Redefining Personal Transport with Wearable Haptic Navigation

By

Kevin Xia - kevinx3

Soo Min Kimm - skimm2

Stephen Battin - sbattin2

Design Document for ECE 445, Senior Design, Spring 2021

TA: Andrew Chen

04 March 2021

Project No. 41

Contents

1 Introduction **1**

1.1 Background 1

1.2 Objective 1

1.3 High-Level Requirement 2

2 Design **3**

2.1 User Flow 3

2.2 Visual Aid / Physical Design 4

2.3 Block Diagram. 4

2.4 Block Diagram Requirements and Functional Overview 5

2.5 Schematics 14

2.6 Tolerance Analysis 16

3 Cost and Schedule **20**

3.1 Cost Analysis 20

3.2 Schedule 22

4 Ethics and Safety **23**

Reference **24**

1 Introduction

1.1 Background

Although biking and skateboarding around cities is not a brand new concept, new technological advancements in recent years along with the dangers that COVID-19 poses to those who take public transportation have led to the rise in this form of transportation [1]. This fact is extremely noticeable in cities and college campuses where the majority of the community would choose to bike, skateboard, or walk instead of driving or taking the bus with the inclusion of new bike and scooter ride-share programs. The convenience of not having to find a parking spot or needing to worry about getting sick is extremely enticing and the result is an environment with high levels of walking and biking in conjunction with high levels of vehicle traffic. However, this new increase in traffic comes with major drawbacks as well. With smartphones being heavily relied upon for children and young adults, more than $20\,\%$ of cycling accidents in 2015 were caused due to the phone use [2].

Wearable technology with haptic feedback can minimize this unnecessary use of smartphones during transportation by removing the biggest distraction from the smartphone, its screen. With this removal, the user will have nothing else to focus one's attention on except for the oncoming traffic and one's speed. This provides a safer atmosphere for not only the user but also the vehicles alongside the user. Also, by giving haptic feedback on the user's left or right arm, redesigns of commonly used software during travel, like turn by turn navigation, can be replicated with simple vibrational motors on either side of the user's arm. The less interaction between the user and user's phone during travel, the lower the chance of injury.

1.2 Objective

As the evidence is clear that phones prove to be a fatal distraction for cyclists, there are two major problems we will address in our design. The first is the overall dependence on smartphone technology modern cyclists have. The second is the lack of convenient, easy to use, and most importantly safe technology that exists as a viable replacement.

To eliminate the smartphone distraction when using our wearable, we are developing technology to replicate the same functionality of the turn-by-turn navigation from a smartphone on our wearable. This system will allow a user to "feel" which direction the user will need to take by giving vibrational feedback on the left and right sides of the user's arm. By varying the intensity, duration, and position of the vibration as the user approaches a turn, we can easily communicate to the user about upcoming turns, whether they be left or right, without the need for visual or audible feedback. This solution will allow the user to safely get to the desired destination and completely remove the phone-to-user interaction that previously would have needed to exist.

To accommodate outdoor athletes, we are developing a battery system that will ensure plenty of battery life while still maintaining the comfort and convenience that people have grown to expect in modern society. With the inclusion of solar panels on our wearable, the battery life can far exceed a single charge and will be able to extend the battery life far past the hour and a half it is designed for, given adequate lighting conditions. This will increase the flexibility of where our product could be used and be able to adapt to all potential use cases.

1.3 High-Level Requirement

- An app-based GUI using the Mapbox API that will allow a user to enter a desired destination and receive turn-by-turn directions to that destination.
- A solar panel mounted on our wearable that will increase the battery life of the device by ensuring a constant $265\,\mathrm{mA}$ output when in direct sunlight.
- A microcontroller that will receive updates from our Android phone via Bluetooth every 2 seconds or less and provide the current of $75\,\mathrm{mA}$ to both of our vibration motors.

