. [4] (pg. 19)
Ipeak: 70mA	In this example, we don't expect the current draw of the whole system to ever spike above this number. The value was selected by adding up the maximum current of all parts and multiplying by 2	For the NEO-M8N: <ul style="list-style-type: none"> 67mA max current rating listed. Rounded up to 70mA [4] (pg 19.)
Vnominal: 3.3V	In this example, this property was chosen based on the design we plan to use in the power supply block.	For the NEO-M8N: <ul style="list-style-type: none"> 3.0V typical voltage listed for M8N [4] (pg. 18)

4.11.5 Verification Process

Test 1: Nominal current test.

Step 1: Turn on voltage supply, make sure output is not enabled and set voltage supply to 5V.

Step 2: Turn on DC load, make sure output is not enabled, and set DC load to 100mA constant current.

Step 3: Connect Vin terminals to 5V supply.

Step 4: Connect multimeter in series with the Vout (pwr_stppr_bltth_mdl_dcpwr, pwr_stppr_fl_mngmnt_dcpwr, and pwr_stppr_gps_mdl_dcpwr interfaces combined) positive line and the positive terminal of the DC load and set the multimeter to current measurement mode.

Step 5: Connect GND terminal of Vout to GND of DC load.

Step 6: Enable output on the Voltage Supply.

Step 7: Enable output on DC load.

Step 8: Record current output shown on multimeter.

Step 9: Shutoff Load.

Step 10: Shutoff Voltage Supply.

Step 11: Disconnect Vin and Vout terminals from Supply and Load.

Test 2: Peak current test.

Step 1: Turn on voltage supply, make sure output is not enabled and set voltage supply to 5V.

Step 2: Turn on DC load, make sure output is not enabled, and set DC load to 400mA constant current.

Step 3: Connect Vin terminals, interface otsd_pwr_stppr_dcpwr, to 5V supply.

Step 4: Connect multimeter in series with the Vout (pwr_stppr_bltth_mdl_dcpwr, pwr_stppr_fl_mngmnt_dcpwr, and pwr_stppr_gps_mdl_dcpwr interfaces combined) positive line and the positive terminal of the DC load and set the multimeter to current measurement mode.

Step 5: Connect GND terminal of Vout to GND of DC load.

Step 6: Enable output on the Voltage Supply.

Step 7: Enable output on DC load.

Step 8: Record current output shown on multimeter.

Step 9: Shutoff Load.

Step 10: Shutoff Voltage Supply.

Step 11: Disconnect Vin and Vout terminals from Supply and Load.

Test 3: Nominal Voltage

Step 1: Turn on voltage supply, make sure output is not enabled and set voltage supply to 5V.

Step 2: Turn on DC load, make sure output is not enabled, and set DC load to 100mA constant current.

Step 3: Connect Vin, interface otsd_pwr_stppr_dcpwr, terminals to 5V supply.
Step 4: Connect Vout positive line and the positive terminal of the DC load and set the multimeter to DC voltage measurement mode.
Step 5: Connect GND terminal of Vout to GND of DC load.
Step 6: Enable output on the Voltage Supply.
Step 7: Enable output on DC load.
Step 8: Touch positive and negative cable of multimeter to Vout (pwr_stppr_bltth_mdI_dcpwr, pwr_stppr_fl_mngmnt_dcpwr, and pwr_stppr_gps_mdI_dcpwr interfaces combined) positive and negative terminals, record the voltage shown on the multimeter.
Step 9: Shutoff Load.
Step 10: Shutoff Voltage Supply.
Step 11: Disconnect Vin and Vout terminals from Supply and Load.

Test 4: Max input voltage.

Step 1: Turn on voltage supply, make sure output is not enabled and set voltage supply to 5V.
Step 2: Turn on DC load, make sure output is not enabled, and set DC load to 100mA constant current.
Step 3: Connect Vin, interface otsd_pwr_stppr_dcpwr, terminals to 5.5V supply.
Step 4: Connect Vout positive line and the positive terminal of the DC load and set the multimeter to DC voltage measurement mode.
Step 5: Connect GND terminal of Vout to GND of DC load.
Step 6: Enable output on the Voltage Supply.
Step 7: Enable output on DC load.
Step 8: Touch positive and negative cable of multimeter to Vout (pwr_stppr_bltth_mdI_dcpwr, pwr_stppr_fl_mngmnt_dcpwr, and pwr_stppr_gps_mdI_dcpwr interfaces combined) positive and negative terminals, observe voltage for 1 minute to see if voltage drops from 3.3V on output (evidence of thermal shutdown).
Step 9: Shutoff Load.
Step 10: Shutoff Voltage Supply.
Step 11: Disconnect Vin and Vout terminals from Supply and Load.

