# Pedal-Powered Smart Bike

ECE 445 Design Document

Alex Sirakides, Karl Kamar, & Anshul Desai
Team 29
TA: Yifan Chen
10/01/2020

# 1 Introduction

## 1.1 Objective

With rising population rates, it is without a doubt that cities across the globe will increase in population density. With over [1] 1.7 billion people living in cities in 2018, estimated to reach 2.5 billion by 2050, one can only imagine how this might further increase traffic congestion resulting in longer commute times. As a result, many individuals are opting for alternative means of commute. [2] In fact, it is estimated that the amount of individuals who use a bike to commute to work will double by 2022. Outside of commuting, biking also serves as an excellent form of exercise. [3] Cycling is great for losing weight, cardiovascular training, as well as building up muscle. [4] Cycling can even benefit an individual's mental health, improving mood, sleep, and creative thinking! On the other hand, using a bike as a primary means of transport may also be risky for a cycler, especially in large, densely populated cities. We are even seeing cyclist-related accidents increase as time goes on. [5] With over 2% of motor vehicle crash deaths, cyclists still represent a significant amount of potentially avoidable accidents every year. [6] In 2018, there were over 857 cyclist deaths due to collisions with automobiles, representing over a 6% increase in deaths from the year prior.

Our project seeks to solve multiple issues that bicyclists face on a daily basis. We plan on creating a multi-tool for bicycles dealing with three main issues: navigation, exercise, and safety. Using a microcontroller, motor, RGB screen, and various other components, we will completely power our multi-tool through user pedalling and a battery. The RGB screen will be used to display important information for the user with regards to their navigation and exercise. A GPS module will inform the user of their heading and current location. Information captured from both the motor and the GPS will capture the user's speed, distance, and exerted power. Finally, an ambient light sensor and buttons will allow the user to be visible in low light conditions as well as inform other drivers and cyclists of their intent to turn.

## 1.2 Background

While there have been a couple attempts to emulate similar experiences, none offer a truly offline experience that allows an individual to discount from his/her devices and fully embrace the outdoors. SmartHalo [8] is a brand that has attempted something similar, yet falls short in aspects that our device allows. Offering certain similar features such as GPS location and light signaling, SmartHalo does not offer a pixel based interface, exercise statistics, and operates off of a battery instead of being powered by user pedaling. Additionally, it encourages the user to connect their phone to the device and the bright, multi-colored display may be distracting to some.

## 1.3 Visual Aid

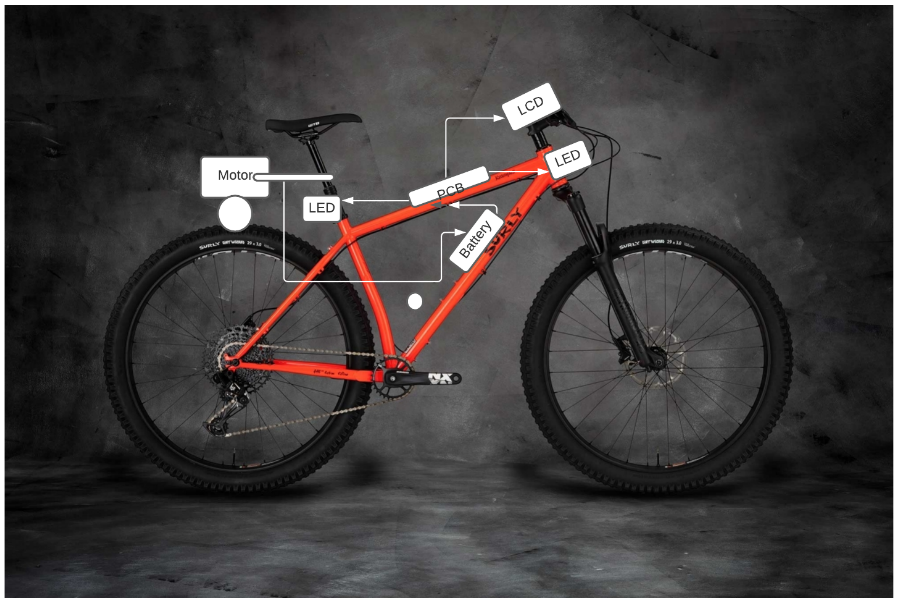


**Figure 1: Rough Physical Design Layout**

## 1.4 High Level Requirements

1. The user must be able to produce at least 9V by pedaling in order to charge the 8V battery.
2. LCD interfaces with the GPS module and displays the user's speed, distance traveled, longitude, latitude, current street name, current time, current date, time of ride at a refresh rate of at least 2 Hz (2 updates per second).
3. The ON/OFF button must open the circuit at the battery output, and Reset button must reset distance traveled and ride time to 0.
4. Turn-light buttons must make turning lights blink when pressed and turn them off when pressed again.

