SOLAR POWERED GPS TRACKER

A Design Project Report

Presented to the School of Electrical and Computer Engineering of

Cornell University

In Partial Fulfillment of the Requirements for the Degree of

Master of Engineering, Electrical and Computer Engineering

Submitted by

Stefen Gerrit Pegels

MEng Field Advisor: V. Hunter Adams

Degree Date: January 2023

Abstract

Master of Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

Project Title: Solar Powered GPS Tracker

Author: Stefen Gerrit Pegels (sgp62)

Abstract: This individual MEng project seeks to develop and test a solar-powered fitness wearable, for use in tracking location for purposes of fitness activities. The project includes a circuit board design and layout from scratch with personally selected components and solar cells, as well as custom microcontroller software implemented on the board to achieve its desired functions. Finally, this project includes extensive testing including running exercises performed by the designer to test its GPS navigation system and path tracking. The major outcome of this project is the physical board itself, which can be used in multiple harnesses to fit many different scenarios, as well as GPS path data from multiple tests and an analysis of the effectiveness of the solar cell design in long-term deployment of the system. The components for this design include specific Surface Mount Technology (SMT) circuit board GPS, microcontroller, and solar charging circuit components. This project may seem like a somewhat primitive version of existing fitness wearables such as smartwatches, but it exists as an investigation into renewable energy for small-scale electronics with practical applications in the personal fitness of the designer. Its ability to charge a battery through solar energy will also be useful as it can run for long periods of time without any replacement power source. The first semester of work on this project encompasses a set of preliminary objectives, notably power sizing and solar cell selection, as well as complete component selection. Second semester work involves board fabrication, microcontroller programming, and outdoor field testing.

Executive Summary

This project resulted in a functioning board that is able to power itself off of a combination of solar and battery power, with battery power fully powering the device in non-ideal sunlight conditions. The software flashed onto the microcontroller unit (MCU) was capable of gathering GPS data over satellite connection, processing this data, sending valid messages to an onboard nonvolatile memory storage device, and finally writing data externally with a button for post processing. An onboard battery charging circuit was able to use solar voltage to charge the battery that was delivering current to the microcontroller and memory system.

Minor issues arose during the course of the design cycle dealing with specific components being out of stock at major distributors, delaying the design process and preventing the possibility of a board revision to fix initial errors. Thankfully there were no critical problems with the fabricated board preventing power and function, although there were inefficiencies with opportunities for improvement. The onboard GPS had trouble generating a consistent GPS satellite connection, resulting in an offboard GPS chip used for specific test scenarios. This issue arose due to inadequate spacing for the onboard GPS antenna. The onboard battery charging IC had a minimum voltage rating that was higher than the voltage of the solar cells in cloudy conditions, meaning they could not contribute power unless in partial sun or direct sunlight. A less powerful charging IC would solve this issue. Finally, one pin on the MCU was left floating which prevented the device from loading the program on startup. This was fixed by soldering a resistor to GND from that pin in order to load programs from flash memory.

The success of this project demonstrates the potential for solar energy as a renewable power source for small-scale microcontroller boards, especially those with battery power as an alternative. The size of this board (roughly 4.5x4.5cm) was constrained by the area of the solar cells, indicating that if more area efficient solar cells become available in the future, this design can become even smaller and compact.

<<<Feedback - anything to add here? Requirements were not immediately clear>>>

Design Problem

Inspiration

Smart watches and other fitness wearables have undergone an industry explosion in the past 10 years as handheld technology becomes more accessible and helpful for fitness and health monitoring. Watches are now able to gather data like user heart rate, calorie expenditure, elevation change, GPS location information, and a slew of others. Encapsulating this wealth of information into a small form factor wearable on the arm with a simple user interface has made the technology more popular with casual athletes and fitness hobbyists rather than professionals. I personally have used a FitBit and an Apple Watch for over three years when I go running in order to track my pacing, heart rate, and my running route to view afterwards.

Specific to this project is the on-board GPS system that nearly all fitness wearables have in 2022, a system that gives pinpoint positioning for activities like biking, hiking, and running. Route tracking data has even spread to its own flavor of social media in websites like Strava, where users share the routes they've been on with their friends. In my own experience using the GPS tracking on my FitBit, I have found it somewhat unreliable in its ability to get a preliminary fix on the wearer and have been disappointed in its high power consumption, requiring me to charge my FitBit more often than I would have liked, if only for use for an hour-long run a few times per week.

My initial inspiration for this project came when I was planning on going for a run in the brisk Ithaca weather, excited to track my statistics and time myself, until I found that I had forgotten to charge my FitBit and would have to go without it. If it were possible to create a fitness tracking device that could charge its battery while I was using it (running outside), I would never have to worry about forgetting to plug it in at home. With my own solar powered GPS tracker, I could investigate the feasibility of renewable solar energy in a small package.

Design Requirements

This project plans a fully custom printed circuit board design. The final deliverable for this project is the rectangular board itself, which is isolated into its rectangular form in order to fit a variety of housings/ slotted devices to fit multiple wearers (wrist, ankle, joint, etc).

Hardware requirements for this board are based on the functional requirement of being able to use solar energy to capture GPS data, process this data, and store it in an onboard memory system for later data reading. Basic hardware requirements and their functions are as follows:

Design Part	Purpose
Solar Cell	Harnesses solar energy to power the device and charge the battery.
LiPo Battery	Powers the circuit in situations with insufficient light.
Battery Charger	Excess solar generated current is used to charge the battery in direct sunlight conditions.
GPS	Interfaces with an antenna to connect to Global Positioning Service satellites to gather position, course, and movement data.
Non-volatile Memory Unit	Stores GPS data on device for later analysis, capable to retaining data when power is off.
Microcontroller	Parses GPS data, sends data to nonvolatile storage, provides device status indication, organizes sending of data externally for analysis.

Figure 1: Required Design Components.

Other complementary hardware components are discussed in the **Hardware Design** section, along with physical design and layout decisions and justifications. Many other smaller components were added to work with the major pieces listed in the table above, including power converters, LEDs, capacitors, debug ports, switches, and antennae.

With all of these components, the principal board design constraint will be its size. Ideally, the design would be as small as possible while still being able to ensure proper, consistent functionality of the specified system, at a high level. Since an important use case for this device is use on the wrist, the upper bound for board size would be the width of the wrist (~5cm). This upper bound limits the available surface area for solar cells; adding more cell area increases available power to run the system and charge the battery. Too little surface area for solar cells would result in insufficient power for the system, creating a lower bound of board size for enough solar cell area to power the system by itself. Power consumption metrics are discussed in the **Hardware Design** section.

Initially, other sensors (IMU, heart rate sensor, etc) were planned to be added to the board, but were removed to save power in order to drive the board size down. There is opportunity to add other sensors in the future given power leeway from the solar cells, or if more energy efficient solar cell technology is developed that can take better advantage of available surface area. Likewise, as better solar cells get developed, or perhaps flexible solar cells designs, the board could be compressed into a smaller footprint, or altered into a flexible circuit board

format. A flexible format was considered due to ease of use and form factor, but was deemed impossible due to solar cells only being available in a rigid format.

Another important design decision was the decision to remove an antenna for communication with a ground station. This was made to reduce overall design complexity, while also considering board layout difficulties involved with placing an antenna with solar cells on the reverse side of the board. Most importantly, the designed use case of this device is to track routes while the user has the board, and not necessarily used for active tracking of a lost item, person, etc. For this reason, active remote tracking is not necessary for system function. The nonvolatile storage is used to take record of GPS locations, which can be viewed at a later time by the user.

For the microcontroller, its system requirements are based on the constraints of the system software being able to perform the following tasks:

- Gather GPS data over UART
- Parse GPS data and send to nonvolatile memory over SPI
- Read nonvolatile memory over SPI and send to host over second UART interface
- Monitor system function and change operational mode of GPS, memory, power system

To meet these requirements, the microcontroller is constrained to have enough onboard memory to hold the firmware for accomplishing these functions. Memory allocation specifics can be found in the **Software Design** section. For the device to function properly, correct software must be written and flashed onto the onboard microcontroller that carries out these tasks. A breadboard test setup with breakouts of relevant components (GPS, nonvolatile memory, CPU) is constructed in parallel with assembly and shipping of the completed circuit board, in order to test the code before flashing it onto the actual device.

Hardware Design

Designing the solar tracker started with a logical electrical schematic design. This was started in the first semester as components were selected, and completed in the fall. Schematic design and board layout were created using Altium PCB Designer, using a student license provided free to Cornell students. An explanation of each of the schematic's logical sections follows, with a deep dive into physical layout decisions thereafter.

Schematic I - GPS

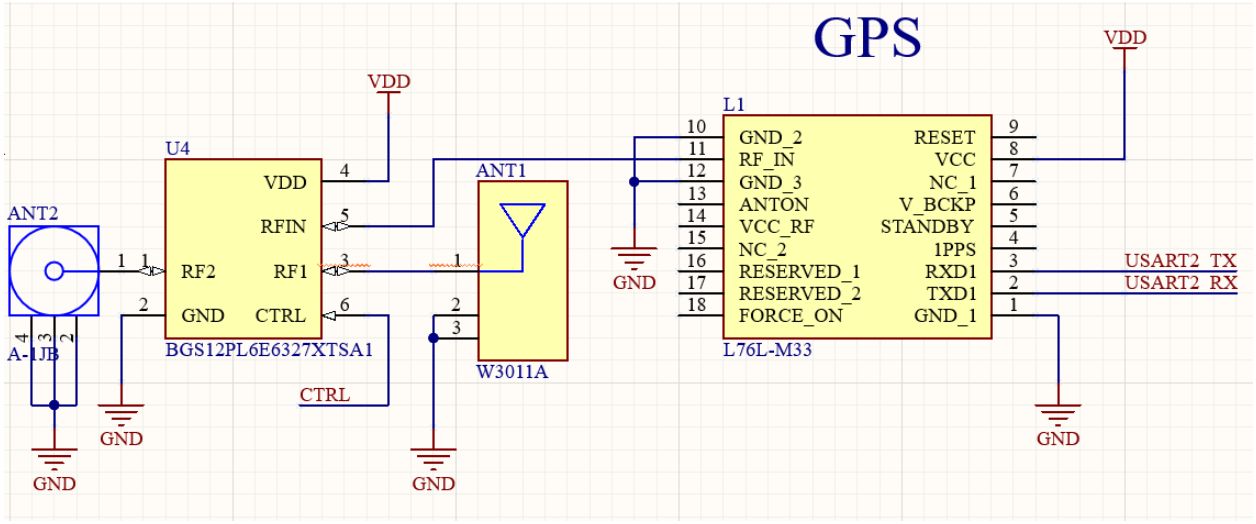


Figure 2: GPS Circuit with RF Switch

Component selection was made before power circuit design, in order to gather preliminary current consumption data to select a sufficiently powerful solar cell. The first component chosen was the GPS, which was settled on as the L76L-M33 [1]. This GPS features concurrent reception of GPS, GLONASS, Galileo, and SBAS signals, making getting a GPS fix much easier. Most important from the datasheet is the metric of 31mA current consumption during GPS acquisition and tracking; the most dominant power consumer on the system. This measurement is critical for solar cell and battery selection.

Visible in *Figure 2*, the GPS *RXD1* and *TXD1* pins are connected to *USART2,* which is the communication interface from which the MCU would decode the GPS NMEA messages [2]. The RF_IN pin on the GPS is routed to an RF switch (BGS12PL6)[3], which allows selection between an SMT antenna on chip (ANT1) and an external COAX port connected to an external patch antenna (ANT2). Both options were added to provide alternate solutions in the event that getting a fix with the built in antenna proved difficult. The switch is controlled by a single output GPIO from the MCU, where a signal of zero uses RF1 and one uses RF2. Unused GPS pins are left floating.

[1] L76L-M33 GPS: https://www.quectel.com/wp-content/uploads/2021/03/Quectel_L76-L_GNSS_Specification_V1.4.pdf
[2] NMEA Messages: https://en.wikipedia.org/wiki/NMEA_0183#NMEA_sentence_format
[3] BGS12PL6: RF Switch Datasheet

Schematic II - FRAM

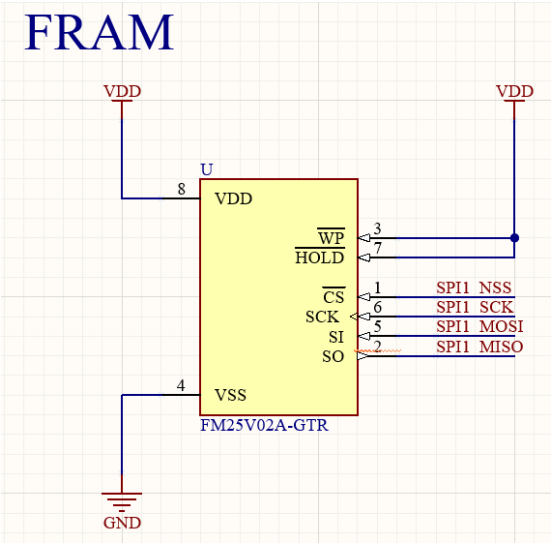


Figure 3: Nonvolatile FRAM Memory Storage

To store GPS data during tracking to read later on, the nonvolatile memory storage selected was the FM25V02A [4]. Nonvolatile storage was needed rather than a volatile memory storage in order to keep data while the device was powered off. The floor for the amount of storage required was calculated based on the use case of tracking GPS positional data for a one hour period (a usual length for a run). During this test scenario, if one NMEA message (approx 80 bytes) would be written to the FRAM every 10 seconds, this would be 360 messages totalling 28kB. Thus the storage chosen was 256kB, the smallest (and cheapest) available for this FRAM product. This chip was also chosen for its 2.5mA current consumption at 40MHz, a frequency that will be decreased due to the low data rate requirements of the system. The FRAM is connected to the microcontroller through the SPI interface, operating as a slave device that will carry out write and read commands as instructed by the MCU. The Write Protect and Hold pins are left unused, and routed to logic level high since they are active low.