4.11.6 References and File Links

[1] "NCV59748 - linear voltage regulator with bias rail, 1.5 a ." [Online]. Available: <https://www.onsemi.com/pdf/datasheet/ncv59748-d.pdf>. [Accessed: 11-Feb-2023].

[2] "BM78-Bluetooth Dual Mode Module - mouser.com." [Online]. Available: https://www.mouser.com/datasheet/2/268/BM78_Bluetooth_Dual_Mode_Module_DS60001380-2999436.pdf. [Accessed: 11-Feb-2023].

[3] "SD Standard Overview: SD Association," *SD Association | The SD Association*, 09-May-2022. [Online]. Available: <https://www.sdcard.org/developers/sd-standard-overview/>. [Accessed: 11-Feb-2023].

[4] "U-blox M8 high precision GNSS modules." [Online]. Available: https://content.u-blox.com/sites/default/files/NEO-M8P_HardwareIntegrationManual_UBX-15028081.pdf. [Accessed: 11-Feb-2023].

4.11.7 Revision Table

Date	Revision
01/23/2023	Added more thorough test steps.
01/30/2023	Added more detailed references for interface definitions.
02/10/2023	Updated document based on feedback.
02/11/2023	Added more figures in Section 3.

4.12 Trip ID

4.12.1 Description

4.12.2 Design

4.12.3 General Validation

4.12.4 Interface Validation

4.12.5 Verification Process

4.12.6 References and File Links

4.12.7 Revision Table

4.13 Main Code

4.13.1 Description

The Main Code block of our design is a representative block which encompasses all aspects of code blocks which operate the mileage tracker device itself. This includes the Bluetooth Handshake(Section 4.2), File Management(Section 4.4), Distance Processing(Section 4.7), and Trip ID(Section 4.12) blocks, as well as the communication

of outputs/inputs to/from the Bluetooth Module(Section 4.1), External Storage Hub(Section 4.5), HMI(Section 4.9), and GPS Module(Section 4.10) blocks.

5 System Verification Evidence

5.1 Universal Constraints

5.1.1 The system may not include a breadboard

Our system currently meets this requirement.

Evidence shown by picture below.



Figure 5.1.1: Final System Design

5.1.2 The final system must contain a student designed PCB with greater than 30 pads.

Our system meets this constraint by having a student designed power PCB that has at least 30 pads on it.