2 Design

2.1 User Flow

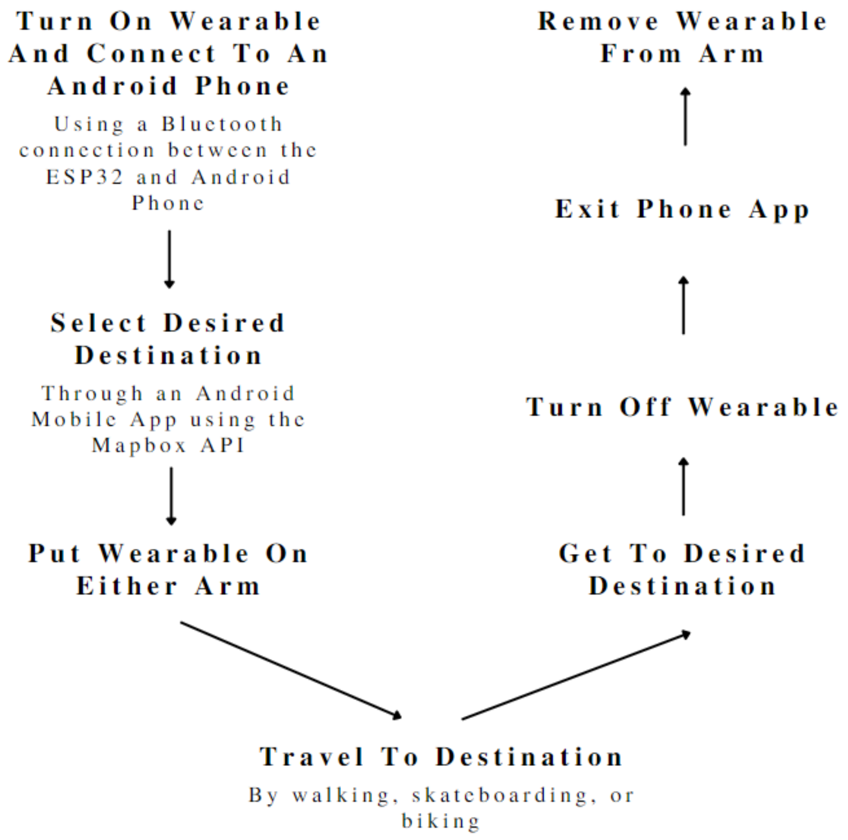
Turn On Wearable And Connect To An Android Phone
Using a Bluetooth connection between the ESP32 and Android Phone
Select Desired Destination
Through an Android Mobile App using the Mapbox API
Put Wearable On Either Arm
Travel To Destination
By walking, skateboarding, or biking
Get To Desired Destination
Turn Off Wearable
Exit Phone App
Remove Wearable From Arm

Figure 1: User Flow

Figure 1 shows how the user will interact with the device and app together. When the user turns the device on, the microcontroller and the Bluetooth receiver will turn on. The Bluetooth receiver immediately goes into a pairing mode where it will search for a device to connect to. The user will then use an Android phone to pair to the device. Once the device is paired, the user will be prompted to select which arm the device is going to be worn on and then will be asked to enter the desired destination. Once a destination is entered, the user can put the device on either arm and begin traveling to the selected destination. Upon arrival at the desired destination, the user will be notified and will be able to take off the device and turn it off. If the user forgets to turn the device off, the app will remind the user as well.