[4] FM25V02A: https://www.digikey.com/en/products/detail/cypress-semiconductor-corp/FM25V02A-G/5210380

Schematic III - MCU

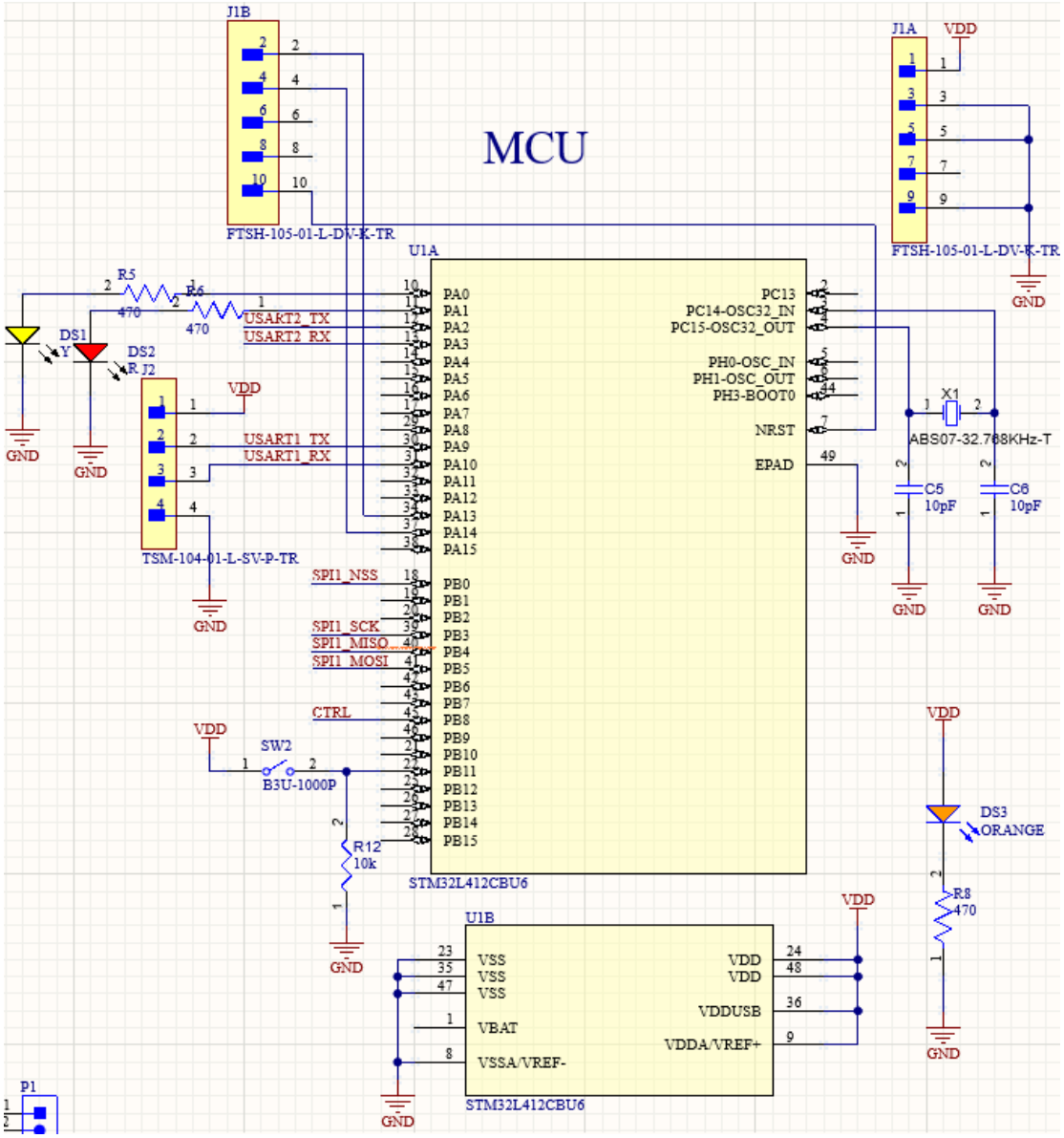


Figure 4: STM32L4 MCU schematic, power connections split to bottom

Building on the basic requirements of the FRAM and GPS, a microcontroller for this system needed UART and SPI functionality to interface with both peripherals. In addition, minimal power consumption coupled with small form factor were primary considerations in the eventual selection of the STM32L412 [5]. With a 7x7mm form factor and a max supply current of 4mA at 32MHz, this MCU was an excellent choice, with the most important design factor being its *availability to be purchased*. Along with many other components on this board, balancing desired functionality with immediate part availability from reputable distributors was a delicate situation. This MCU was initially selected in Spring 2022, but fell out of stock over the summer

[5] STM32L412CBU6: https://www.st.com/resource/en/datasheet/stm32l412kb.pdf

after which a smaller MCU was chosen with less programmable flash until the STM32L412 magically appeared back in stock in October 2022, just in time before the boards were ordered.

The MCU features 48 pins, separated into two 16-pin GPIO partitions PA and PB. The other 16 pins have specific functions relating to power, ground, external clocks, and reset. Important MCU functions are listed below, with their relevant pin connections illustrated in *Figure 4*.

MCU Power

Illustrated by block U1B in *Figure 4*, the MCU is powered by a 3.3V supply, and foregoes any other alternative power options for simplicity. Thus other power domains like VDDUSB or VDDA are shorted to VDD (3.3V), as specified by the datasheet. The VSS pins, or GND pins, are connected to the common ground of the system.

SPI

Serial Peripheral Interface (SPI) on the MCU, used to interface with the FRAM data storage, utilizes *SPI1* on pins PB0/PB3/PB4/PB5. These GPIOs correspond to Chip Select (CS), Serial Clock (SCK), Master In Slave Out (MISO), and Master Out Slave In (MOSI), respectively. Details about the setup of SPI1 within the MCU can be found in the **Software Design** section of this report.

USART2

Universal Synchronous/Asynchronous Receiver/Transmitter (USART) is used to communicate with the GPS module on the board, and utilizes PA2 and PA3 for *USART2_TX* and *USART2_RX*. Since the MCU allows configuration of multiple UART ports in multiple places due to the GPIOs having moderately configurable functionality, pin positions for USART as well as SPI and others were chosen to put them in a convenient location to route to their corresponding peripheral (see **Hardware Design - Layout** for physical layout).

USART1

A second USART is used to communicate with the external host device to transfer GPS data off board for analysis and processing. This function operates with the use of a button, connected to PB11. This button, when pressed, is read as a GPIO input HIGH, signaling to the MCU that data should be read from the FRAM over SPI and sent over USART1 to the external 4-pin header connected through PA9/PA10. This 4-pin header is used to connect to a USB-UART cable that allows the GPS data to stream into an external machine.

External Oscillator

A 32.768MHz crystal oscillator is connected to PC14/PC15, dedicated pins for an external 32MHz input. This oscillator is used as the system clock as a more stable alternative to

the internal RC oscillator, which has less precision for the system needing faster data rates for gathering GPS data and communicating over SPI.

Debug

In order to program the device, a 10-pin JTAG header is placed, represented in *Figure 4* as J1A and J1B. This header utilizes the SWDIO [6] programming interface, and connects directly to PA13/PA14 on the MCU. This JTAG header is connected to the host programming computer through a USB-SWDIO breakout, and is used to both flash code onto the board and do line-by-line debugging.

LEDs

The MCU portion of the board also has 3 LEDs, with two set up on PA0 and PA1 as status LEDs to be used for important board status information such as successful GPS fix, SPI transaction, or outbound UART data transfer. The third LED is connected to VDD to signal successful operational device power.

Schematic IV - Battery Charger

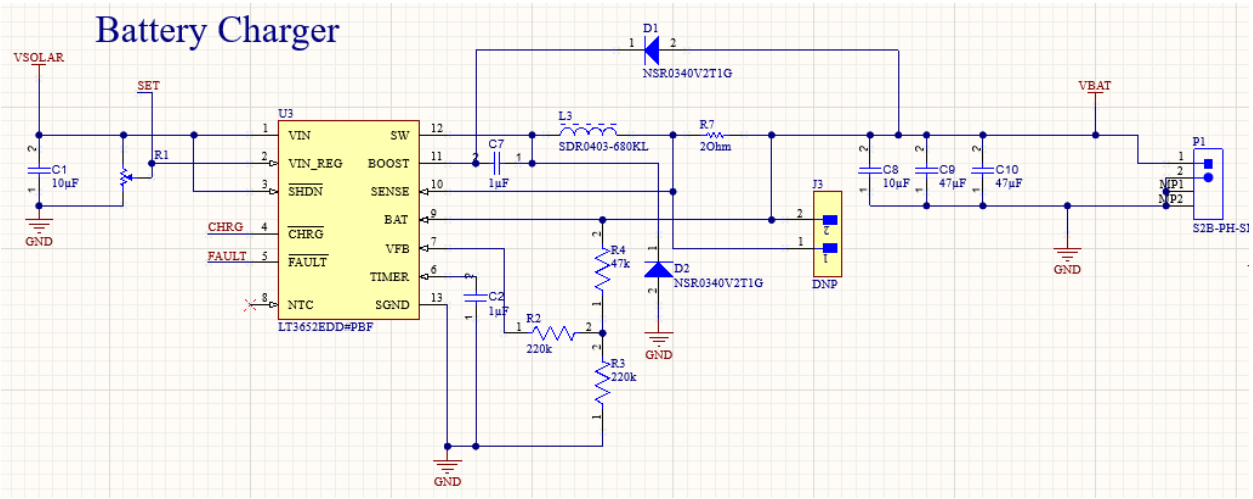


Figure 5: LT3652 Solar Battery Charging Circuit

To power the design, an early design component selected was the battery charging circuit, which was mostly taken from a Sparkfun Solar Charger [7]. This component utilizes an LT3652 battery charging IC[8], which takes a solar cell voltage as an input with an adjustable potentiometer. The resistor divider circuit on the output fixes the charge voltage to a constant 3.7V, the voltage of most common LiPo batteries used as backup power. The output voltage line SW has a series inductor and resistor that dictates the charge current. A resistor of 2Ω (R7) was used to set the default charge current as 50mA, which can be further increased by adding a

[6] SWDIO: https://developer.arm.com/documentation/101761/0100/Debug-and-trace-interface/Serial-Wire-Debug--SWD--signals
[7] Sparkfun Sunny Buddy: https://cdn.sparkfun.com/datasheets/Prototyping/SunnyBuddy-v13a.pdf
[8] LT3652: https://www.analog.com/media/en/technical-documentation/data-sheets/3652fe.pdf

parallel resistor to the jumper connection J3, given the following charge current equation: I_{chg} = 0.1 / (R7 || R_{J3}). Various decoupling capacitors are added, and VBAT (3.7V) shoots off from the line that connects to the P1 battery charging port to make its way to power the load.

The output of the battery charger carrying solar current is combined in series with the output from the battery, allowing the battery to assist in powering the load when the solar current is insufficient. In a similar manner, with excess solar current than is required to power the load through VBAT, the current will trickle into the battery positive lead and charge the battery. This design was chosen rather than a strictly solar powered approach to avoid instances where the system power would fail in the absence of sufficient solar power.

Schematic V - Buck Converter

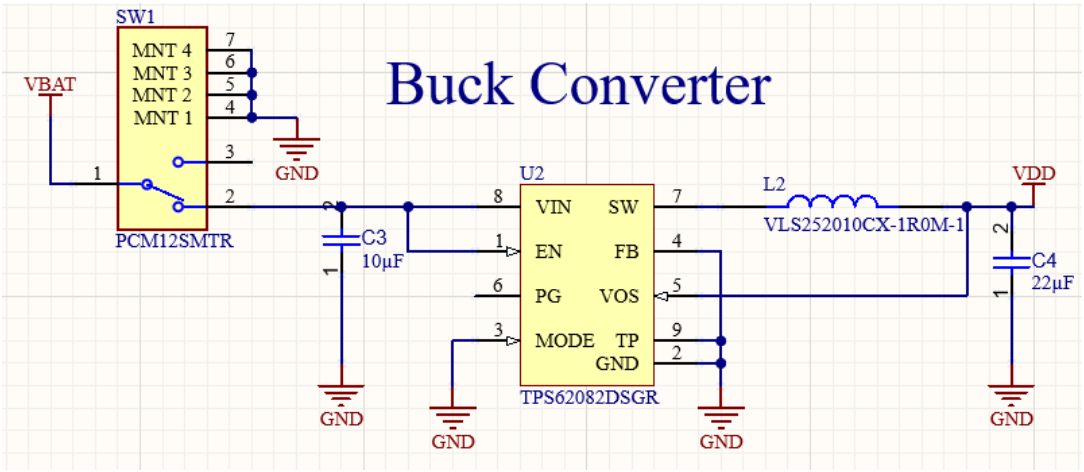


Figure 6: 3.3V Buck converter circuit including SW1 power switch.

In order to power the MCU-peripheral system laid out in Schematic sections I-III, a reliable source of 3.3V power is required to allocate to the VDD power plane. As indicated in the battery charging circuit, VBAT is fixed at 3.7V, so a step down buck converter [9] is used to fix the voltage at 3.3V to protect the MCU and related circuitry. Buck converters come with a cost of efficiency, usually dropping 10% of input power for step down functionality. This design decision was taken into account when considering solar cell and battery power sizing, and is necessary to avoid power issues attempting to power the device at 3.7V, which is above the 3.6V maximum rating for onboard devices such as the FRAM. Unused pins on the buck converter IC are shorted to GND or left floating, per the spec sheet. Visible in *Figure 6,* A switch is used to control the flow of VBAT into the buck converter circuit. This acts as a "power switch" for the entire device; while the battery is able to be plugged and unplugged from the board, the solar cells remain fixed. This switch prevents the solar cells from constantly attempting to power the board when operation is not desired. If the battery remains plugged in, this switch in the OFF position still allows for passive charging of the battery (at an increased rate without the circuit load).