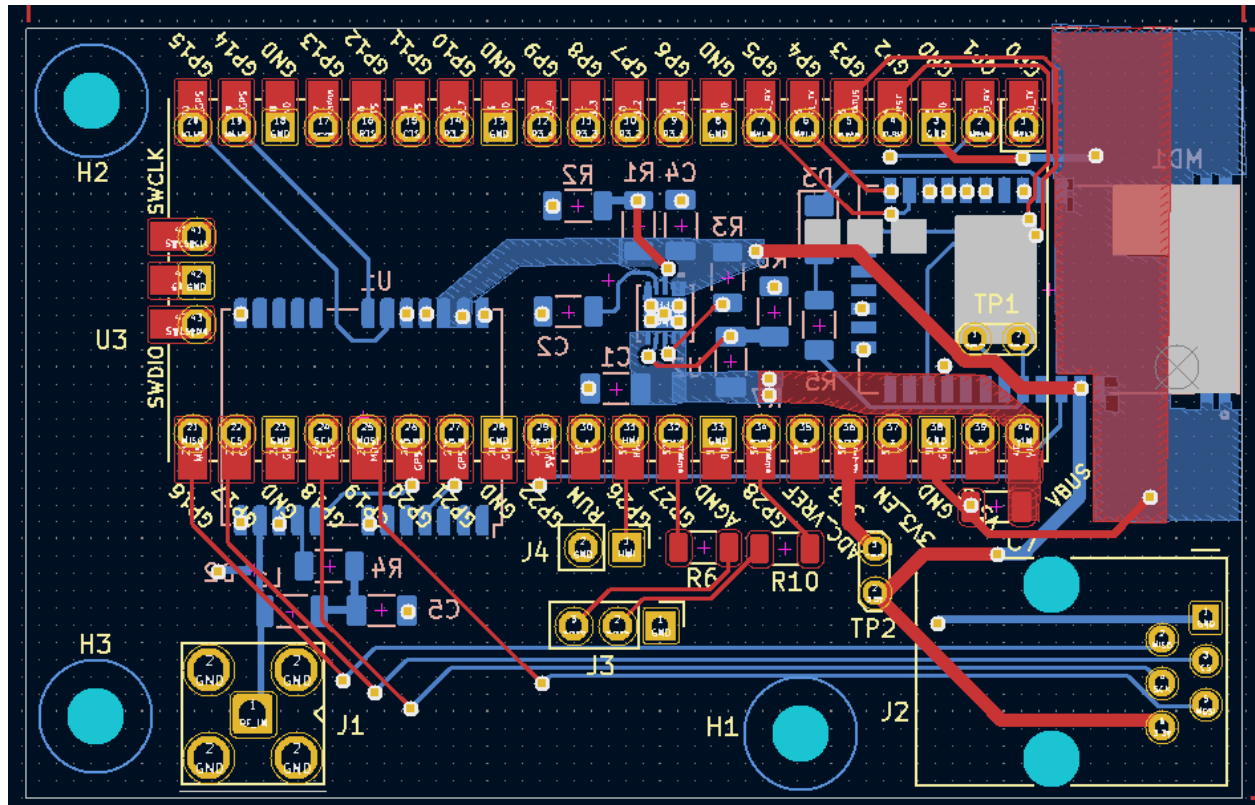


Figure 5.1.2: Shows our surface mounted Pico that meets the 30+ pad requirement by itself.

Total pads (excluding through hole mounted components) totals to 140 pads.

5.1.3 All connections to PCBs must use connectors.

Our system meets this constraint.

Evidence shown by image.



5.1.4 All power supplies in the system must be at least 65% efficient.

Our system meets this constraint by calculating the ratio between input power and output power. It is calculated to be 64.9% efficient.

Evidence is from block validation checkoff for the power stepper block. The math from that video shows $P_{in} = I_{in} * V_{in} = 0.505 \text{ W}$ and $P_{out} = I_{out} * V_{out} = 0.329 \text{ W}$. Efficiency is $P_{out}/P_{in} = 0.651$ which means the Linear Regulator is **65.1% efficient**.

Video Link:

https://drive.google.com/drive/folders/1frckB6ai1IFUrfxvq06DCMyX1Lv-9Yic?usp=share_link

5.1.5 The system may be no more than 50% built from purchased 'modules.'

Built Blocks	Purchased Blocks
Enclosure	External Storage Hub
HMI	
Power Stepper	
Bluetooth Module	
GPS Module	
Microcontroller	

We have more hand-built blocks in our system than we do purchased blocks. Our final percentage of built is **85.7% built**.

Photo evidence below.