## 2 Design

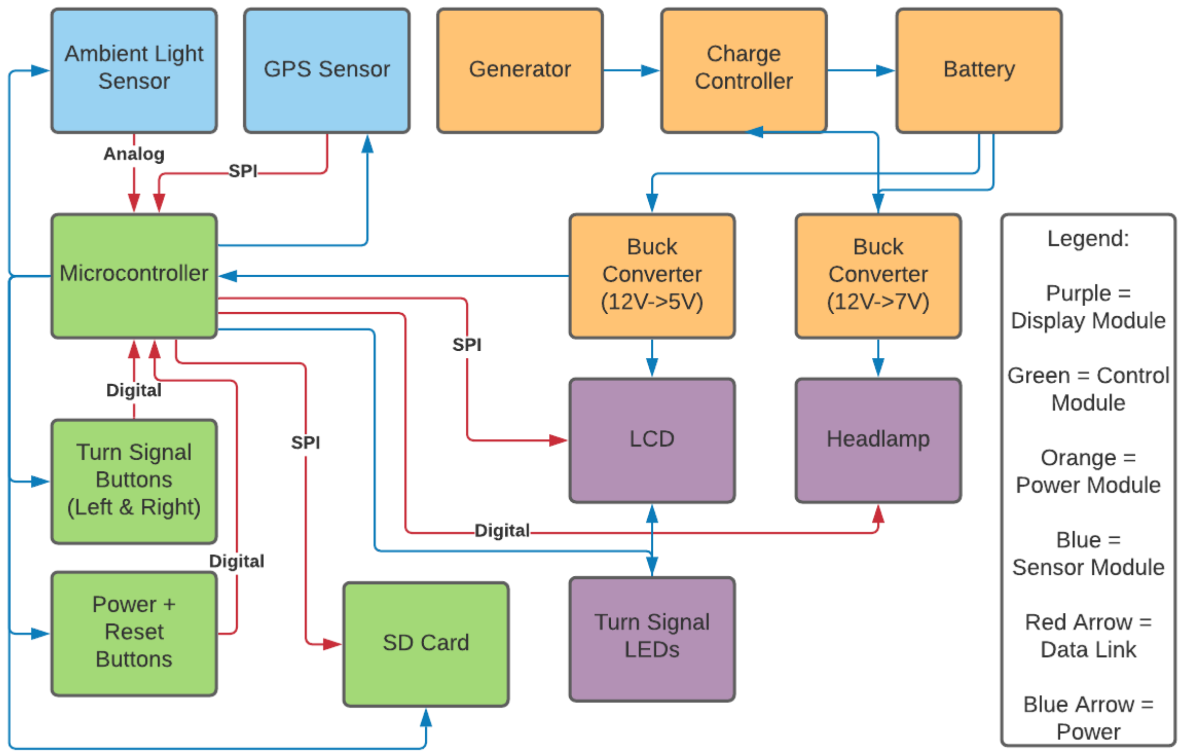


**Figure 2: Block Diagram**

### 2.1 Power Supply

The power supply system consists of a motor charging a battery aided by a controller for protection and a mechanical system for rotation. All that will be triggered by the user's pedaling motion. The motor must output 9V in order for the battery to at least maintain its current energy, and 9.6V to charge the battery.

#### 2.1.1 Motor

The motor will be attached to the back frame of the bike using a hinge system and will be hanging right over the back-wheel tyre. Its brush has an 11-tooth sprocket for a #25 chain. It will turn as the wheel turns using a rotation system we will describe next. The motor is rated at 50 Watts with an RPM rate of 3500 at 24Volts and an efficiency of 79%. It has a diameter of 4.0in and a length of 4.3in and a brush diameter of 0.47in. It will supply DC power to the battery. Using the rotation system, the torque can be calculated as

***T = F x sinθ x (length of motor brush)***, where F is the friction force exerted on the small metal wheel by the tire. That friction force can be calculated as ***F = μk x N***, where ***μk*** is the kinetic friction coefficient between rubber and steele and **N** is the normal reaction exerted by the tire on the brush which is equal to the opposite of the Weight of the motor, ***5.6lb x g.*** According to a table about friction coefficients, ***µk = 0.5*** [9]. So we have:

***F = 0.5 x 5.6x0.453592 x 9.81 = 12.46N***

***T = 12.46 x sin90 x 1.1in x 0.0254in/m = 0.348 N-m.***

According to our chart, at 0.348N-m, our motor turns at 2320 rpm producing 185W and 6.8A for an efficiency of 76%. That yields a voltage of ***185x0.76/6.8 = 20.7V.*** This is sufficient enough as we only need 9.6V to charge the battery and 9V to maintain it at a constant charge.

| Requirement | Verification |
|---|---|
| Needs to turn easily enough and not create too much drag. | Test the motor's resistance to turning as soon as possible at the machine shop and change position of motor to minimize resistance. Max torque: 5N-m . |
| Must turn enough times per minute to generate enough power. Have a big enough ratio of turning gear diameter to motor brush diameter to produce as high a voltage (>9V) | Use the wheel as the trigger to turn the sprocket. Use a voltmeter to test how much voltage is output. (Should be at least 9Volts) |

### 2.1.2 Rotation System

The rotation system will be composed of an intermediary piece composed of a small wheel (~1.5in diameter) that will be attached to the motor's brush. The small wheel will be in contact with the bike tyre and turn at the same speed and thus higher rate. It will thus turn the motor brush and produce electrical energy.

| Requirement | Verification |
|---|---|
| Needs to create enough friction to turn the motor instead of sliding. | Gregg will make the piece such that it is indented and created friction.<br>Ideal friction: 0.65N-m. |