[9] TPS62082 Buck Converter: https://www.ti.com/product/TPS62082/part-details/TPS62082DSGR

Schematic VI - Solar Cells

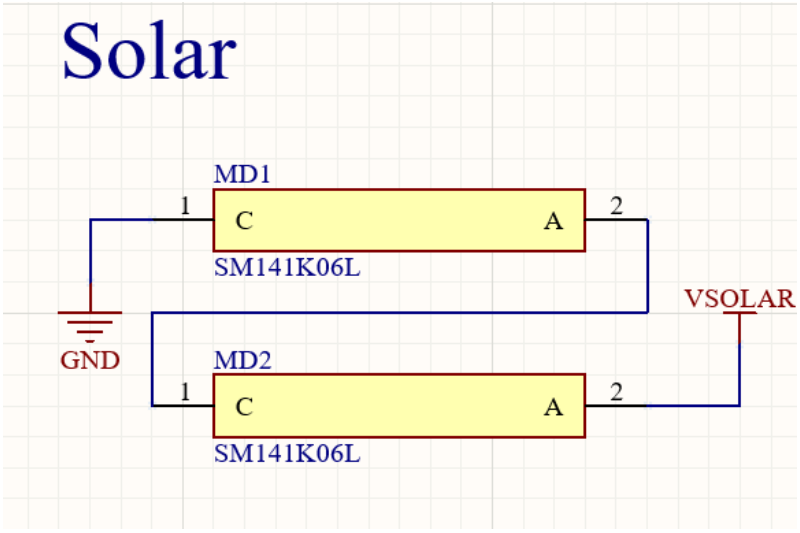


Figure 7: Solar cell connection in series.

The solar cells selection and organization was decided on after selecting relevant SMT components, in order to get a rough power estimate to select a solar cell system that provides ample power to charge the battery and the circuit. Initially, one large solar cell was considered, which would determine the board outline with the solar cell on one side and all of the ICs on the other. However, upon investigating the LT3652 solar charging circuit, an input voltage minimum of 4.95V was required. To accomplish this, and retain the battery charging design from SparkFun, two identical solar cells were wired in series to provide a voltage threshold above 4.95V and extra output current (at max power point) to reduce risk of current drops and strain on the battery. The solar cell selected for this purpose was the 184mW SM141K06L [10]. At max power point, each solar cell produces 4.35V and 55mA, meaning just one might have been enough to power the load all by itself, with no converter. However, the robust battery backup allows for more consistent operation and function in non-ideal light conditions, with a series combination of two cells at 8.7V (ideal) and 55mA providing more than enough power to support device operation. This extra solar power also helps mitigate any efficiency loss from the LT3652 battery charger and buck converter, as well as small current draw from LEDs and resistors not viewed with such scrutiny before ordering the boards.

Physical Layout

After the schematic was completed with no Electrical Rule Checker Violations, the physical board layout began in Altium PCB Designer. The physical layout would determine where the components were placed, how they were routed together, as well as other physical parameters like board thickness, keep-out spacing for sensitive RF components, power and

[10] SM141K06L: https://www.digikey.com/en/products/detail/anysolar-ltd/SM141K06L/9990462

ground plane distribution, and others. After the physical design was complete and passed Design Rule Checker with no errors, output Gerber files and a component Bill of Materials were generated for fabrication, to be discussed at the end of this section.

Layout I - Board Outline & Solar Cells

The 2 SM141K06L solar cells were the largest components for the system, and thus drove the design decision for the overall board area. Since the solar cells needed to be on their own side of the board, all other components needed to be placed on the underside of the PCB. Thus the board outline was rectangular in nature, with the length of one solar cell and the width of two solar cells arranged next to each other. A small buffer region was inserted around the solar cells to prevent them from damage to the sides of the board. This brought the board dimensions to 4.8 x 4.3cm.

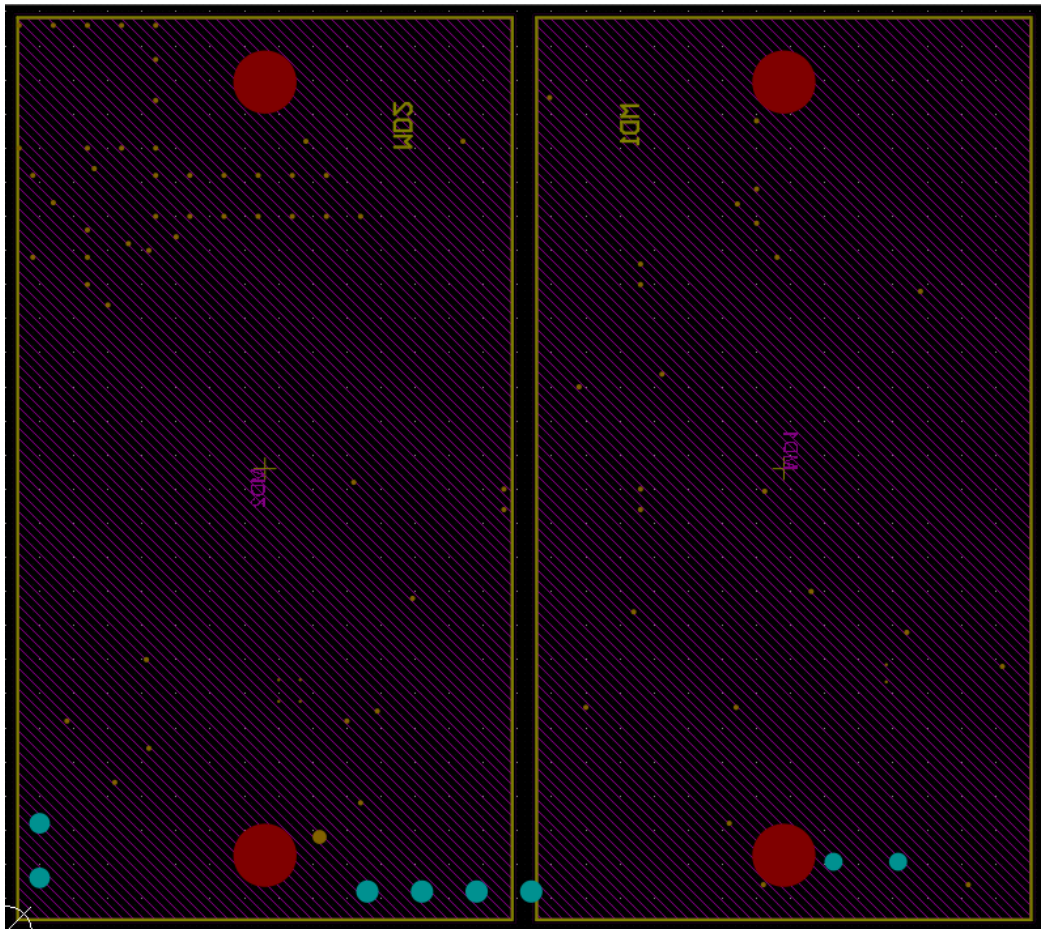


Figure 8: Board Outline with SM141K06L Solar Cell Footprints.

Layout II - Battery Charger

Each of the other schematic sections were organized logically near each other on the board, to minimize copper route length since nearer logical sections had close connections with each other. The most dense of these sections was the Battery Charger, located at the bottom left

corner of the board, which takes the solar voltage (VSOLAR) as input to the LT3652 battery charger (U3), and sends output to the battery connector (VBAT). Decoupling capacitors and resistor networks are connected to minimize trace length. Traces that directly connect to VSOLAR or VBAT have wider trace widths to account for larger amounts of current, even though the minimum trace width visible on the schematic (10 mil = 0.254mm) is rated for 1 Amp, which is more current than expected in any path on this circuit.

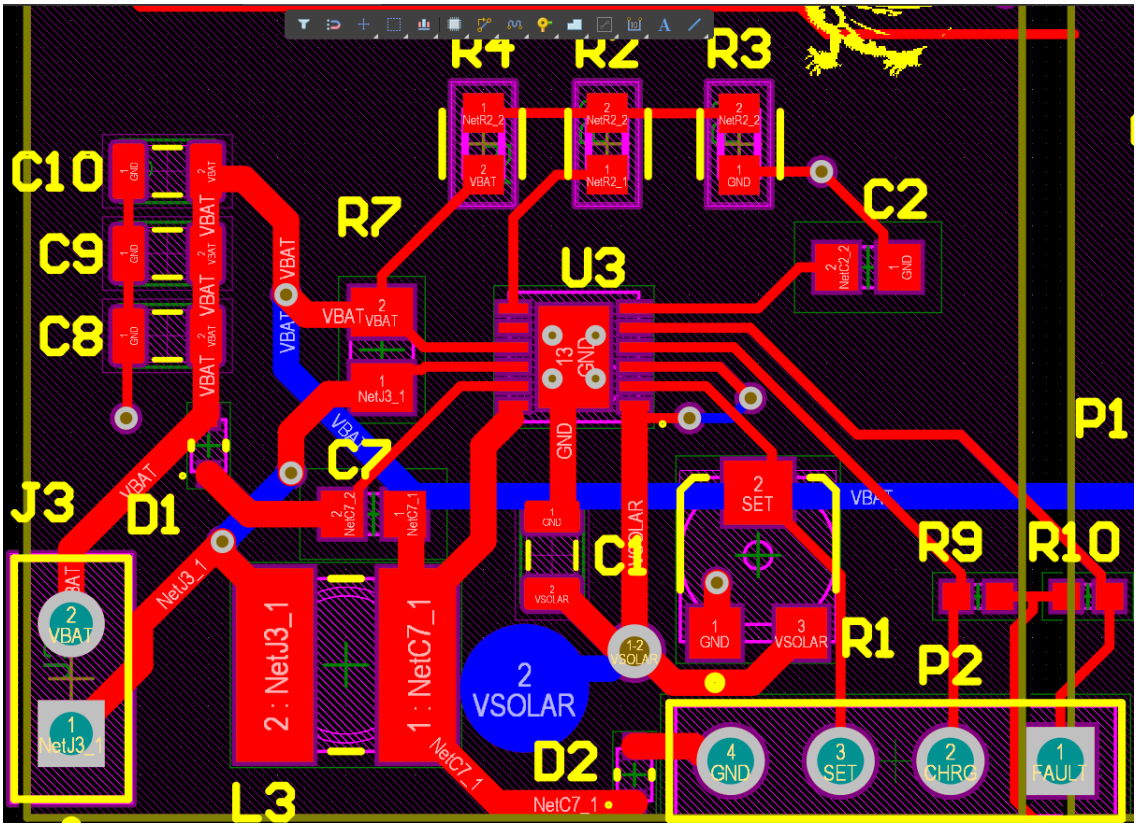


Figure 9: Solar charger circuit physical layout.

A generated 3D view of the solar charging circuit with placed components is shown in *Figure 10.* Visible is the inductor L3 and potentiometer R1, used in the charger circuit. The header pins shown were not placed in the final design and are used as test points for relevant signals.

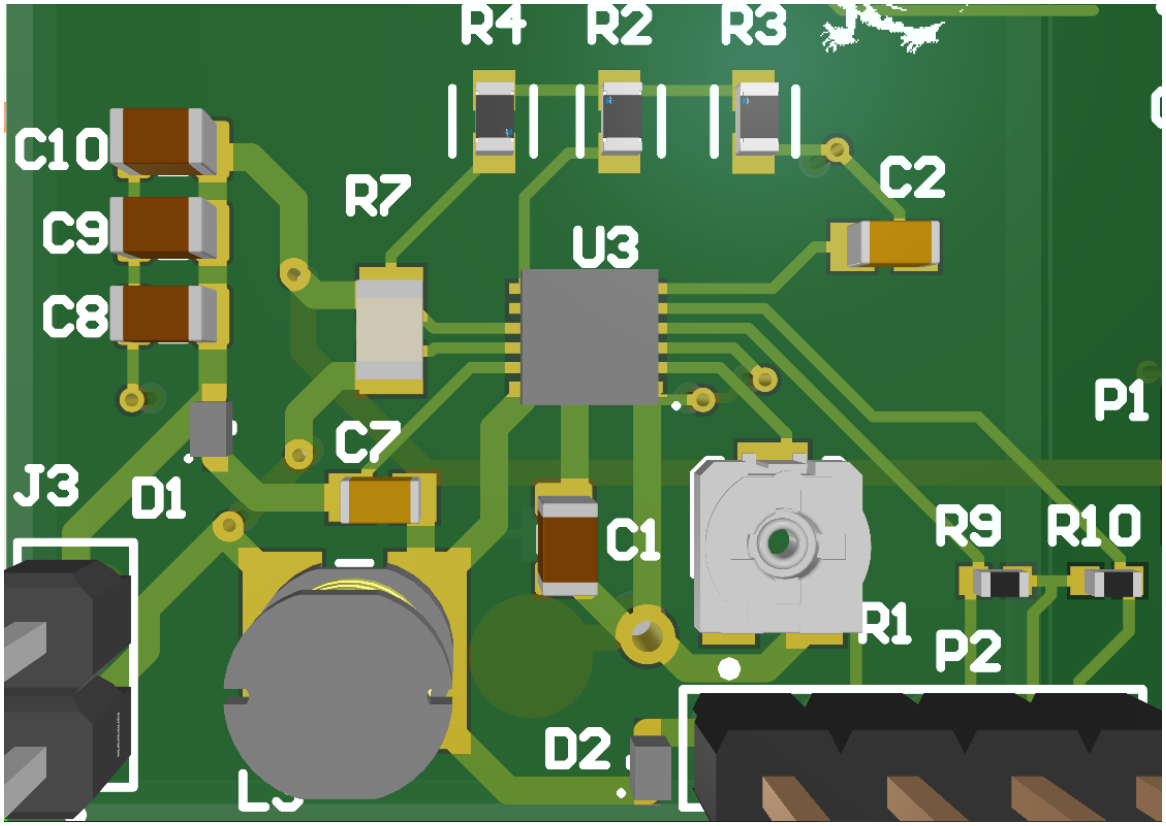


Figure 10: Generated 3D view of Solar Charger board group.