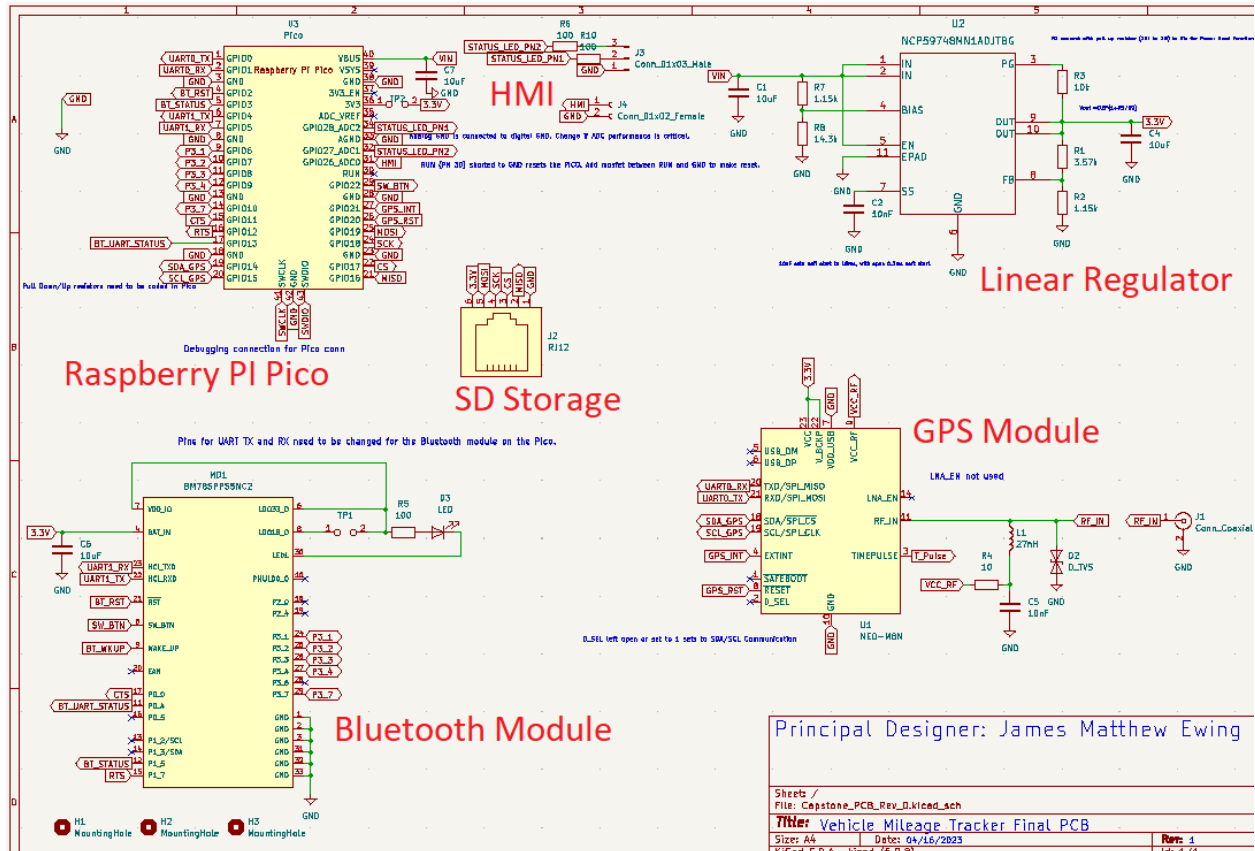


Figure 5.1.5.1: This picture shows all of the components included on the PCB and also therefore shows all of the components that are built.

5.2 Project Requirements

Below are the engineering requirements we must meet with the final system. These are derived from the requirements set forth by our project partner, and are converted to measurable engineering requirements with set tests to confirm that each has been met.

5.2.1. Bluetooth Configurable

5.2.1.1. Project Partner Requirement:

Mileage, odometer readings, and trip intention are editable via Bluetooth by users from mobile devices.

5.2.1.2. Engineering Requirement:

The system will enable 9 out of 10 users to edit odometer and trip purpose data stored on local memory of the mileage tracker device via mobile devices and report that 'it was easy to use'.

5.2.1.3. Testing Method:

Demonstration.

5.2.1.4. Verification Process:

1. Instruct the user to open the mobile application's guided walkthrough.
2. Upon finishing the walkthrough, instruct the user to start a trip, then end the trip through the application.
3. Instruct the user to open a file, edit the data on a trip, save the edit, and then confirm that the edit was saved.
4. Ask the user if the mobile application was "easy to use".
5. Repeat for 10 users.

5.2.1.5. Pass Condition:

Test passes if 9 out of 10 users consider the file editing process to be easy to use.

5.2.1.6. Testing Evidence:

No testing evidence available at this time. The bluetooth block of the main device is not currently functioning, and thus this functionality cannot be fulfilled.

N/A

5.2.2. Compact Packaging

5.2.2.1 Project Partner Requirement:

A small unit capable of recording trip information.

5.2.2.2 Engineering Requirement:

The system will be smaller than 3.75x2.75x2 inches.

5.2.2.3 Testing Method:

Inspection.