### 2.1.3 Charge Controller

The charge controller's input will be connected to the motor and will output to the battery. It will ensure the battery does not over charge and that the voltage applied to is not excessive since the motor can output from 12V to 24V. The charge controller also acts as diode ensuring the current only flows in a single direction. That is, from the motor to the battery. (LM7808 Linear Voltage Regulator)

| Requirement | Verification |
|---|---|
| Should act as a diode. | Place it in a simple circuit between the battery and the motor. If the motor turns, then the controller is dysfunctional. |
| Should prevent the battery from overcharging. | Use a voltmeter to monitor voltage at the output of the controller while pedalling. Specs: 24-12V to 9.6V . |

### 2.1.4 Battery

The battery we have is a 8V, 2Ah Acid Lead battery. It is a rechargeable battery and will be connected to the charge controller from which its charge will come. It will in turn feed the rest of the system (control and GPS modules, sensors, lights).

| Requirement | Verification |
|---|---|
| Needs to have the capacity to power up the circuit. | Make proper calculations to not overload the battery. Max capacity of 3.2Ah. Duration and capacity values in datasheet [10] |

### 2.1.5 Buck Converters

We will use at least two buck converters between the battery and the circuit's components. These have a role of stepping down the voltage from the battery to the different peripherals depending on their nominal voltages. Our microcontroller has a nominal voltage of 5V, the GPS sensor has a Vcc of 3.6V and the ambient light sensor will range from 3.3V to 5V. The chosen LCD Display has a nominal voltage of 5V. Our rear LEDs will also have voltages ranging from 3V to 5V and our headlight LEDs might range from 5V to 8V. The ambient light sensor can be powered by the AVR microcontroller through the AVCC port. That is the supply voltage for the Port A which has an A/D Converter capability. There will be one buck converter for the Microcontroller, one for the headlights, one for the three rear lights and one for the GPS module (may be the same as for AVR).

| Requirement | Verification |
|---|---|
| Should step-down the voltages as needed. | Use multimeters to test them. |

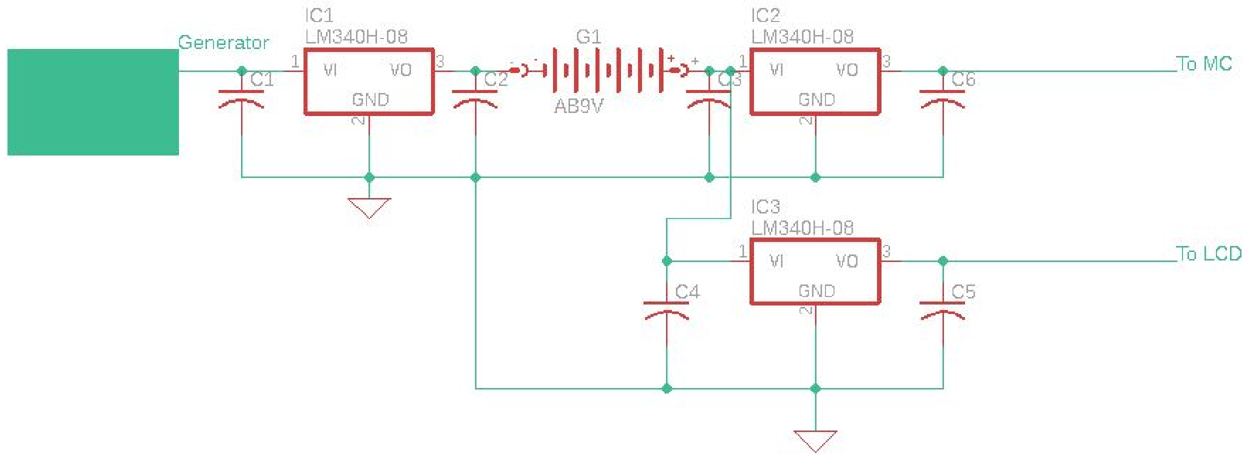


**Figure 3: Power Module**

## 2.2 Control Module

### 2.2.1 ATMEGA328 AVR Controller

In order to stream serial data from our various sensors for the processing of exercise analytics, we plan on using an AVR Controller, specifically the Microchip Technology ATMEGA328-PU. With a 28 pin configuration as well as 32 KB of flash program memory, this microcontroller will allow us to write our own programs and interface with our hardware sensors. The ATMEGA328 supports communication with multiple digital peripherals with its UART, SPI, and I2C ports. The AVR controller also operates from 1.8-5.5 V. Our microcontroller also has 2 KB of SRAM which is necessary to drive a video buffer to our LCD display. Upon validation, we can see that we can successfully accommodate pins for each and every sensor. More on this in each respective module's section.

| Requirement | Verification |
|---|---|
| Pin occupation requirements for LCD display | LCD display pin #4 (DC) to ATMEGA328 pin #14 (PBO), pin #7 (SCLK) to pin #19 (PB5), pin #8 (DIN) to pin #17 (PB3/MOSI), pin #15 (CS) to pin #16 (PB2), pin #16 (RST) to pin #15 (PB1). NOTE: We can change our pins using software SPI, we will need to in order to occupy pins for the SD card reader. |
| Pin occupation requirements for GPS module | GPS module TX pin to ATMEGA328 pin #23 |

| | |
|---|---|
| | (PC0) |
| Pin occupation requirements for SD card reader | SD card reader CLK pin to ATMEGA328 pin #19 (PB5), DO to pin #18 (PB4/MISO), DI to pin #17 (PB3/MOSI), CS to pin #16 (PB2). Although these overlap with our LCD display pins, we will change those pins through software SPI |