2.2 Visual Aid / Physical Design

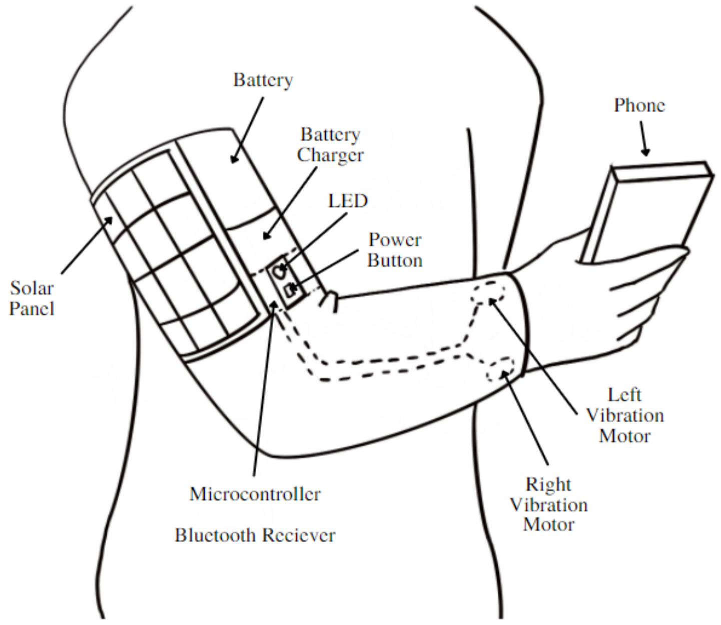

Figure 2: Physical Design

Figure 2 shows how the device will look when the user is wearing it. The main part of the device will contain all of the components except for the vibration motors which will be located farther down the user's arm, near the wrist. The vibration motors will be connected via wires running off of the PCB located on the bicep. The user will use the phone at the beginning of the trip to connect to the device and enter a destination. After this, the phone should be put away to prevent the user from getting distracted. The solar panel is located on the main part of the device facing out to maximize its efficiency.

2.3 Block Diagram

Figure 3 provides a general overview of how the components should be powered and communicate data. The device will be operated based on five different subsystems: user interface, power supply, control unit, input device, and output device. The user interface allows the user to communicate with the device through his/her phone application which shows the map. When the user presses the power button, the device will activate the battery which will power on the microcontroller, ESP32–WROOM. When the user sets the destination on the phone, the data will be transferred to the Bluetooth receiver that is built in the microcontroller. This data which will include the distance from a turn as well as which direction to turn will be used to determine which of the two vibration motors to activate and for how long they should be active.

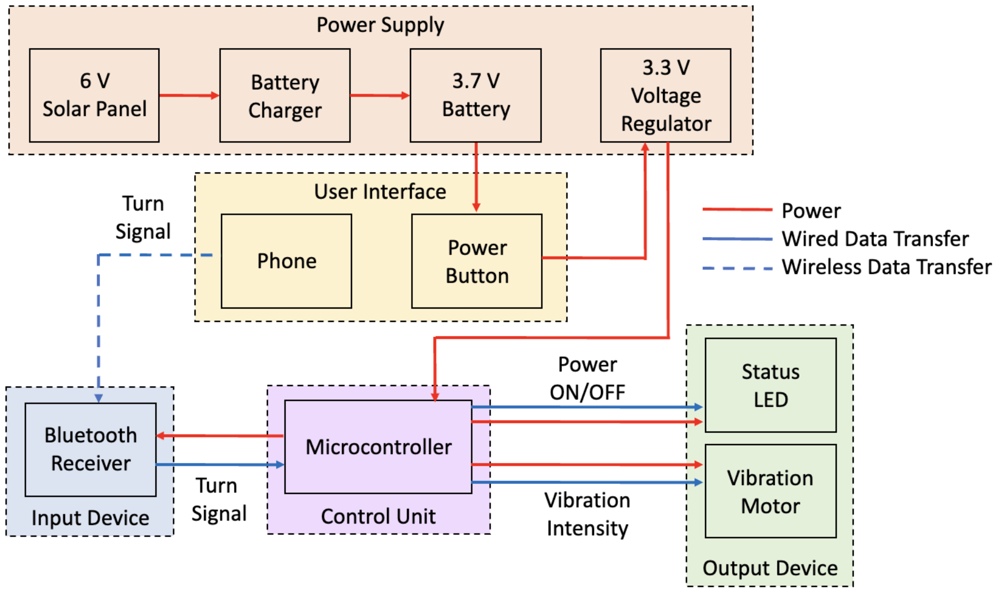

Figure 3: Block Diagram

2.4 Block Diagram Requirements and Functional Overview

2.4.1 Power Supply

Every customer will have some different environments and use cases for our product, and to best serve all of them, we are developing a battery system that will ensure a long battery life as well as a comfortable experience for the user. The power supply unit includes a solar panel to help increase the amount of time that the device will be able to function. In addition, this will allow the device to charge while not in use by simply leaving it out in the sun. We still have a conventional charging system that allows the user to plug the wearable directly into the wall using USB. All of this power will be stored in a battery. The release of this energy for the system to use will be controlled through a voltage regulator which will ensure that the components are not damaged from unregulated voltages and currents.