Layout III - Buck Converter and Battery

Directly to the right of the battery charger group of components resides the buck converter group, which is in charge of regulating the 3.7V battery output to 3.3V to power the VDD plane used by the microcontroller, GPS, debug, and memory systems. This section also features the battery jack for connection with the LiPo and the power switch that prevents the circuit from running (SW1). There is also an orange LED (DS3) used to indicate that the board is successfully powered. The buck converter (U2) takes the voltage input from the switch and outputs directly to the power plane.

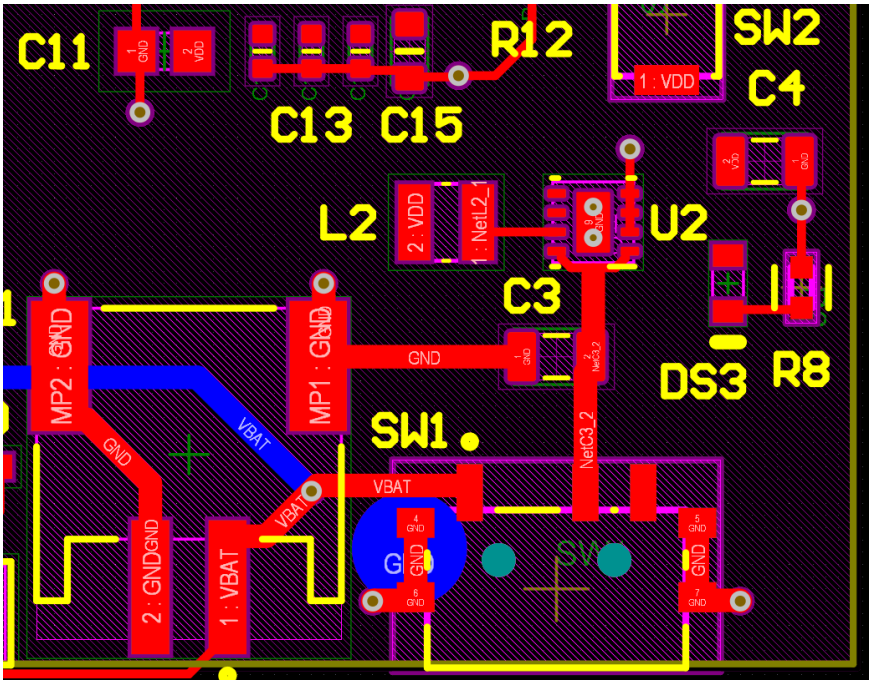


Figure 11: Buck converter(U2) board routing region with battery port (bottom left) and power switch (SW1, bottom right).

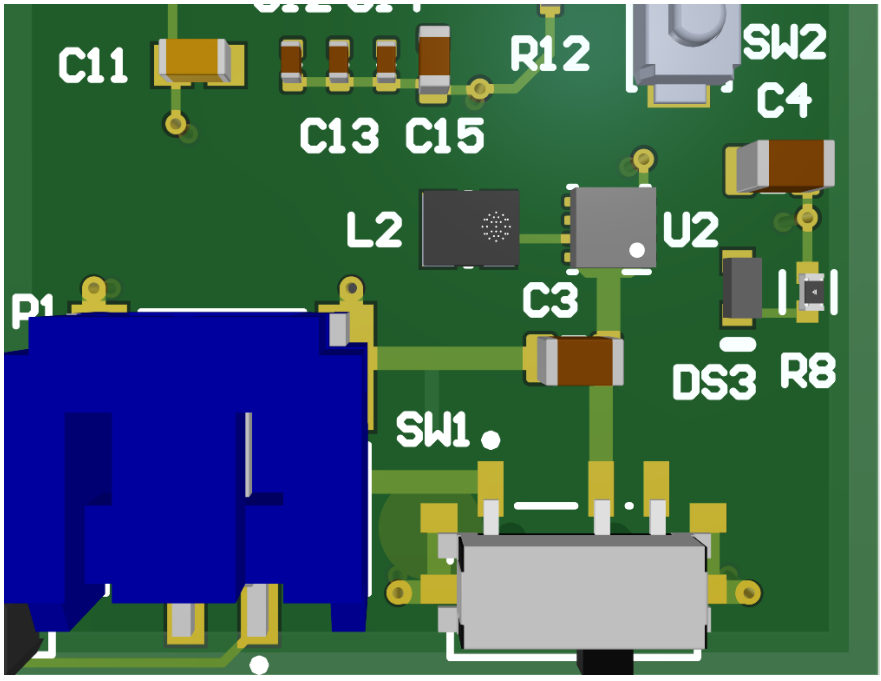


Figure 12: Mock 3D layout for buck converter group, showing battery connector(blue), power switch(bottom) and buck converter(U2).

Layout IV - MCU

Situated above the buck converter is the MCU layout region, with the STM32L412 microcontroller in the middle (U1). Relevant components follow in a counterclockwise manner. Directly to its right is a 4 pin header J2 used to interface with a host device through UART. J1 is the 10-pin JTAG header, used for debugging over SWD and flashing code onto the MCU. U is the FRAM, which the MCU interfaces with over SPI and writes GPS data for permanent storage. X1 is the crystal oscillator used as an optional Low Frequency precision clock source. DS1 and DS2 are two LEDs used for status indication, specifically for a GPS fix and to indicate successful SPI write to the FRAM. C12-C16 are decoupling capacitors used for current regulation on the power plane, and SW2 is a push button used to initiate UART data transfer over the J2 pins.

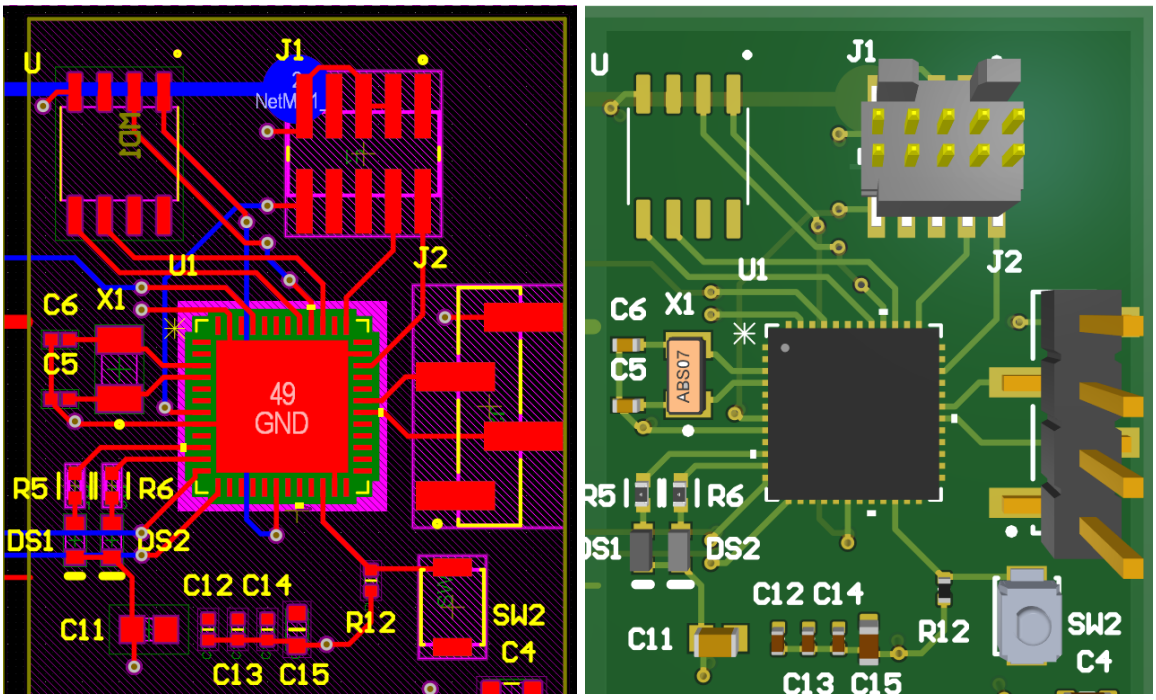


Figure 13: Microcontroller layout group showing routing (left) and mock 3D generation of placed components (right).

Layout V - GPS

To the left of the MCU sits the GPS unit (L1), with two nearby antennas (ANT1 & ANT2) and an RF switch (U4). Since RF components are usually very electrically sensitive, this area of the board is the most sparse, with ground holes added in accordance with manufacturer recommended layout and spacing specifications for the onboard antenna ANT1. ANT2 is a port for an external antenna, added as a backup option in case there are connectivity issues with ANT1. The switch U4 is used to select which antenna to use. The GPS is connected to the MCU through two UART wire connections, visible on the bottom right. Another notable feature of the GPS region is that the traces to the antennas and the RF Out pin of the GPS are 50 Ohms, as

decided by the width of the trace in relation to overall board dimensions and thickness. 50 Ohm traces are preferred for RF designs per manufacturer specification.

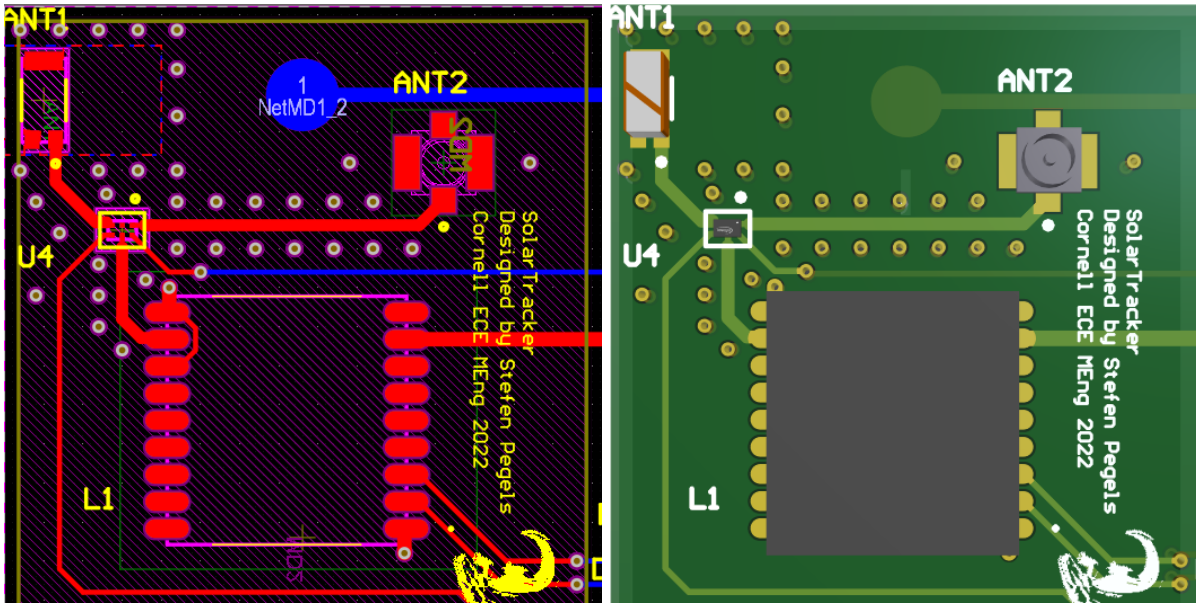


Figure 14: GPS board group, showing trace routing (left) and mock 3D placement(right).

Layout VI - Power and Ground Planes

The final notable layout design choice is the location of the power and ground planes, used as polygon pours to minimize routing for VDD and GND. On the top layer (visible in *Figure 15,* a GND plane exists on the top left region for RF shielding purposes for the GPS system. On the right is the VDD plane, used in the region of the MCU since all components are powered by 3.3V in that section. There is no need for a power plane in the bottom left (charger region) or bottom right (battery region) since those operate in different power domains (VSOLAR and VBAT, respectively).

On the bottom layer of the board, a GND plane is poured throughout the entire surface, except for a cutout region in the top left per GPS antenna ground plane placement specifications. This plane allows for easy grounding of any top layer components, as a Via can be inserted from almost anywhere to connect a top layer pad to GND. Cutting through this plane are various traces from the top layer that had segments on the bottom layer to avoid collisions with other top level traces. Since the two solar cells take up all of the surface area, the bottom side of the board is incredibly sparse, allowing a lot of freedom with traces and vias.

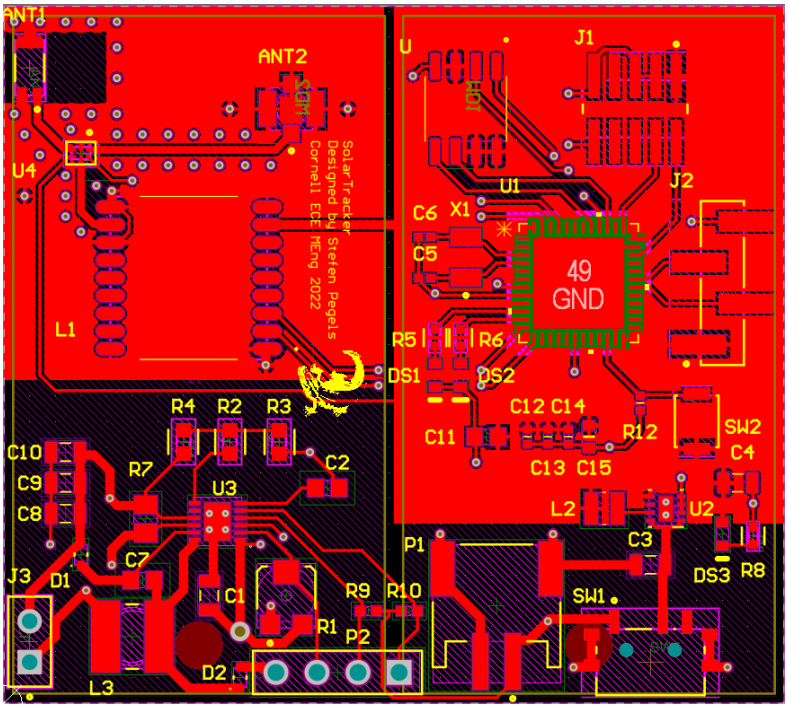


Figure 15: Top layer pours, with a GND plane on the left and VDD (3.3V) on the right.