### 2.2.2 microSD Card Reader

We plan on using our AVR controller to write GPS sensor data onto our SD card in order to capture data respective for the entire trip. We will accomplish this by using the Adafruit microSD Card Breakout Board. While the size of the SD card is somewhat arbitrary since we will only be encoding unparsed sensor data kept as Strings, a good minimum we are looking for is at least 1 GB, up to 2 GB, classifying our SD card as FAT16 storage. Additionally, if we would like to preallocate data for potential reverse geocoding, the dataset we will be using is approximately 300 MB. Most standard SD cards support 2.7-5V. Anywhere past 5V will destroy the SD card. Our communication protocol for our microSD Card Reader is SPI.

| **Requirement** | **Verification** |
|---|---|
| FAT16 Requirement | Forcing a storage maximum of at most 4 GB qualifies as FAT16, however we would like at least 1 GB of storage of potential data |
| Pin assignments for microcontroller | Refer to pin listings in the microcontroller section. We can load an Arduino library in Atmel Studio 7.0, conveniently titled *SD*, to interface. More popularly, FatFs may also be used. This helps avoid having to create an entire filesystem for the microcontroller. |
| Min read/write benchmark > 10 KB/s | With a preferred GPS module capture rate of 5 Hz, we would like to write the most data possible into the card. The SRAM of our microncontroller is 2 KB, with 1 KB being reserved for our video buffer. Assuming we write all of the SRAM buffer (including video buffer for sake of minimum expectation), we need 5 writes * 2 KB / second. Using CircuitPython, benchmarks show a minimum read speed of 120 KB/s and a minimum write |

| | |
|---|---|
| | speed of 160 lines of data/s. This is well above our requirements and this is a minimum benchmark. |

### 2.2.3 USB Programmer

We will need to use a standard USBASP AVR programmer in order to program our AVR microcontroller. By connecting to GND and 3 pins on our microcontroller, we can write programs in a variety of languages (C/C++/Assembly) in order to receive data from our GPS sensor and display it on our OLED display. Since this is a fairly standard and proprietary device, there aren't any specific requirements.

## 2.3 GPS Module

The GPS module we've decided to use is the uBlox NEO-6M GPS Module. A very simple sensor, we only require power, a ground connection, and a pin available on our microcontroller to receive our transmitted data. The GPS module expects power of 2.7-5V, much like the other components of our circuit. The operating current of our GPS module is 45 mA. Data is captured as a String, so we need to ensure that the data transfer rate of the GPS module is high enough.

| **Requirement** | **Verification** |
|---|---|
| Transfer Speeds fast enough, > 1 KB/s | With a starting baud rate of 9600, we can transfer 9600 bits per second through our UART serial port, roughly 1.2 KB/s. |
| Refresh rate $\geq$ 2 Hz | Our GPS Module operates from 1 Hz (default, 9600 baud rate) to 5 Hz (maximum, 230400 baud rate). While we are aiming for the maximum possible frequency for the most accurate measurements, we believe at least 2 updates per second will be sufficient. |
| GPS Module provides relevant data | The GPS module provides time in UTC, latitude (degrees, cardinality), longitude (degrees, cardinality), speed over ground (knots), date, and magnetic variation through the $GPRMC sentence. |
| GPS data is digestible, we can create an array | We can leverage the TinyGPS++ library in |

| | |
|---|---|
| of 6 (or 8) pieces of data for each of our requirements | Atmel Studio 7.0 to scrape this data into digestible file types (whatever we would like). If not, we can parse the concatenated sentence for our values. |
| Pin assignment for microcontroller | Refer to the pin listing in the microcontroller section. |