- **Solar Panel**

Requirement	Verification
1. Maintain the output over $0.1\,\mathrm{A}$ and $0.2\,\mathrm{V}$ which are the minimum requirement for the battery charger.	1. Set up the testing environment, assuming that the solar panel is under the full sunlight, by checking the luminosity using the photoresistor. 2. Place the solar panel under the testing environment and measure the open circuit voltage using the voltmeter. 3. Connect the output with the reference load to measure current using the ammeter. 4. Change the luminosity of the environment to find the least luminosity that satisfies minimum voltage and current requirements.
2. Regulate the output current and voltage under $1.5\,\mathrm{A}$ and over $4.2\,\mathrm{V}$ respectively under the power protection setup.	1. Place the solar panel under the full sunlight condition to maximize the output voltage and current. 2. Place diodes and capacitors suggested from the datasheet at the output of the solar panel [3]. 3. Measure the voltage and current and change the value of resistors and capacitors if required. 4. Connect with the battery charger when all conditions are met.

- **Battery Charger**

Requirement	Verification
1. Set the resistor value such that the output current is over $100\,\mathrm{mA}$, preferably $500\,\mathrm{mA}$ for the fast charging.	1. Set the battery charger circuit according to the typical application circuit from the datasheet [3]. 2. Place the reference load at the output of the battery charger to measure the current using the ammeter. 3. Change the resistor value between ISET (Fast Charge current Programming Input) and GND, which can range from $590\,\Omega$ to $8.9\,\mathrm{k}\Omega$ to set $500\,\mathrm{mA}$.
2. Force stop charging the battery if the temperature of the battery is too high to avoid damage to the skin.	1. Place a $10\,\mathrm{k}\Omega$ NTC thermistor between TS (External NTC Thermistor Input) and the battery. 2. Place the solar panel under the full sunlight condition and place the battery and battery charger in the container with a heat source which sets the temperature of the container to $35\,^{\circ}\mathrm{C}$, assuming hot summer days, then measure the temperature of the battery. 3. Cover the battery with thicker fabrics or put refrigerant around the battery if required.

- **Battery**

Requirement	Verification
1. Maintain 3 7 – 4.2 V of output when charged properly.	1. Fully discharge the battery. 2. Charge the battery fully under full sunlight conditions and measure the time taken. 3. Measure the voltage across the battery using the voltmeter.
2. Check the battery life of the device.	1. Fully charge the battery. 2. Run the device under no light condition, where the battery will not be charged while running the device, and measure the time taken until the device is turned off.

- **Voltage Regulator**

Requirement	Verification
1. Supply the output of 3.3 V and 800 mA from the input voltage in the range of 3 7 – 4.2 V.	1. Set the voltage regulator circuit according to the typical application circuit from the datasheet [4]. 2. Supply the power from the battery and measure the output voltage and current. 3. Change the value of resistors to satisfy the condition if required.

2.4.2 Control Unit

The control unit consists of a microcontroller, the ESP32–WROOM. The microcontroller is responsible for receiving all signals from the input device and deciding next actions based on received signals. It can communicate with the user's phone through the embedded Bluetooth module. Since the Bluetooth module on the microcontroller is an independent feature, it be discussed under the input device unit.

- **Microcontroller**

Requirement	Verification
1. Send two distinct power signals out at a time with each being no more than $80\,\mathrm{mA}$.	1. Build a simple circuit with two LEDs on a breadboard. 2. Attach the microcontroller as the power source for each LED with each LED being hocked up to a different pin on the microcontroller. 3. Write a program to power each pin and then to lower the power. 4. Use the ammeter to check if the output current satisfy the requirement.
2. Send a power signal to the vibration motors turning them on.	1. Attach the vibration motors to the pins tested in the requirement 1 of the microcontroller. 2. Rerun the program that was written for requirement 1 and verify that the vibration motors begin to vibrate.
3. Interpret the Bluetooth signal that the microcontroller is receiving	1. Hook the microcontroller up to the circuit from requirement 1. 2. Write a tester mobile app which sends a signal of 1 for the first LED and a signal of 2 for the second