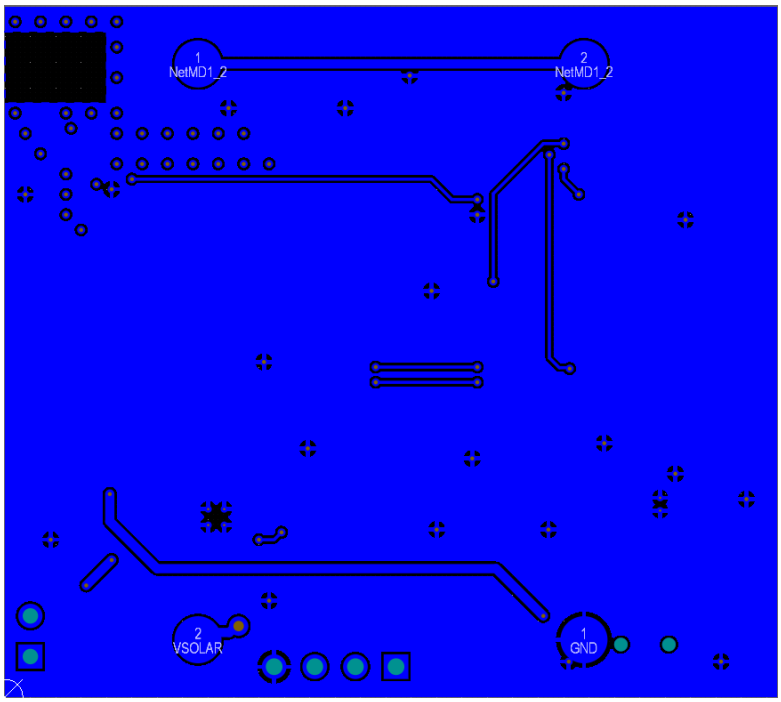


Figure 16: Bottom layer GND pour.

Software Design

Software for this system was designed to run on the embedded STM32L412 microcontroller, and was programmed and debugged with the Eclipse-based STM32CubeIDE. This IDE was selected for its unique advantage of a GUI for system pin and software configuration, which would automatically include drivers for peripherals used, and generate initialization code for each software function. This made experimenting with new features and getting accustomed to the STM32 and FreeRTOS APIs much more streamlined.

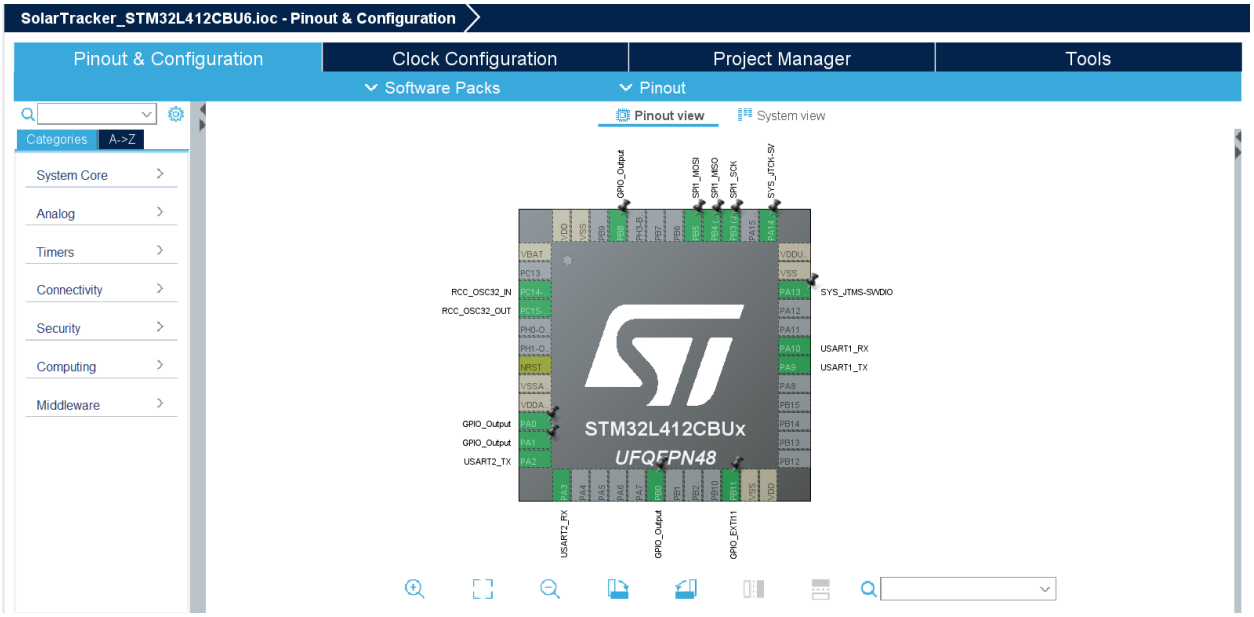


Figure 17: .ioc file for SolarTracker project, showing pin assignments for all GPIOs and board peripherals.

FreeRTOS

The software for this project is organized as a Real Time Operating System using FreeRTOS[11], an open source API with a suite of functions for task management, preemption, interrupts, and mutual exclusion primitives. In an RTOS, code is organized into tasks that run on the single-core CPU for fixed time intervals and pass execution to others. To accomplish the minimum software functionality of this system, a task is constructed for each relevant board function, restated here:

- Gather GPS data over UART
- Parse GPS data and send to nonvolatile memory over SPI
- Read nonvolatile memory over SPI and send to host over second UART interface
- Monitor system function and change operational mode of GPS, memory, power system

In order to prevent a single task from looping through itself forever, and to create a system where some tasks run less frequently than others, 3 binary semaphores exist within the

[11] FreeRTOS: https://www.freertos.org/

system that are posted and taken by various tasks. A flow chart of the tasks with their core functions and semaphore interactions follows, with a table explaining the semaphores in more detail below.

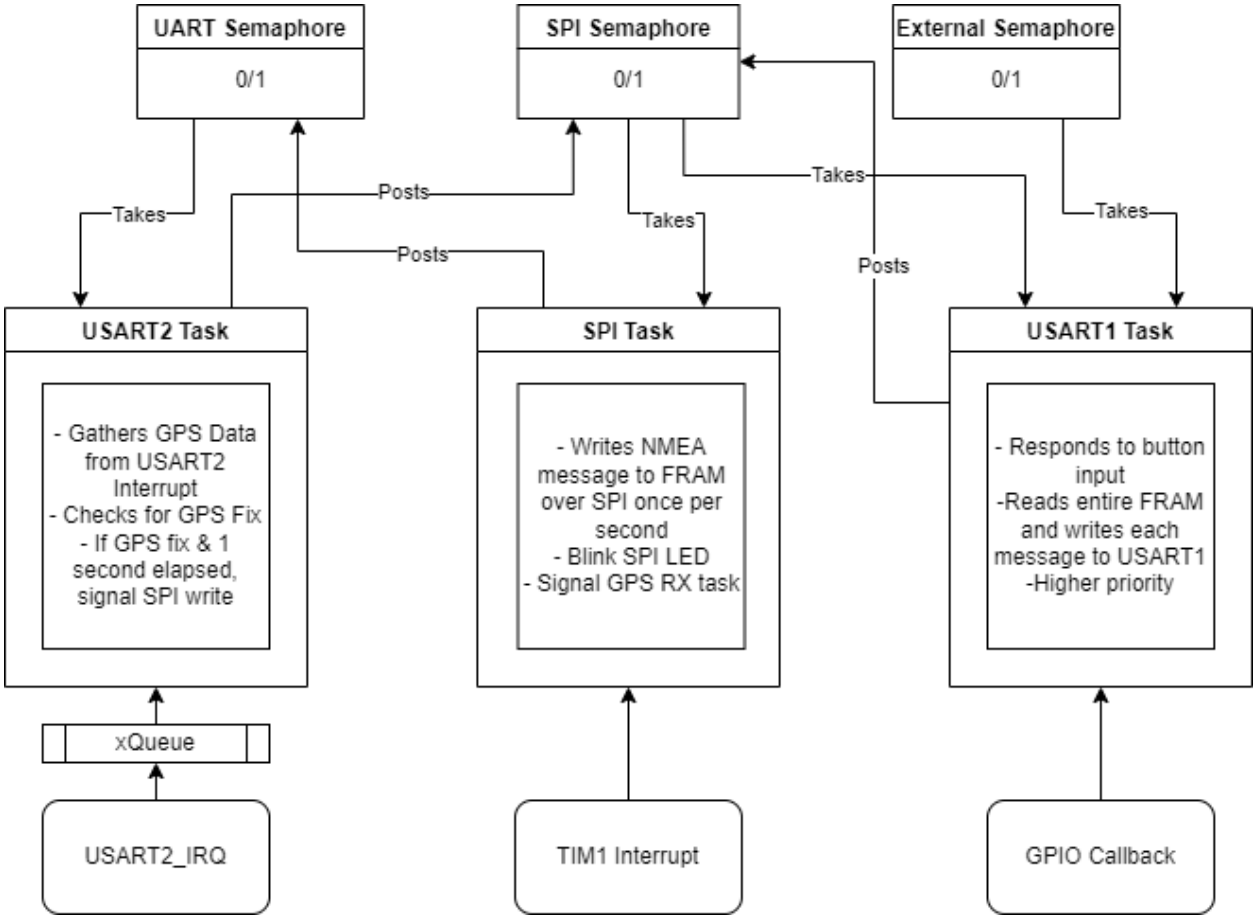


Figure 18: FreeRTOS task organization for software, with 3 binary semaphores used for task execution passing and interrupts/callbacks for each task.

Semaphore	Taken By	Posted From	Purpose
UART Semaphore	GPS Reading Task (usart2_task)	SPI Write Task (spi_task)	Controls execution of GPS reading task using USART2 RX interrupts
SPI Semaphore	SPI Write Task, Export Data Task (usart1_task)	GPS Reading Task, Export Data Task	Controls execution of SPI writing task to FRAM, ensures at most one task reading or writing to FRAM at a time.
External Semaphore	Export Data Task	GPIO Button Callback	Signals data should be exported, which reads from FRAM over SPI and writes over USART1

Figure 19: Semaphore function table explaining which tasks post(increment) and take(decrement) the binary semaphores, along with the semaphore's general purpose.

GPS Reading Task

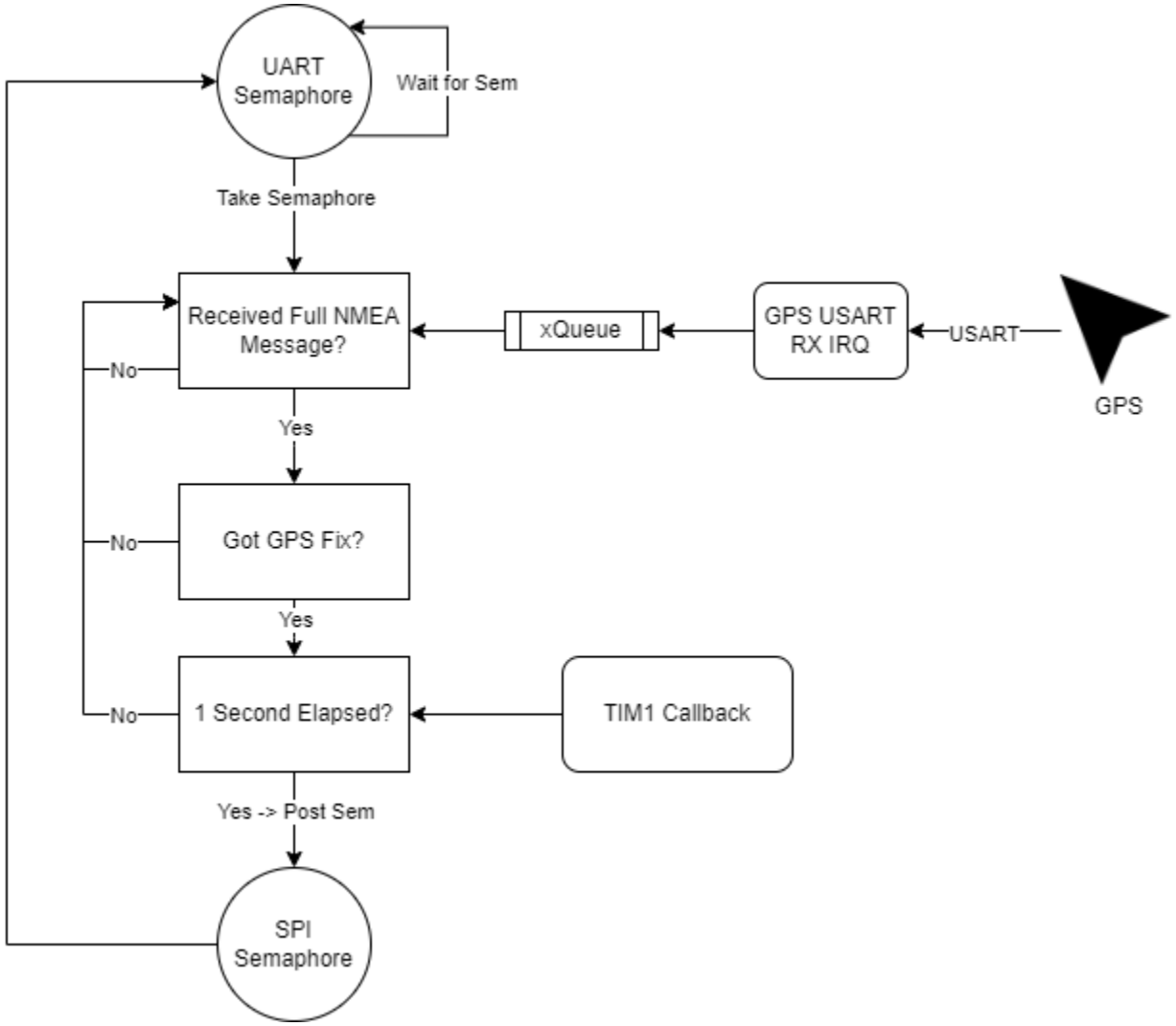


Figure 20: GPS Reading Task flow diagram.