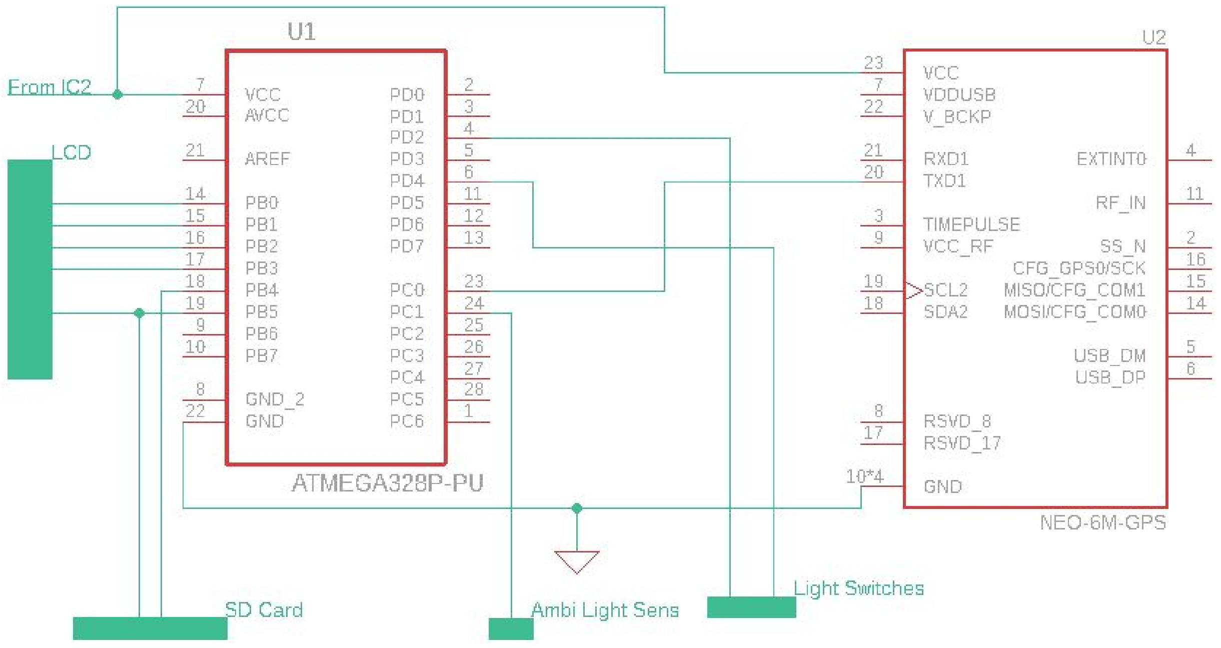


**Figure 4: Control Module**

## 2.4 Sensor and Lighting

Our project will equip the bicycle with the necessary lights to ensure the biker's safety at night. This consists of an ambient light sensor that will determine the lighting of the headlight and rear light as well as right and left turn lights controlled via buttons at the user's fingertips.

### 2.4.1 Ambient Light Sensor

Ambient light sensors require 3.3V to 5V of supply voltage. Since this is an analog sensor output sensor, it will be powered through the AVCC pin of the AVR ATMega Microcontroller we are using and will output data to a pin of Port A of the AVR. This is because the Port A of the microcontroller has the feature of serving as an analog input to the ADC of the microcontroller. That is, its Analog to Digital Converter. The code will determine at which specific input value received from the sensor do we switch the LEDs ON or OFF. The goal is to use the analog light sensor as a data input to control the LEDs and switch them on when it is dark outside. The output wire from the microcontroller carrying the command data regarding the sensor's input will be connected to the LEDs using transistor switches.

| Requirement | Verification |
|---|---|
| Must communicate real time information to the microcontroller. | Connect it to the microcontroller with a simple code and turn room lights on and off to see how fast the sensor reacts. |
| Powered by the ATMega chip. | Maximum 5V. |

### 2.4.2 Transistor Switches

The transistors will be used as switches between the buck converters and the powered peripherals using the corresponding output data of the microcontroller as the control pin of the switch. That way we would only power the LEDs depending on the code of the microcontroller. If not, they will constantly receive power from the battery and stay on the whole time.

| Requirement | Verification |
|---|---|
| Must support the current and voltage applied to it and output same as input voltage to the peripheral component. | Buy a transistor that can withstand 5V and test it by using a voltmeter/multimeter to check the output. |

### 2.4.3 Headlight and Rear Light LEDs

The headlights will be LEDs emitting a white light strong enough to illuminate the biker's way for at least 5 meters ahead. This requires 100-300 LED lumens which does not exceed 4W power which our battery produces easily. The rear light is the same but needs even less energy and brightness since it is red and only needs to be perceivable by vehicles behind the bicycle.

| Requirement | Verification |
|---|---|
| Provide the needed illumination to ensure user | Power the LEDs at night on a dark street to |

| | |
|---|---|
| safety at night. | check efficiency. |
| Lights must not exceed battery capacity | Ideally 3.3V LED |

### 2.4.4 Turn Signals

The turn signals will be positioned on each side of the bike at the rear. They will also be controlled via the microcontroller and transistor switches. However, these will receive their control data from buttons instead of the ambient light sensor that way they can be used throughout the day.

| Requirement | Verification |
|---|---|
| Lights must blink at an adequate rate. | Test the lights by coding the blinking pattern and commanding them to see if they can blink fast enough. |

## 2.5 Buttons

Our design requires 4 high-level control buttons. These are the ON/OFF button, which turns off the whole circuit, the Reset button, which resets the LCD Display, and two turn buttons, one for each side to control the turn lights described above.

### 2.5.1 ON/OFF Button

The ON/OFF Button will be connecting the battery to the buck converters acting as a general switch for the entire circuit. It will function as a regular switch. It will be big in size and will be placed in a convenient place for the user, either at the front or at the center of the bike frame.

| Requirement | Verification |
|---|---|
| Button must turn the whole circuit ON or OFF with no exception. | Get a regular electromechanical switch to avoid coding requirements and bugs. Test is in any circuit. |

### 2.5.2 Reset Button

The Reset button will be connected to the microcontroller. Once pressed, the microcontroller will react by sending an output signal to the LCD display which will reset the data on the screen. This is to be used by the user when starting a new trip. For instance, it will reset the time of trip and distance travelled to 0. This will be a push button.

| Requirement | Verification |
|---|---|
| Button must reset data once and not affect the circuit until pushed again. | Test buttons in a simple circuit with an LED and voltage supply where the LED turns on only when push-button is pressed.<br>State only happens once if the button is pressed.<br>Leaves FSM Reset State even if the button stays pressed for longer than state time.<br>Code must be tested so that the reset state happens only once for each presse of the button. |

### 2.5.3 Turn-Signal Buttons

These two buttons will be placed near the handles of the bike, one on each side. They will be big to make it convenient for the biker to use them. As soon as one of them is pressed/pushed, the microcontroller will output alternating HIGH and LOW signals to the corresponding light's switch with 1/4 second delays. This will make the corresponding turn light blink to inform the vehicles behind the bicycle which way the biker will turn at the next crossing. The lights will only be turned off when their corresponding button is pressed again.