	3. Write a program for the microcontroller that will turn on the pin with the first LED when a 1 is received and will turn on the pin with the second LED when a 2 is received. 4. Using an ammeter, ensure that a current is flowing and the pins are in fact turned on

2.4.3 Input Device

The input device unit provides data required for the microcontroller to control other components from the phone. However, it will not be interacting with the user directly. The Bluetooth receiver is integrated into the microcontroller that we are using. This adds a level of simplicity since the Bluetooth signal will then be directly able to be interpreted by the microcontroller. The Bluetooth receiver updates data from the phone every few seconds, which will then be transferred to the microcontroller. Since the data transfer does not have to be continuous, BLE, Bluetooth Low Energy, will be used, which will help with lowering our power usage.

- **Bluetooth Receiver**

Requirement	Verification
1. Communicate with the phone via its Bluetooth module.	1. Write a mobile app that sends a Bluetooth signal when the button on the phone screen is touched. 2. Wire the microcontroller up to the LED which turns on when the microcontroller receives the signal. 3. Write a program for the microcontroller to power the pin that the LED is connected to when it receives the signal.

	4. Activate the Bluetooth module on the microcontroller and connect with the phone's Bluetooth to send signals and check if the LED turns on when the button on the phone screen is touched.
2. Control both vibrational motors with the incoming Bluetooth signal.	1. Wire the vibration motors up to the microcontroller. 2. Send a Bluetooth signal from the mobile app. 3. Measure both vibrational motors with an ammeter to ensure at least $75\,\mathrm{mA}$.

2.4.4 Output Device

The output device unit is what physically interacts with the user. A status LED indicates that the device is turned on. The vibration motors that are located on the either side of the user's wrist communicates to the user when to turn left or right. The intensity and duration of the vibration motors will be controlled by the microcontroller based on the signal received from the phone.

- **Status LED**

Requirement	Verification
1. Turn on the LED when the device is turned on using less than $20\,\mathrm{mA}$.	1. Connect the LED with the microcontroller. 2. Write a program for the microcontroller to power the LED on when the microcontroller gets the supply voltage through 3V3. 3. Measure the output current flowing through the LED.

- **Vibration Motor**

Requirement	Verification
1. Regulate the intensity of the vibration to avoid hurting skins.	1. Hook the vibration motor up to the maximum power the motor can handle. 2. Place the vibration motor on one's arm with several layers of padding in between. 3. Remove layers one at a time ensuring no harm from the vibration. 4. If the vibration is too strong for the bare skin, lower the power to the vibration motor.
2. Vibrate to a level that the user can detect.	1. Hook the vibration motor up to the power and the ground. 2. Slowly increase the power to the vibration motor until one can feel the vibration.
3. Vibrate for at least 10 seconds.	1. Hook the vibration motor up to the power and the ground. 2. Time 10 seconds and verify that the vibration motor is still vibrating.
4. Turn off within 1 second of the power being removed.	1. Hook the vibration motor up to power and ground. 2. Turn off the power supply and measure how long it takes to turn off.

2.4.5 User Interface

The phone is responsible for running the app that will assist the user in the functionality of the device. In the app, the user can set which arm the device will be worn on. This decision determines which motor vibrates depending on the direction that the user needs to turn. In addition to setting the arm position, the user should be able to enter a desired destination to navigate to. The app will then get the step-by-step directions to arrive at the desired location. Once the directions are found, the app will begin communicating to the device when and which motor should vibrate. This is determined based on the GPS location of the user's phone in relation to the directions.

- **Android Phone**

Requirement	Verification
1. Send Bluetooth signals to the microcontroller every two seconds or less.	1. Turn the Bluetooth module on the microcontroller on and connect the phone to the Bluetooth module by searching for it in the Bluetooth menu. 2. Verify a successful Bluetooth connection by waiting for the Android device to say "Connected" next to the microcontroller.
2. Get turn by turn directions from the user's current location to the user's selected location.	1. Launch the app 2. Verify that the blue bubble icon is located on the map in the user's actual current location. 3. Search for a destination and show the route to this destination from user's current location. 4. Click a button named "Start Navigation" and have a 3D GUI display that shows the user turn by turn directions to user's selected destination.