The GPS read task is in charge of receiving NMEA messages from the onboard GPS, which is connected to USART2. An RX interrupt is set up on USART2, whose handler is called for each byte received. This IRQ has a static local buffer to build an NMEA sentence, and checks for UART errors(overrun, noise, framing) with each received byte. If the received character is not a newline '\n' or carriage return '\r', it is added to the buffer. Once an end-of-line condition character is found, the buffer is copied to an xQueue, a FreeRTOS data structure designed to pass data between ISRs and Tasks.

The GPS read task first takes the UART semaphore, then spins until a flag is written that data has been written to the xQueue from the ISR. The thread then gathers the NMEA message header and checks to ensure it equals "$GPRMC", the header for the NMEA message that displays data in the following format. If the header does not match, the message is discarded.

GPRMC NMEA Message Specification				
Num	Structure	Description	Format	Example
1	$GPRMC	Log header		$GPRMC
2	utc	UTC of position	hhmmss.ss	144326
3	pos status	Position status (A = data valid, V = data invalid)	A	A
4	lat	Latitude (DDmm.mm)	llll.ll	5107.001774
5	lat dir	Latitude direction: (N = North, S = South)	a	N
6	lon	Longitude (DDDmm.mm)	yyyyy.yy	11402.32916
7	lon dir	Longitude direction: (E = East, W = West)	a	W
8	speed Kn	Speed over ground, knots	x.x	0.08
9	track true	Track made good, degrees True	x.x	323.3
10	date	Date: dd/mm/yy	xxxxxx	210307
11	mag var	Magnetic variation, degrees	x.x	0
12	var dir	Magnetic variation direction E/W Easterly variation (E) subtracts from True course. Westerly variation (W) adds to True course.	a	E
13	mode ind	Positioning system mode indicator, see Table	a	A
14	*xx	Check sum	*hh	*20
15	[CR][LF]	Sentence terminator		[CR][LF]

Figure 21: GPRMC message specification. Each message is 47 bytes in length.

If a header match is encountered (meaning a correct NMEA message has been received, the position status field is checked (num 3, or byte 18 in the array). If this field is V, no fix has been acquired, and the task keeps running and collecting messages. If a fix has been acquired, this thread posts the SPI semaphore to signal the SPI thread once per second to write the current position to FRAM nonvolatile storage. The thread is only signaled once per second to avoid filling up the FRAM too quickly with redundant information. The system keeps track of time using a global timer interrupt running at 1kHz that increments a global tick variable.

SPI Writing Task

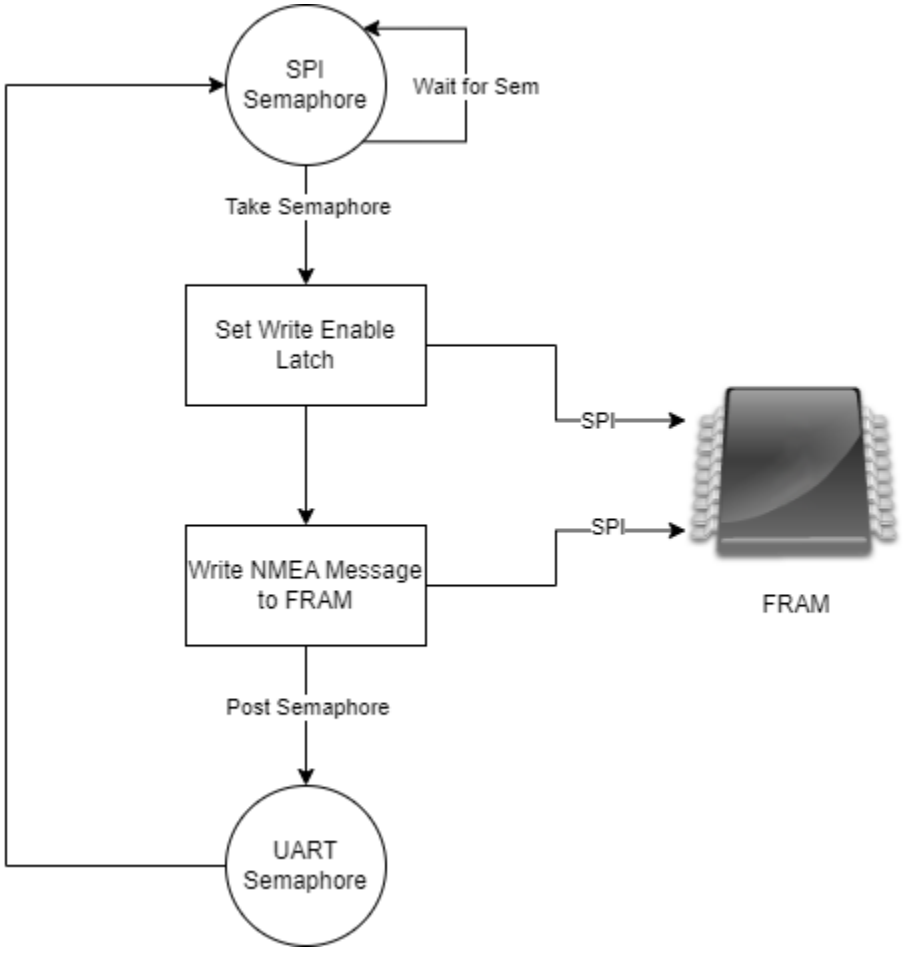


Figure 22: SPI Writing Task Flow Diagram.

The SPI write task is designed to write a complete NMEA sentence (71 bytes) to the onboard FRAM storage once per second in situations where a GPS fix has been acquired. Depending on the frequency of GPS messages read by the MCU it's possible that more than one valid NMEA message will arrive every second, so the SPI semaphore is only posted once per second from the GPS reading task. The SPI write frequency of once per second ensures that the device can gather over one hour of NMEA data before running out of the 256KB storage on the FRAM.

At the beginning of program execution, the SPI task echoes 3 bytes of test data to the FRAM and reads the status register, checking for error codes. In the main loop, the task waits for the SPI semaphore, sets the Write Enable latch to enable WRITE commands, then writes 72 bytes of the GPS task NMEA message buffer to the FRAM. A diagram of an FRAM write procedure is shown in *Figure 23.* A GPIO output pin is used as Chip Select (CS) and manually driven high or low, as recommended by the MCU manufacturer. Even though the NMEA messages are 69-71 bytes, an 80 byte offset is used for padding and an even multiple of 8 bytes and for strict delineations between NMEA messages for debugging purposes. After writing to the

FRAM, the task posts the GPS reading semaphore to allow that task to continue executing. For status information, this task blinks an onboard LED after every successful SPI write, for indication of system functionality when looking at the board visually. The blink is accomplished through the use of the vTaskDelay() FreeRTOS command, which delays the task for 200ms for a blink.

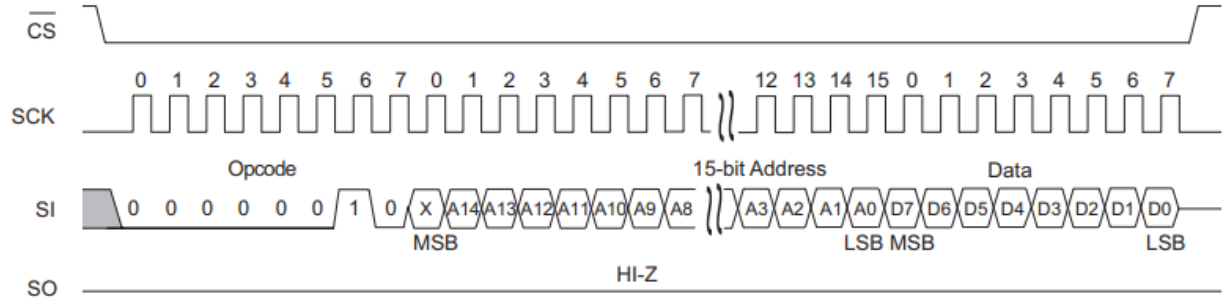


Figure 23: FRAM memory write operation. CS is pulled low as a clock signal is introduced. On the Slave In line (SI), the 1B WRITE opcode is written followed by a 2B address followed by data. Bringing CS high signals the end of the transaction.

Export Data Task

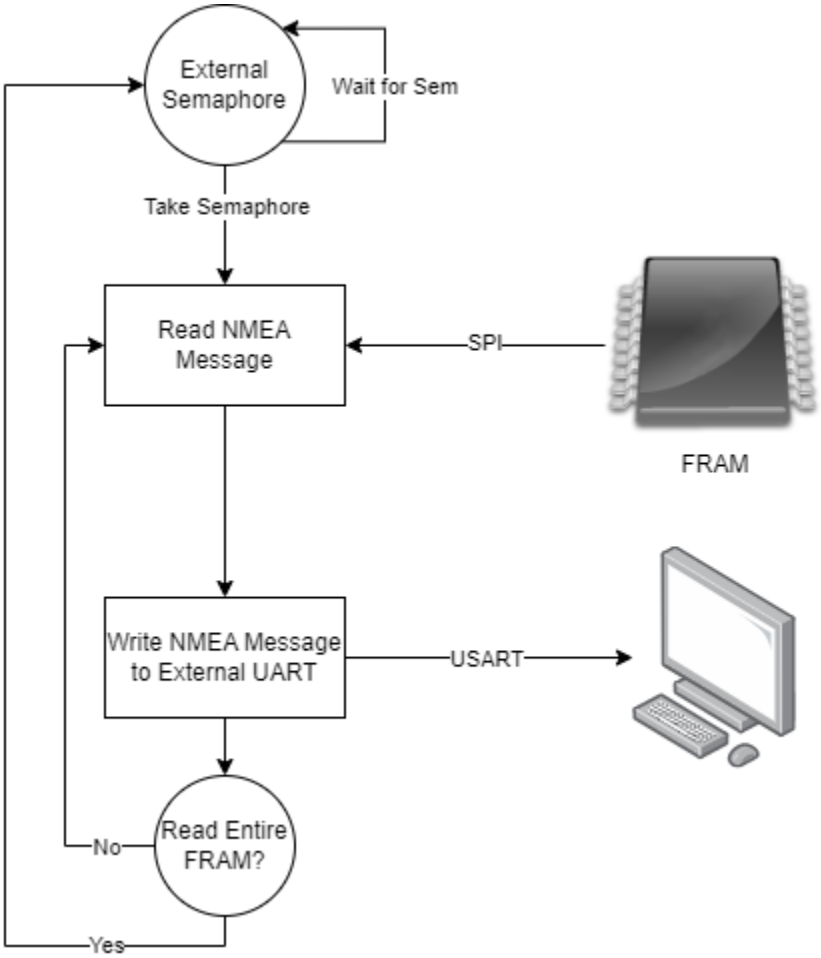


Figure 24: Export Data Task flow diagram.

The third task in the firmware is the export data task over USART1, that moves data from the onboard FRAM to an external host computer for data collection and processing. This task relies on input from a push button to post its semaphore, after which it runs at a higher priority than all other tasks until it finishes. It also disables interrupts during its execution to turn off the RX interrupt on USART2. This was mostly used during board testing using an external GPS board for faster fix times, as this used USART1 for the GPS RX and also the export TX.

The goal of this task is to exhaust all of the onboard FRAM data and parse it on a host computer using a Python script, so this task's main loop runs through the entire FRAM address space (which might not always be filled). After reading 80 bytes of data (one NMEA message), these bytes are copied into a local task buffer and then written over UART. Description of the external UART connection to host computer and off-board data processing and storage can be found in the **Testing** section. After looping through and reading the entire FRAM, this task re-enables interrupts and goes back to sleep while waiting for another button input.

Testing

This design project was a dual software and hardware adventure, with each side requiring its own testing to ensure system functionality and find any problems. Since the hardware design included a lengthy schematic and physical design portion and fabrication time, no hardware testing could be performed until the final boards were delivered. In a similar vein, the system software could not be tested until the physical boards arrived, but it could be tested with similar components put on breakout boards tested in a breadboard fashion to "simulate" the hardware communication between specifically the MCU, FRAM, and GPS system. This test setup allowed the software to be built and tested in small pieces, for example only the MCU receiving data from the GPS or sending test values to the FRAM. An example image of this breadboard setup is shown below.

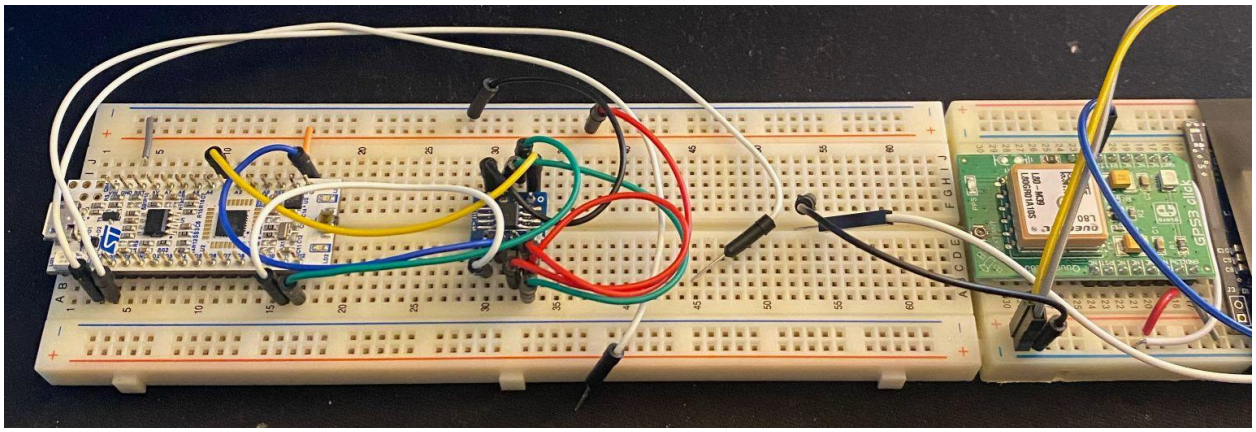


Figure 25: Breadboard early software test setup. STM32 MCU breakout board (left) connects to FRAM breakout (middle) and GPS breakout (right).