5.2.2.4 Verification Process:

1. Turn on digital calipers and set units to inch by pressing the mode button.
2. Measure length of the enclosure with the digital calipers.
3. Measure width of the enclosure with the digital calipers.
4. Measure height of the enclosure with the digital calipers.

5.2.2.5 Pass Condition:

Length is measured to be less than 3.75 inches, Width is measured to be less than 2.75 inches, and Height is measured to be less than 2 inches.

5.2.2.6 Testing Evidence:

Take pictures of the length, width, and height measurements shown on the screen of digital calipers.



Figure 5.2.2.6.1: This image shows the length measurement of the enclosure.



Figure 5.2.2.6.2: This image shows the width measurement of the enclosure.



Figure 5.2.2.6.3: This image shows the height measurement of the enclosure.

Our system has dimensions of 3.39" x 2.33" x 1.33", which fits the requirement of being within 3.75" x 2.75" x 2".

5.2.3. Driver Safety

5.2.3.1 Project Partner Requirement:

This device should not cause driver safety concerns.

5.2.3.2 Engineering Requirement:

The system will not allow the user to make changes via the user interface within 30 seconds of beginning travel.

5.2.3.3 Testing Method:

Inspection.

5.2.3.4 Verification Process:

1. Have one user designated as the driver while another user is the data collector. Do not begin driving until Step 3.
2. TEST 1: Start timer and select Begin Trip in the mobile application. Input the current odometer and stop the timer upon landing on the lockout mid-trip page, note the time, and end the trip within the application.
3. TEST 2: While remaining on the main page of the mobile application, restart the timer and begin driving.
4. Upon the application automatically progressing to the lockout mid-trip page, stop the timer and note the time.

5.2.3.5 Pass Condition:

Test passes if both tests demonstrate a lockout time of less than 30 seconds from the start of the trip.

5.2.3.6 Testing Evidence:

No testing evidence available at this time. The bluetooth block is not currently functioning and is required for the function of this requirement.

N/A

5.2.4. End User Documentation

5.2.4.1 Project Partner Requirement:

The device should include instructions for how to use the device.

5.2.4.2 Engineering Requirement:

The system will include documentation that 9 out of 10 users consider to be sufficient for operation of the system.

5.2.4.3 Testing Method:

Demonstration.

5.2.4.4 Verification Process:

1. Ask the user to read the instruction manual or use the Walkthrough option.
2. Have the user follow the guide step by step.
3. Upon finishing reading, instruct the user to demonstrate what the guide taught them by having them begin and end a trip, then edit a trip entry within a trip log.
4. Ask the user "was the documentation sufficient for the operation of the system?"

5.2.4.5 Pass Condition:

Test passes if 9 out of 10 users find the guided walkthrough sufficient for operation of the system.

5.2.4.6 Testing Evidence:

Video of one user test (survey feedback).

Google Drive Link:

<https://drive.google.com/drive/folders/1b3wl3K27PiIPM2xEdML3AsyuCw4i6ric?usp=sharing>

5.2.5. Neatly Presented Data

5.2.5.1 Project Partner Requirement:

Keep track of relevant business trip information automatically and deliver it to the user to help the user complete their yearly taxes.

5.2.5.2 Engineering Requirement:

The system will present accessed data such that 9 out of 10 users report it is easy to locate data from a specific trip using both the user interface and the csv file output and the recorded data will conform to IRS requirements for mileage reporting.

5.2.5.3 Testing Method:

Demonstration.

5.2.5.4 Verification Process:

1. Have the device generate an output on the SD card.
2. Showcase the output on the Mobile App.
3. Showcase the output on a PC.
4. Ask 10 people if they find the output on both platforms is easy to locate data from a specific trip and that it contains IRS relevant data fields such as “date”, “odometer readings”, “mileage”, “business purpose”.

5.2.5.5 Pass Condition:

Test passes if 9 out of 10 users agree that data from a specific trip is easy to locate.

5.2.5.6 Testing Evidence:

Video of one user test (survey feedback).

Google Drive Link:

<https://drive.google.com/drive/folders/1-x68g9djhBC1IQ1IGsat9sAKHiRNtSJ5?usp=sharing>

5.2.6. Tracking Vehicle Mileage

5.2.6.1 Project Partner Requirement:

The device will need to track trip distance accurately enough for IRS submission.