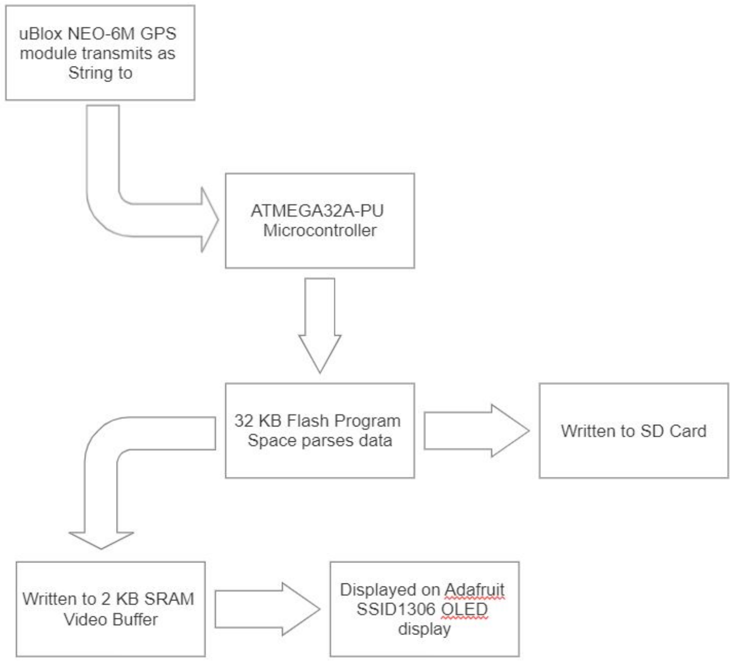
| Requirement | Verification |
|---|---|
| Buttons must send data adequately. | Test buttons in a simple circuit with an LED and voltage supply. |

## 2.6 LCD Display

For our LCD display, we have decided to go with the Adafruit Monochrome 1.3” 128x64 OLED display. The driver chip being used is the SSD1306, which commonly supports both SPI and I2C communication. As both are supported by our microcontroller, we found this to be the best option. As our microcontroller has a 2 KB SRAM, we can actually push a video buffer which requires at least 1 KB SRAM from our microcontroller. We opted for the use of monochrome, since we are only displaying text for the user. This display requires 3.3-5V and will generate around 40mA of current. The communication protocol we will be using is I2C.

| Requirement | Verification |
|---|---|
| Microcontroller video buffer > 1 KB | Our microcontroller has an SRAM of size 2 KB, which is twice our minimum requirement for our LCD display |
| Pin assignment for microcontroller | Refer to the pin listing in the microcontroller section. Since the pins are being shared with the microcontroller, we will reprogram the SPI pins through software, effectively halving our draw rate but allowing interface of both an LCD and SD card reader. |
| Response rate ≤ 200 ms | Due to the potential 5 Hz refresh of our GPS module, we will need our screen to update with relevant data AT LEAST 5 times within a second (200ms). The UG-2864ASWIG01 OLED display manufactured by WiseChip Semiconductor reports a response time of ≤ 10 µs, which is blazingly faster than our requirements. In theory this allows us to scale up the refresh rate of the GPS sensor past 5 Hz, if it were possible. |

## 2.9 Software


uBlox NEO-6M GPS module transmits as String to
ATMEGA32A-PU Microcontroller
32 KB Flash Program Space parses data
Written to SD Card
Written to 2 KB SRAM Video Buffer
Displayed on Adafruit SSID1306 OLED display

### 2.9.1 GPS Module Interface

As we are planning on processing the GPS module data in order to provide useful exercise and location statistics to our user, we will have to interface our GPS module with our AVR microcontroller. Fortunately, there is only one serialized TX output coming from our NEO-6M GPS module which we assign to our PD0 pin of our microcontroller. Our GPS data comes in as a serialized String, such as with the following example:

```
$GPGGA,141848.00,2237.63306,N,08820.86316,E,1,03,2.56,1.9,M,-54.2,M,,*74
```

We are given a String separated by commas with the first, “GPGGA,” referring to “Global Positioning System Fix Data.” The following value is a timestamp in -GMT, useful in providing the current time to the user. The next four values refer to the latitude and longitude, as well as their respective hemispheres, providing geopositional data to the user.

When our data streams to our AVR microcontroller, we can use both the change in timestamps and change of distance between two consecutive points of GPS measurement data to calculate speed. Based on the change in direction, we can also calculate the heading that the user is directed towards. There are also many libraries that can provide even more information to allow for more accurate measurements. The benefit of using an ATMEGA328 is that we can use simple C/C++ code to create all of these calculations.

Alternatively, we can use the $GPMRC sentence which contains heading as well as speed over ground in knots, which reduces the need for calculations to be made on the microcontroller itself. If we have enough leeway with program storage, we can even use a library like TinyGPS++ to parse data and perform calculations for us.