Board Programming Testing

Once the boards arrived, a new test setup was created that uses the STLink-v2[12] debug probe to connect to the JTAG header on the board to allow for code flashing and step-by-step debugging of the flashed firmware. This programmer was chosen for its ability to be plugged directly into the USB of the host computer. This test setup is shown below.

[12] STLink-v2 programmer: https://www.adafruit.com/product/2548

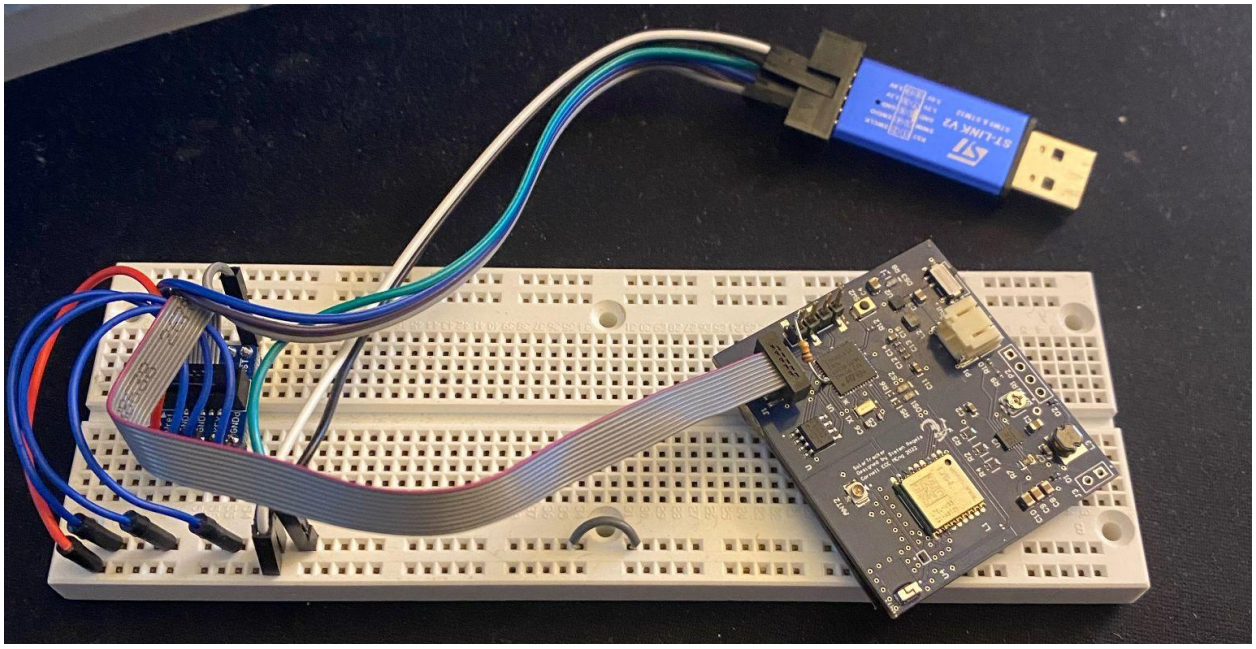


Figure 26: Board programming test setup, with ST-Link v2 USB programmer (top), SWD to JTAG converter (left) and JTAG cable to SolarTracker board (right).

Powering the board through the JTAG interface allowed for the MCU and 3.3V components to be powered with a constant power source, allowing for board testing before trying to power the device with a battery (3.7V) or solar cells (6.7V). The firmware was finalized during this time, in order to generate good success data without the risk of frying the board due to unforeseen issues with the solar circuit. One important bug discovered during this process concerned the BOOT0 pin on the MCU. From the datasheet[13], the BOOT0 pin is used for selecting the boot space for the program code when powering on the device.

Table 6. Boot modes

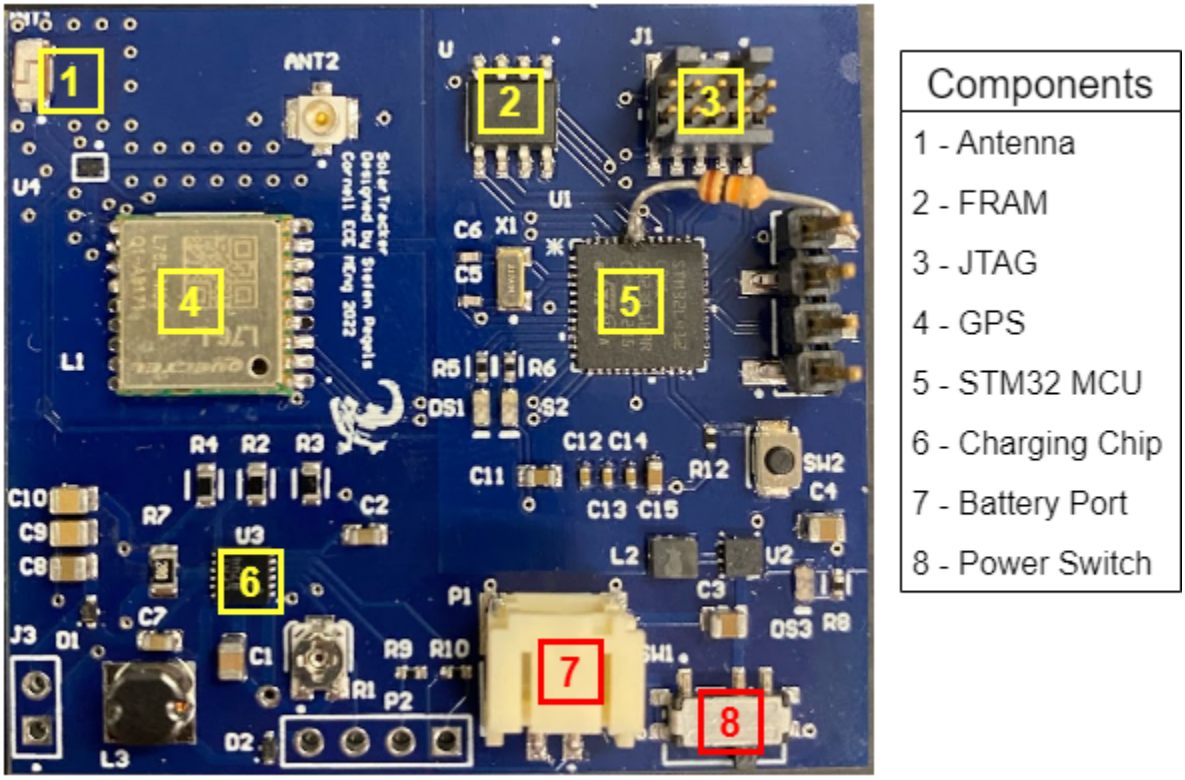
BOOT mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as boot space
0	1	System memory	System memory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM is selected as boot space

The values on the BOOT pins are latched on the 4th rising edge of SYSCLK after a reset. It is up to the user to set the BOOT1 and BOOT0 pins after reset to select the required boot mode.

Figure 27: Boot mode information for STM32L412 microcontrollers.

[13] STM32L4 Hardware Design application note

When designing the hardware schematic, this pin was left **floating,** as the value for BOOT0 was confused for BOOT1. This created undefined behavior when trying to flash code on the device as the program stack pointer would oscillate between flash memory (where the program was held) and system memory. To solve this, a resistor was soldered onto the pin and soldered to a nearby GND pin, allowing for successful boot of flash memory. A redesign of the board would include an added SMT resistor attached to BOOT0 to account for this problem.



Components
1 - Antenna
2 - FRAM
3 - JTAG
4 - GPS
5 - STM32 MCU
6 - Charging Chip
7 - Battery Port
8 - Power Switch

Figure 28: Image of Fabricated Board with BOOT0 10k resistor visible in the top right. Other important components are circled and labeled.

Another important hardware bug discovered was the inefficiency of the onboard SMT GPS. The reason for this occurrence is nebulous: it might be inadequate component spacing for the antenna or the RF switch prevented the component from receiving a fix, among others. A lesson learned here is to follow the manufacturer design specifications to a T for GPS antenna design. Since GPS acquisition was finicky with the SMT antenna, an offboard GPS chip connected through external UART pins was used occasionally for more consistent data collection. A future board redesign would spend more time investigating the GPS antenna system, with possible selection of a new antenna or direct contact with antenna manufacturers to approve the board layout prior to fabrication.

On the software side, the most time-consuming bugs came from the SPI firmware, which took the longest to get running. Initially the system was tested with a simple echo task that sent 3

bytes of random data and attempted to read them back immediately. Initially no data was ever able to be read, even though SCLK signals were visible on the oscilloscope. This bug was tied to the behavior of the SPI Write Enable Latch, controlled by the WREN command. Hidden in the FRAM datasheet was the phrase that “WREN is cleared after every write transaction”. This meant that the WREN command needed to be sent before every WRITE command. After this, SPI was finally run when nonblocking transmit and receive functions were replaced by their blocking counterparts. At the moment it is unclear why the nonblocking SPI calls were not functioning, perhaps due to a register enabling SPI interrupts not set properly.

Battery Testing

After functionality was verified powering the device through JTAG, A 3.7V LiPo battery[14] was used to confirm functionality of the buck converter circuit. This was tested walking around Cornell and getting a GPS fix, recording bursts of data, then returning to a host computer and sending the NMEA messages to a python script for parsing and logging. Visible in *Figure 29* is an example of a burst of data from the system, taken while walking in front of Statler Hall. This shows the precision of the GPS functionality of the device, as it is able to pinpoint over 10 locations close to each other with only a few meter radius for each one, with minimal error. This result is crucial in verifying the correctness of the GPS data recorded by the device, and its resistance to noise interrupting GPS data collection.

[14] 3.7V, 400mAh LiPo battery: https://www.adafruit.com/product/3898

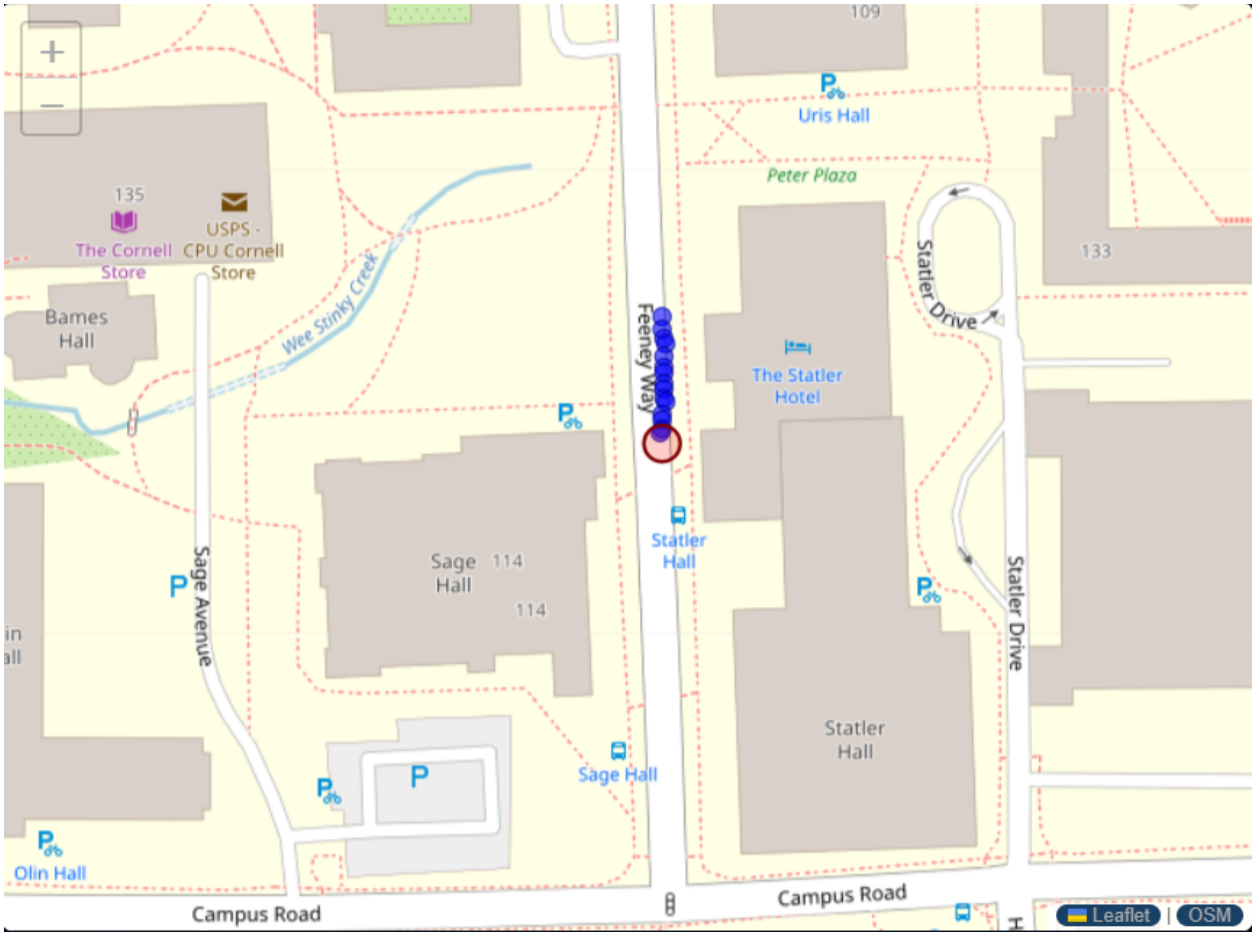


Figure 29: Small section of data taken walking in front of Statler Hall from device.