5.2.6.2 Engineering Requirement:

The system will record user input mileage data and an automated backup to ensure an error no greater than +/- 0.1 miles for every 5 miles of trip.

5.2.6.3 Testing Method:

Test.

5.2.6.4 Verification Process:

1. Create a route on Google Maps.
2. Turn the device on.
3. Using Business Mode of the device, record a trip that accords to the created route.
4. Compare the recorded GPS distance with the mileage shown on Google Maps.

5.2.6.5 Pass Condition:

If the error between the automated backup data and the mileage from Google Maps is less than 10%, then the requirement passes.

5.2.6.6 Testing Evidence:

The mileage recorded is 3.418 miles for the trip that we took from Dearborn to Winco. We expect to see 3.6 miles according to Google Maps. This means that our system has an error of $(3.418-3.6)/3.6 = -5.06\%$

Google Drive Link:

<https://drive.google.com/drive/folders/1eAmSDSJkmk6yeV-PTnHt9b0pLaKe5DfJ?usp=sharing>

5.2.7. Trip Identification

5.2.7.1 Project Partner Requirement:

The device will indicate the current trip mode to users—business or personal. Users should be able to change the mode of operation within a second.

5.2.7.2 Engineering Requirement:

The system will indicate trips as either business or personal in both the user interface and the csv file output.

5.2.7.3 Testing Method:

Demonstration.

5.2.7.4 Verification Process:

1. Provide power to the system at 5V.
2. Observe onboard LEDs.
Confirm state: Blue = Personal, Green = Business.
3. Press the onboard button once.
4. Observe the onboard LED once more. The LED will have swapped colors, indicating the shift to the other trip mode.
5. Record a trip, then check the trip CSV.
6. Confirm that the trip state displayed by the LED aligns with the trip type listed in the most recent trip entry.

5.2.7.5 Pass Condition:

The system will correctly indicate the current trip mode as reflected in the trip entry on the CSV.

5.2.7.6 Testing Evidence:

Video of process and screenshot of CSV opened in Microsoft Excel and or Google Sheets.

Video Link:

https://drive.google.com/drive/folders/169ahmKNtQqTK02LYJ3oYEvNN9mZbifFO?usp=share_link

5.2.8. Vehicle Powered

5.2.8.1 Project Partner Requirement:

The device will operate on vehicle power or battery power. The device should not use more power than what's charging the vehicle battery.

5.2.8.2 Engineering Requirement:

The system will draw vehicle or battery power without exceeding 5V or a 500mA current draw.

5.2.8.3 Testing Method:

Test.

5.2.8.4 Verification Process:

1. Turn on a power supply in the lab.
2. Set power supply to 5V with at least a current supply of 500mA on Channel 1 for Input Voltage.
3. Measure with a multimeter to ensure we are receiving 5V on our input to the power converter.
4. Look at power supply current being drawn to measure the nominal current load for the board and that it doesn't exceed 500mA while logging a trip.

5.2.8.5 Pass Condition:

The requirement passes if the system operates at 5V at no more than 500mA.

5.2.8.6 Testing Evidence:

Video showing the procedure and the power supply values after the steps completed.

Google Drive Link:

https://drive.google.com/drive/folders/1nO8unCqpr4vz7m8Y8Am76QLq6gawf5vj?usp=share_link

6 Project Closing

6.1 Future Recommendations

This section details future recommendations for potential further development of this project. Several areas of focus are considered, to include technical, global impact, and teamwork recommendations.

6.1.1 Technical Recommendations

6.1.1.1 Android OS Implementation: Currently, the system has only been developed to include an application utilizing Apple's iOS and CoreBluetooth framework. As development time was limited to this 9-month project, a decision needed to be made on a platform to focus development for, and iOS was most easily accessible to the team. With additional development time allotted, the creation of companion apps on other platforms, such as the equally popular Android OS, would allow for broader access to allow users to interact with the system. [1]

6.1.1.2 Odometer Scanning: Enable number recognition from the camera on the mobile device would improve the efficiency of user input on odometer readings of the vehicle. We have the option to input odometer readings manually on the Mobile App, which strengthens the accuracy of mileage tracking. Adding the odometer scanning functionality would allow the user to keep track of their trip more accordingly to their vehicle. [2]