### 2.9.2 LCD Display Interface

Utilizing the PC4 and PC5 corresponding to the SDA and SCL pins for our Adafruit display, we can easily connect our AVR microcontroller via IC2. The process of creating a display buffer becomes relatively simplified by extending various free to use libraries that enable us to use simple write commands to display anything we would like on our monochrome display. By creating an always-on display template, we can simply create fields which are dynamically updated based on data read and processed from the GPS module and microcontroller which is stored onto flash program memory. Our display even supports 16 shades, however since we are just displaying text, we just need pixels to be on or off.

Adafruit provides an SSD1306 library on Github which is accessible by Atmel Studio 7, our preferred IDE for writing code for our ATMEGA328. There are also various libraries that aid in converting ASCII symbols into a renderable format. Our algorithm will utilize a 128x64 array (1 = Pixel on, 0 = Pixel off) and utilize the 2 KB SRAM as well as our reprogrammed serial out pin to buffer our display. We plan for our screen to update at the same rate of our GPS module.

## 2.10 Tolerance Analysis

The most challenging part of our project is arguably the power supply block, specifically the power generation using the motor. In fact, motors react by producing a resistance when they are turned. The faster they are made to turn, the bigger the drag and the harder it is to pedal. Done incorrectly, it can become impossible to turn. If that were to happen, our project would fail completely since the cornerstone of the project is its pedal-powered aspect. To elucidate this issue, we must refer to our manufacturer's website from where we have ordered the motor [10] . The chart on the website for our specific motor shows a clear decrease in the power produced by the motor as the torque exerted on it increases [11] .The efficiency has a max of 79% and we observe that it slowly decreases after the torque exerted on the motor exceeds 0.6N-m. We can make a simple calculation to get the average torque exerted on the pedal at the hub. That is ***2 x mass x g x π x radius***, which is a pedal distance from the hub of 15-20cm on average and an average person's mass of 65kg. We can reduce the mass value given that we are not standing on the pedals.

The torque value is therefore **50*9.81*0.17*sin(θ) ranging from 0 N-m to 83.4 N-m** depending on the angle at any given time. That it is a very big value and is not accounted for in the chart we have as it would stop the motor due to excessive drag. However, it is not a problem because we will not be exerting that much torque on the motor. The motor will be hanging above the back wheel and it's brush will be turned by a small system consisting of a wheel that will turn at the same rate as the tire due to friction force and another sprocket attached to it will be turning the brush motor's sprocket therefore exerting an efficient torque on the motor (calculated in part 2.1.1) and at a good enough speed since the ratio of the wheel to that of the motor's sprocket is very big. That means that for every turn of the wheel, the motor will turn multiple times. The exact ratio is as follows: A bike wheel is 23in in diameter and the 11-tooth #25 sprocket is 1.002in in diameter. That yields a ratio of 23/1.002 = 23. So for every wheel turn, the motor will turn 23 times.

## 2.11 COVID-19 Contingency Plan

In the event that COVID-19 forces us to return to online classes before the project is completed, we will need to shift the way that we approach our project. That said, the degree to which the project needs to shift will be contingent on which milestones are achieved prior to the time that we are sent home. Given that our project is centered around the proper application of a motor attached to our bike, the central milestone will be properly installing the motor and battery on the bike with the help of the machine shop.

Should we fail to reach this mark prior to being sent home, we will need to change the design to a proof of concept more so than a finished product. Meaning, mounting the motor on the bike will have to be scrapped and we will have to create a rig which allows for some stationary pedaling. From here, we should be able to build out the same project and test most functionality in a similar way. Lights can still be tested, buttons should still work as intended, and the display should still have all functionality. The biggest issue will be whether or not the gps module functions and for this we could still test it by charging the battery sufficiently and then demonstrating its use in a car.

Conversely, should we successfully mount the motor and battery prior to being sent home, we should be able to continue the project as planned. One of our teammates, the one who owns the bike being used, has access to a soldering iron and basic electronics equipment and should be able to build out the system from home as needed. From there, the rest of the team can still adequately participate in schematic and PCB design, advising, and all necessary software components of the project.

This means that the project should be able to continue in sufficient capacity whether we are sent home or not and having already delivered the motor and bike to the machine shop, we are quickly approaching the pivotal milestone in our contingency plan.

## 3 Cost and Schedule

### 3.1 Cost

The average time we are spending on working for this project each per week will be approximately 8 hours. Considering an Electrical/Computer engineer's average hourly wage, and adding the wages of two technicians helping us and two janitor for the building, we would estimate the labor cost to be:

**40$/hour x 2.5(2 technicians and 2 janitors) x 8hours/week x 3engineers = 2,400$/week**

The material costs is as follows:

- Battery: $22.95
- Motor: $77.13
- Transistor Switches: 0$, use existent
- Linear Voltage Regulators: 4 x 1.55 = $6.2
- ATMega328 microcontroller:
- LCD Display:
- NEO 6M ublox:
- PCB:
- Headlight and Rear Light: $13.99
- Turn Lights: $11.98
- Miscellaneous Buttons: $8.99

### 3.2 Schedule

The schedule for our project is as follows:

- Test our circuit on breadboards (Window: 10/1/2020 - 10/8/2020)
- Start working on mechanical part and installation with Machine Shop (Window: 09/29/2020 - 10/29/2020)
- Get PCB drawn and approved(Window: 10/8/2020 - 10/15/2020)
- Write code and test it (Window: 10/5/2020 - 10/29/2020)
- Finalize code (Window: 11/2/2020 - 11/13/2020)
- Finalize installation (Window: 11/9/202 - 11/20/2020)

| Week of | Monday | Tuesday | Wednesday | Thursday | Friday |
|---|---|---|---|---|---|
| 09/28 | | Machine shop started working on motor installation | | | Order electronic components for the circuit. |
| 10/5 | Start writing codes for microcontroller | | | Have our circuit prototypes built on breadboards and test them | |
| 10/12 | Test our codes on breadboards and try. | | | Have our PCB done and approved | |
| 10/19 | | | | | |
| 10/26 | | | | Order new PCB if needed | Test our PCB |
| 11/2 | Finalize code | Finalize code | Finalize code | Start installing the PCB and Display on the bike frame. | Start installing the PCB and Display on the bike frame |
| 11/9 | | | Mock Demo | Mock Demo | Mock Demo |
| 11/16 | | | DEMO | DEMO | DEMO |

## 4 Safety and Ethics

We believe the main ethics concerns in our project stems from the responsible design and implementation of the mechanical components, which have the greatest potential for harm to the user.

To address this, we refer to IEEE 7.8 # 1. “to hold paramount the safety, health, and welfare of the public…” and 6. “...to undertake technological tasks for others only if qualified by training or experience…” which both guide our approach to the design. The way 7.8.1 applies is in the design itself.. By creating a system whose goal is to allow bicyclists to travel the road more safely, we are looking out for public interest. The implementation of guideline 7.8.6 will require that we consult others with more mechanical design experience to ensure that our design is both safe in principle, and practice as far as the user is concerned.

Additionally, there are environmental and health concerns associated with the use of a lead-acid battery, especially one deployed on a vehicle which will likely be stored outside. To ensure that the battery cannot cause harm to the user or environment, it will have to be properly enclosed such that weather and other external factors cannot effect it.

With regards to software, since we are relying for this experience to be powered completely by the user as well as information fed from the GPS module, one concern that arises is the lack of accurate measurements and statistics. Fortunately, the GPS receiver we had decided to use, the uBlox NEO-6M, is a 50-channel receiver [12]. While we need at most 12 channels in order to receive the most accurate measurement of one's location possible, a 50-channel receiver will ensure that we are able to pull location statistics based off of the 24 different GPS satellites in the sky right now.

## 5 Citations

[1] un.org, Department of Economics and Social Affair, News, "Around 2.5 billion more people will be living in cities by 2050, projects new UN report", 16, May 2018. [Online]. Available: https://www.un.org/development/desa/en/news/population/2018-world-urbanization-prospects.html

[2] Bicycling, "New Study Says Bicycles Are the Future of Urban Transportation", Jan 15, 2020. [Online]. Available:
https://www.bicycling.com/news/a30518994/deloitte-2020-study-bicycle-transportation/
https://www2.deloitte.com/content/dam/insights/us/articles/722835_tmt-predictions-2020/DI_TMT-Prediction-2020.pdf

[3] Harvard Health Publishing/Harvard Medical School, "The Top 5 Benefits of Cycling", August, 2016. [Online]. Available:
https://www.health.harvard.edu/staying-healthy/the-top-5-benefits-of-cycling

[4] Mensline Australia, "Cycling - The Exercise For Positive Mental Health". [Online]. Available:
https://mensline.org.au/mens-mental-health/cycling-positive-mental-health/#:~:text=Regular%20riding%20helps%20synchronise%20your,proper%20regenerative%2C%20deep%20sleep%20difficult.&text=Improves%20creative%20thinking.,both%20physical%20and%20mental%20function.

[5] IIHS HLDI, Fatality Facts 2018, Bicyclists, December, 2019. [Online]. Available:
https://www.iihs.org/topics/fatality-statistics/detail/bicyclists

[6] NHTSA, Bicycle Safety, "Overview". [Online]. Available:
https://www.nhtsa.gov/road-safety/bicycle-safety#:~:text=In%202018%2C%20there%20were%20857,fatalities%20on%20our%20nation%27s%20roadways.

[7] Ieee.org, "IEEE IEEE Code of Ethics", 2020. [Online]. Available:
https://www.ieee.org/about/corporate/governance/p7-8.html

[8] SmartHalo, “SmartHalo 1.” [Online]. Available:
https://www.smarthalo.bike/smarthalo1/
[9] Tribology-ABC, “Coefficient of friction for a range of material combinations”.[Online]. Available: https://www.tribology-abc.com/abc/cof.htm
[10] Battery Mart, “PS-832 8Volt 3.2Ah Rechargeable Lead Acid Battery.” [Online]. Available:
https://www.batterymart.com/pdfs/sla-ps-832.pdf
[10] AmpFlow, “Pancake Motors”. [Online]. Available:
https://www.ampflow.com/pancake_motors.htm
[11] AmpFlow, “P40-350, 24V Chart”. [Online]. Available:
https://www.ampflow.com/P40-350_Chart.gif
[12] uBlox, “NEO-6 Series”, 2011. [Online]. Available:
https://www.u-blox.com/sites/default/files/products/documents/NEO-6_ProductSummary_%28GPS.G6-HW-09003%29.pdf