Battery testing also encompassed larger tests around campus, with larger sets of data to get a picture of the tracking proficiency of the board on longer trips, to simulate the use case of taking it for a run. In *Figure 30*, a walk from the Alumni Fields to Phillips Hall was recorded with data written to memory every 5 seconds. Powered by battery, the device performed very well, capable of holding a fix and not hanging on any errors. The only negative of this was the relatively long fix acquisition time, which took about 10 minutes. An improved version of the system would incorporate some kind of soft wakeup that allows for faster GPS acquisition rather than cold starts on power up. GPS sleep/low power mode is available for the L76-L M33 model used, but was not implemented due to the simpler scope and limited timeline of this project.

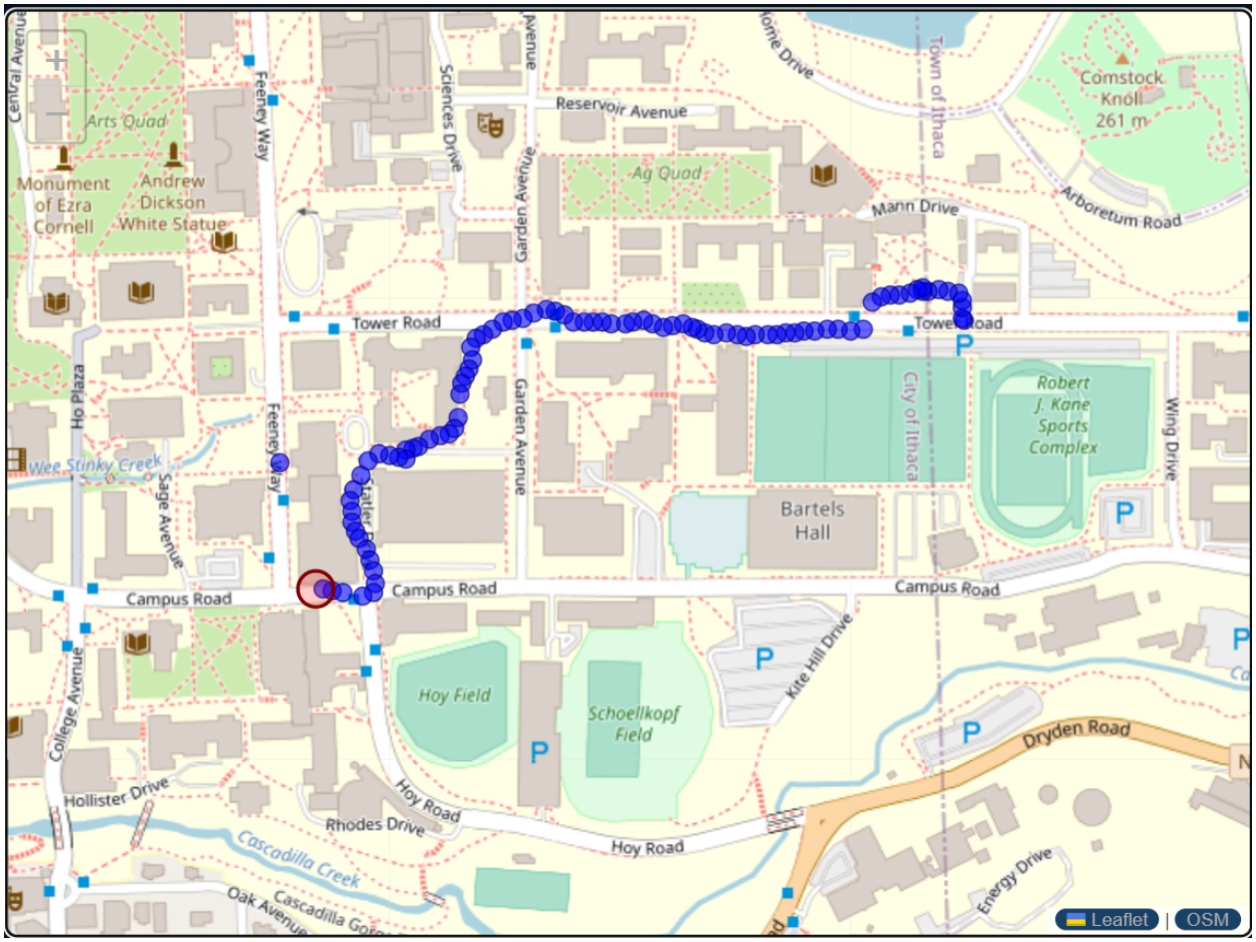


Figure 30: GPS data taken from a walk around campus - data recorded every 5 seconds.

Solar Testing

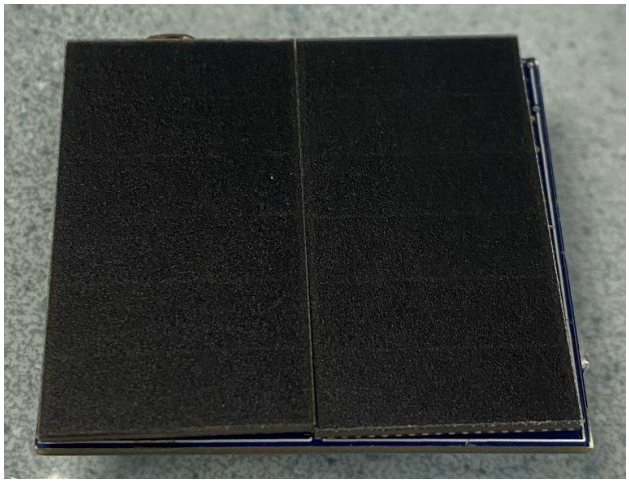


Figure 31: SolarTracker Board with solar cells mounted. Reverse side has all ICs as shown in Figure 28.

To test the functionality of the solar cells, the SMT cells were soldered onto the board and the VSOLAR was read in multiple contexts.

Location	VSOLAR
Indoors (Facing Floor)	0.24V
Indoors (Facing Light)	0.53V
Outdoors (Overcast)	4.48V
Outdoors (Sunny)	8.00V

Figure 32: VSOLAR measurement in different weather conditions. VSOLAR represents voltage input to the battery charger.

Indoors, there is understandably little to no solar power available for the device, and it runs on battery alone. Cloudy skies outdoors recorded 4.48V, which is nontrivial, but given the minimum voltage for the battery charger to activate is 4.95V, fully overcast skies do not provide enough solar voltage to charge the battery, and the battery runs the circuit by itself. In direct sunlight the cells effectively reach their max power point, measuring 8.00V which is enough to power the device by itself (though barely). This is the ideal setting, with the battery providing the alternative energy to keep the device powered when this voltage dips below the direct sunlight conditions. No power failures were recorded running with the solar cells and the battery simultaneously. GPS data taken in this context was similar to *Figure 30.*

Outcome Analysis

Other alternative designs for this system forgo the battery in exchange for a simpler power system, as this would remove the charging circuitry and free up some area on the board. This would also be a purely solar device, and could perhaps be a “purer” examination of the potential of solar energy to power a wearable device, which is the principal investigative question of this project. However, a system like this would be too unreliable except in cases of direct sunlight, and frequent power outages without any energy storage would create many soft resets of the firmware, leading to all sorts of software problems. Another possible option would be a small solar cell used for the sole purpose of trickle charging a battery; this solar cell would not have enough power on its own to power the entire device. While this design mirrors some industry applications with a solar component (ex some Garmin watches[15]), it moves the design problem away from solar energy and more towards prioritizing a battery system with an extended lifetime. In this design, the battery is a complementary feature to improve robustness and the solar cells are capable of powering the device by themselves. One design alternative was a flexible PCB, as this would create a more user friendly and adaptable system. Unfortunately, recent flexible solar cell manufacturers have gone out of business, leaving only rigid cells as the

[15] Garmin Solar Watch: https://www.garmin.com/en-US/p/679335

possibilities to be used. Since the rigid cells occupy the entire necessary surface area of this board, there was no option to make any part flexible.

This project is a first attempt at a hardware prototype created with no reference design; it understandably has a few modifications and hindsight realizations that would make the system more robust in the future. As mentioned earlier, a more robust GPS antenna circuit would make acquisition faster, as well as a software implementation of GPS sleep mode and warm start to hold a fix even when the device is not being used. Keeping the GPS in sleep mode would require a small amount of current through the device at all times, which might not hurt the system as long as it is used enough to have the solar cells charge the battery.

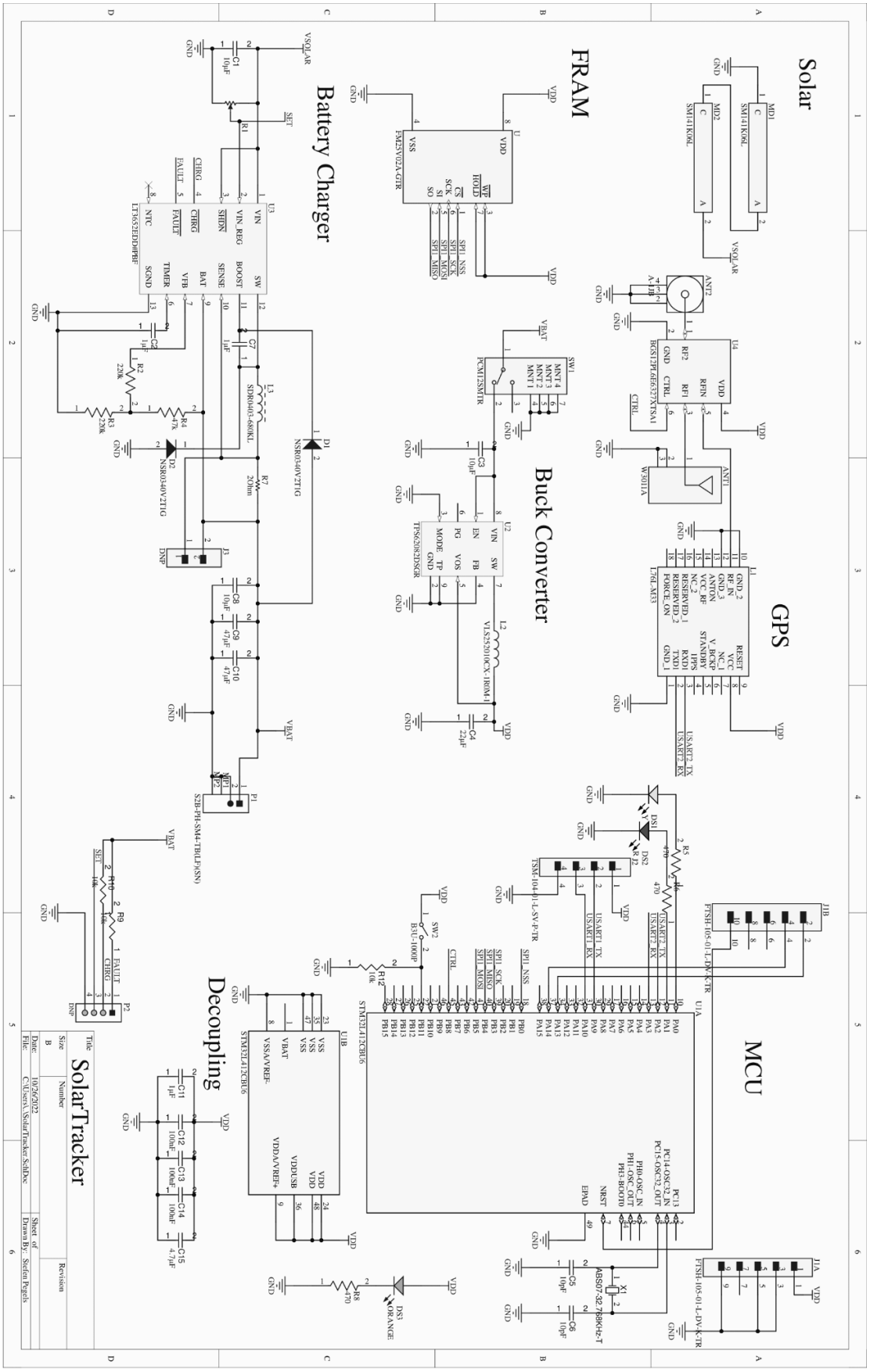
Another possible improvement is a selection of a different battery charging IC. The LT3652 has an operational voltage range of 4.95-40V, with these solar cells only generating 8V in direct sunlight. If a charging IC with a narrower voltage range and a lower minimum voltage were used, it would allow the cells to contribute to powering the device in cloudy conditions. In overcast weather the cells only generate 4.48V which contributes nothing to device power as the battery charger is in shutoff.

Conclusion

The initial inspiration for this project was creating a system that could track my location when I went for a run without having to worry about charging it beforehand. After a year of schematic, board, and software design I would say that this criteria has been fulfilled. Understandably there are a few shortcomings with the board due to design oversight or errors encountered during testing such as the inefficiency of the GPS or high voltage minimum for the battery charger. Outcomes like this would drive some different design decisions in a future iteration of this board which I plan to carry out in the future as a hobby project after leaving Cornell. This project involved a heavy amount of schematic reading and part research to drive selection and design decisions, many of which were made without an exhaustive search of all possible alternatives. This process could be expanded upon with a longer design cycle to iron out any kinks in board component selection that make the system less efficient. Besides that, the function of the board met expectations and stands as a successful prototype for this design project. I would like to thank Prof. V. Hunter Adams for his assistance as my mentor throughout this design process; his help will always be invaluable for this project’s success as well as towards my growth as an engineer.

Appendix

Source Code Repository: https://github.com/sgp62/SolarTracker_STM32L412CBU6.git
Altium PCB Project:
https://altium.com/viewer?token=D8im3zE%2BnUSYXA5e8St%2F%2BjeK
Datasheets: Embedded as footnotes throughout the document.
<<<Anything else to add?>>>


Solar
FRAM
GPS
Battery Charger
Buck Converter
MCU
Decoupling
SolarTracker