- **Power Button**

Requirement	Verification
1. Pass the power from the battery to the voltage regulator to turn on the device.	1. Build a simple circuit with an LED, a resistor, a power button with the power supply. 2. Flip the button and check if the LED turns on.

Table 1: Power Button RV

2.5 Schematics

2.5.1 Power Supply

Figure 4 provides the physical connection of the components within the power supply unit. A power button, which the user can interact with, is located in between the battery and the voltage regulator. This is more efficient than putting the button right after the voltage regulator since there might be unnecessary power loss through the voltage regulator even when not in use. For the power protection, diodes and capacitors are used whenever the power is being supplied to another component. The value of the resistors and capacitors will generally follow the recommendations from the datasheet provided [3] [4], but may change after testing each component to fulfill the requirement for supply voltage and current. The solar panel generates $6\,\mathrm{V}$, which will then flow through the battery charger, and to the $3.7\,\mathrm{V}$ Li–Ion battery. $10\,\mathrm{k\Omega}$ thermistor limits the temperature of the battery to ensure that it does not exceed its operating region. R_1 and R_2 connected between OUT and ADJ, ADJ and GND respectively of the voltage regulator determine the output voltage. $V_{OUT} = 1.25 \cdot (1 + \frac{R_2}{R_1})$ is given by the datasheet [4]. The operating voltage of the microcontroller is $3.3\,\mathrm{V}$, so if $R_1 = 121\,\Omega$, then $V_2 = 1.25 \cdot (1 + \frac{R_2}{121}) = 3.3\,\mathrm{V} \iff R_2 = 198\,\Omega$.