6.1.1.3 Broader Positioning Network Access: While GPS is the standard positioning signal utilized in the United States, other nations have their own satellite systems in place which may provide more accurate readings when within their regions of focus. Adding more possible networks to access would allow for as accurate as possible tracking wherever you are when one positioning signal is not as strong or unavailable, and allow for more accurate tracking around the globe. [3]

6.1.1.4 Recognize the Usual Route: Add an algorithm on the microcontroller to recognize the usual GPS pings and suggest the user a potential Business Purpose based on the recognition of the route. This would save time for the user, as they would no longer need to type in details about the usual route. [4]

6.1.2 Global Impact Recommendations

6.1.2.1 Additional Language Support: The system is currently entirely developed in English only, and English is not a universal language. Adding additional localization support would allow for greater access to the system with a broader range of consumers, allowing for access globally beyond America and Europe. [5]

6.1.2.2 Adding Color Blindness Settings: We currently use 3 colors to represent the modes of operation of our device – Blue, Green, and Red. It would be a great addition to provide an option to indicate the modes of operation using blinks on the LED. With 1Hz blinks being Personal Mode, 2Hz blinks being Business Mode, and 3Hz blinks used for Error Reporting. [6]

6.1.3 Teamwork Recommendations

6.1.3.1 SMART Goals Presentation: Share with the team our SMART goals we each set at the end of lecture time, and present the progress we made on each SMART goals that we set out previously to keep each other accountable on weekly progress. [7]

6.1.3.2 Meeting Regularly: While our team did meet regularly, the meetings were often focused on other work in the course and less on collaborative work toward progress on our system itself. We recommend any future team developing on this system to meet up on a more regular basis, at least twice a week, to ensure that collaboration is being met across the board on a project where every block relies heavily on the function of another. [8]

6.2 Project Artifact Summary with Links

Microcontroller State Machine Code:

<https://drive.google.com/drive/folders/1hxeApKtO5aENFclurYF6VkTsA-tJdyJi?usp=sharing>

System Level PCB:

<https://drive.google.com/drive/folders/1hxeApKtO5aENFclurYF6VkTsA-tJdyJi?usp=sharing>

Raspberry Pi Pico Datasheet:

<https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html>

<https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf>

Neo M8N GPS Chip Datasheet:

https://content.u-blox.com/sites/default/files/NEO-M8-FW3_DataSheet_UBX-15031086.pdf

https://content.u-blox.com/sites/default/files/products/documents/u-blox8-M8_ReceiverDescrProtSpec_UBX-13003221.pdf

https://content.u-blox.com/sites/default/files/NEO-8Q-NEO-M8-FW3_HIM_UBX-15029985.pdf

Linear Regulator Datasheet:

<https://www.onsemi.com/pdf/datasheet/ncp59748-d.pdf>

BM78 Bluetooth Module Datasheet:

<https://www.mouser.com/datasheet/2/268/BM78 Bluetooth Dual Mode Module DS60001380-2999436.pdf>

TVS Diode (ESD Protection, Fast ns Response compared to Zener) Datasheet:

<https://www.digikey.com/en/htmldatasheets/production/1062384/0/0/1/aoz8131.html>

GPS Antenna Information:

<https://drive.google.com/file/d/1mcDZ3XW-l5dL1MnoJhqfL61UWaW1ToNe/view?usp=sharing>

6.3 Presentation Materials

COLLEGE OF ENGINEERING

Electrical Engineering and Computer Science

ECE.08



Project Overview

Background

Many Americans drive vehicles as a major part of their occupation. The Internal Revenue Service (IRS) offers incentives for businesses or individuals to report the mileage they have driven throughout the year in order to receive a reduction in their taxes based on that mileage.

MAKE:	MODEL:	YEAR:
DATE	START	END
TOTAL		PURPOSE

Mileage Tracking Form

However, most vehicles do not offer a way to track individual trips, so we often go to:

- manually fill out trip records on a physical sheet of paper
- subscription based mileage tracking devices that ask for monthly fees

These are some hurdles for people who want to report their business mileage.

Summary

Our team is tasked to create a cost-effective mileage tracking device.