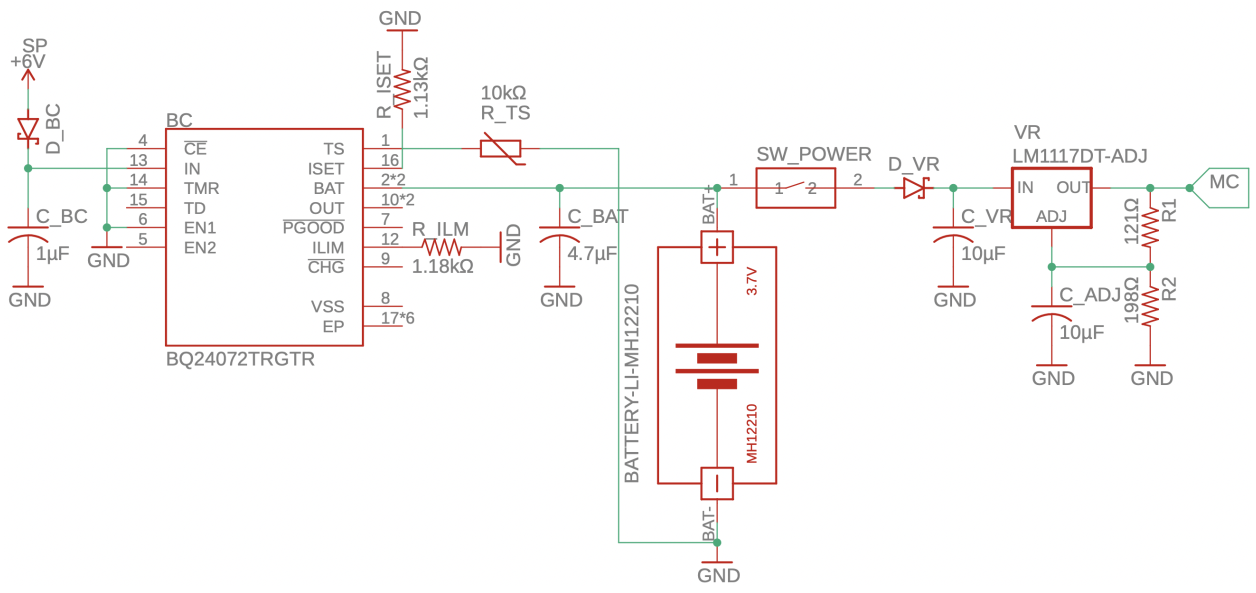

Figure 4: Power Supply Schematic

2.5.2 Control Unit

Figure 5 is a schematic of all components other than those concerned with the power supply. The resistors and the capacitors are suggested from the datasheet [5]. The power being supplied to the vibration motors and LED would come from the microcontroller itself, rather than the power supply. The vibration motors and LED are connected using the IO pins on the microcontroller. USB–UART is used for the programming of the microcontroller from a computer. Assuming that the program is properly written, this device can be detached from the chip. IO0 that is used as a BOOT will flash the microcontroller when the switch is pressed. EN acts as the RESET pin, but since we do not need to reset our controller once deployed, this can be ignored. JTAG is for debugging the PCB, which will allow testing other components connected to the microcontroller without the physical access. If all the components are working properly, JTAG can also be detached.

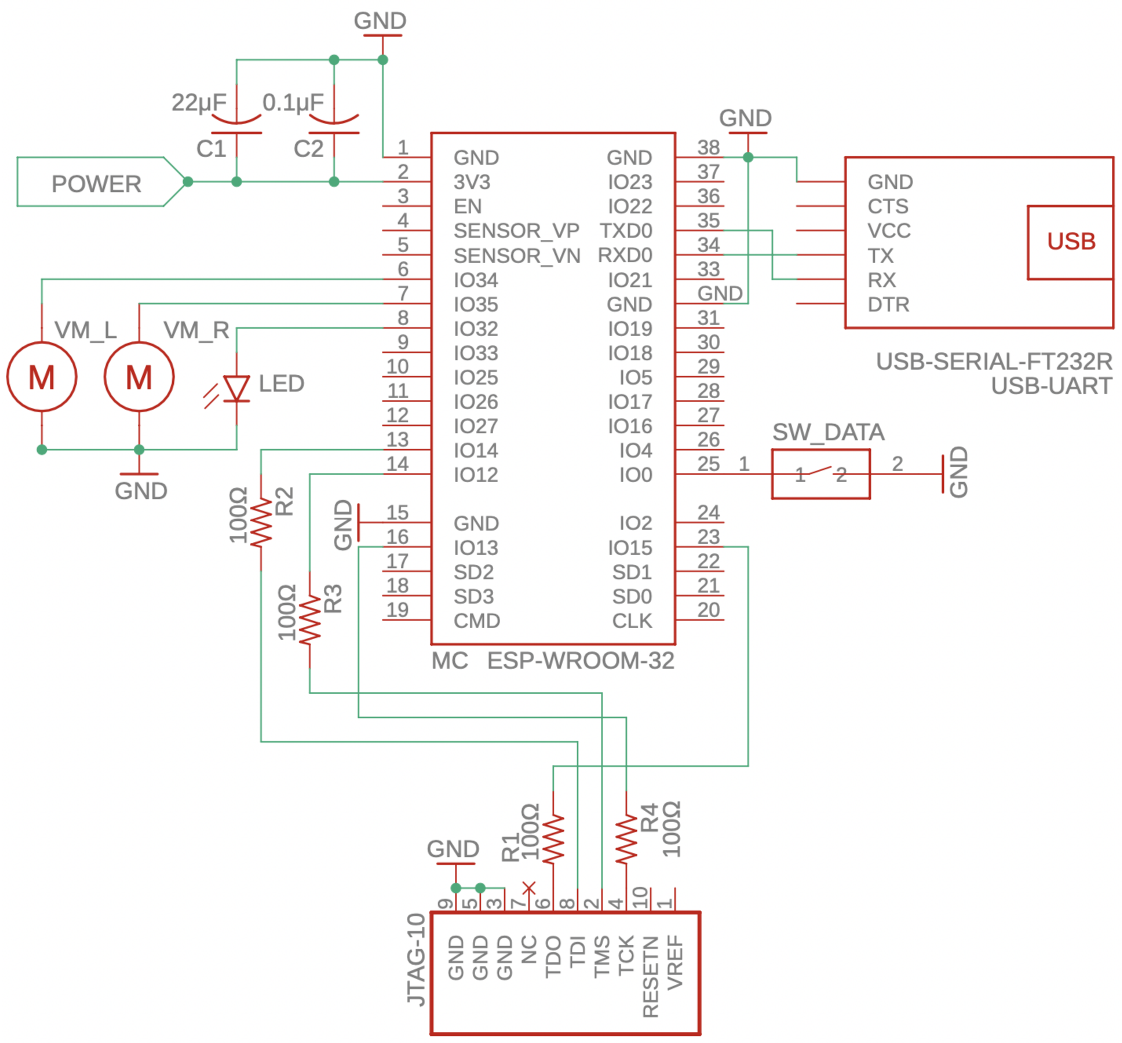
GND
22μF
0.1μF
C1
C2
POWER
VM_L
VM_R
LED
M
M
GND
R2 100Ω
R3 100Ω
GND
GND 3V3 EN SENSOR_VP SENSOR_VN IO34 IO35 IO32 IO33 IO25 IO26 IO27 IO14 IO12 GND IO13 SD2 SD3 CMD
GND IO23 IO22 TXD0 RXD0 IO21 GND IO19 IO18 IO5 IO17 IO16 IO4 IO0 IO2 IO15 SD1 SD0 CLK
MC ESP-WROOM-32
GND
GND CTS VCC TX RX DTR
USB
USB-SERIAL-FT232R
USB-UART
SW_DATA
GND
R1 100Ω
R4 100Ω
GND
JTAG-10
GND GND GND NC TDO TDI TMS TCK RESETN VREF