We offer:

- no subscription fees to the user
- low component costs
- companion mobile application to access the recorded trip info via Bluetooth
- onboard SD card storage for the user to access the recorded trip data on PC



Vehicle Mileage Tracker

A tool to report business mileage to the IRS, Internal Revenue Services, for tax write off purposes.



Block Diagram of Vehicle Mileage Tracker



Physical Device

HOW THE SYSTEM WORKS

This system records both business and personal trips automatically. Once GPS signal is received, the microcontroller (MCU) is able to determine the motion of the vehicle, and start a trip from there. Once a trip is started, the MCU will log GPS pings every 2 seconds and use them to calculate distance. Each time the microcontroller detects a pause in motion, it will add the calculated distance to the trip record. On the mobile app, we can manually start a trip, as well as adding details to the trip records, such as Business Purpose or Destination. The device has a push button attached for the user to switch modes of operation-Business or Personal. The current mode is indicated by the LEDs on the device, with green for Business and blue for Personal. The microcontroller is powered using a cigarette lighter adapter that provides 5V from the vehicle.



Microcontroller, GPS, and other components



PCB Layout



Mobile App Interface



Recorded Trip Data Example



About the Team

Caleb Lacey
Mobile App, Bluetooth
laceyc@oregonstate.edu

MingChu Chiu
Distance Processing, External Storage
chiu@oregonstate.edu

James Ewing
GPS, Power, HMI, Enclosure
ewingj@oregonstate.edu

Engineering Requirements

The Mileage Tracker system is required to:

- Allow Bluetooth access to trip mileage data through iPhone application interface.
- Keep a small form factor.
- Actively avoid distractions or issues for driver safety.
- Include a guided walkthrough for use of the system.
- Organize data in a neat and understandable manner that is easy for any user to comprehend.
- Track vehicle mileage automatically and accurately, within an error of ± 0.1 miles per 5 miles driven.
- Indicate current trip mode (Business or Personal), and allow users to swiftly switch the mode of recording.
- Utilize vehicle power during operation.

ShowCase Link:

<https://eecs.engineering.oregonstate.edu/project-showcase/projects/?id=0M5L73tckRA0RyRW>

6.4 References and File Links

[1] V. Nova, "Converting IOS apps to Android: Software and design priorities for sustainable innovation," *Heady*, 19-Mar-2021. [Online]. Available:

<https://www.heady.io/blog/converting-ios-apps-to-android-software-and-design-priorities-for-sustainable-innovation>. [Accessed: 28-Apr-2023].

[2] A. Rosebrock, "Recognizing digits with OpenCV and Python," *PyImageSearch*, 22-Mar-2023. [Online]. Available: <https://pyimagesearch.com/2017/02/13/recognizing-digits-with-opencv-and-python/>. [Accessed: 28-Apr-2023].

[3] "Other Global Navigation Satellite Systems (GNSS)," *GPS.gov: Other Global Navigation Satellite Systems (GNSS)*. [Online]. Available: <https://www.gps.gov/systems/gnss/>. [Accessed: 28-Apr-2023].

[4] L. ZHAO and Y. LI, "Identifying origin-destination trips from GPS data – application in travel time reliability of dedicated trucks," *Promet - Traffic&Transportation*, vol. 34, no. 1, pp. 25–38, 2022. <https://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=1134&context=matcreports>

[5] B. Racoma, "Importance and benefits of mobile app localization," *Day Translations Blog*, 14-Apr-2022. [Online]. Available: <https://www.daytranslations.com/blog/why-app-localization/>. [Accessed: 28-Apr-2023].

[6] *Colour Blind Awareness*, 13-Apr-2022. [Online]. Available: <https://www.colourblindawareness.org/>. [Accessed: 28-Apr-2023].

[7] The 10 main benefits of goal-setting, <https://www.goucher.edu/experience/getting-involved/leadership/documents/Goal-Setting.pdf> (accessed May 16, 2023).

[8] "The true purpose of a team meeting (+best practices and tips)," RSS, <https://www.hugo.team/blog/purpose-of-a-team-meeting#:~:text=Team%20meetings%20provide%20a%20space.their%20achievements%20from%20last%20week>. (accessed May 15, 2023).

6.5 Revision Table

4/28/2023	Chiu, Ewing, Lacey - Project Closing Creation
5/12/2023	Added references to teamwork recommendation
5/12/2023	Created Project Artifact Folders