Figure 5: Control Unit Schematic

2.6 Tolerance Analysis

One important block that we need to verify the functionality is the power block. Without this block, nothing in our project will function properly. For our project, we aim to have an hour and a half of usage from the battery alone assuming that the solar panel is not providing any power to the system. For us to achieve this time rating, we need to take into consideration the mAh of the battery along with the current draw of the components. To ensure that the battery life of the device will be at least an hour and a half, we will need to use the equation: $\frac{\text{Battery Capacity (mAh)}}{\text{Current Draw (mA)}} = \text{Battery Life (h)} \iff \text{Battery Capacity (mAh)} = \text{Current Draw (mA)} \cdot \text{Battery Life (h)}$. Because of this, we will indeed calculate the current draw of our system. The ESP32 microcontroller needs $500\,\mathrm{mA}$ to run and cannot exceed $1200\,\mathrm{mA}$. The vibration motors each can use at most $80\,\mathrm{mA}$ and the LED will use $20\,\mathrm{mA}$. This means that we will need $1200 + 80 \cdot 2 + 20 = 1380\,\mathrm{mA}$ to power our device worst case. Best case we will need $500 + 80 \cdot 2 + 20 = 680\,\mathrm{mA}$. To better estimate our usage we will also take the average of $\frac{1380+680}{2} = 1030\,\mathrm{mA}$ for our calculations.

Now that we have the current draw for our best case, worst case, and average case we can begin to look at how large of a battery we will need for the device to function for an hour and a half. Plugging the value that we equation that we derived earlier we get that for the worst case we will need $1380\,\mathrm{mA} \cdot 1.5\,\mathrm{h} = 2070\,\mathrm{mAh}$. Similarly, we will need $1020\,\mathrm{mAh}$ for our best case and $1545\,\mathrm{mAh}$ for our average case. Real world batteries never achieve their advertised values though so we will need to take into account this discrepancy. We will assume a $25\,\%$ – $30\,\%$ difference between the advertised and actual value of a battery [6]. With this assumption of a 25% reduction, it means that worst case we will need $\frac{2070\,\mathrm{mAh}}{75\,\%} = 2760\,\mathrm{mAh}$. Best case we will need $1360\,\mathrm{mAh}$ and our average will be $2060\,\mathrm{mAh}$. Because of this, we have chosen to use a $2500\,\mathrm{mAh}$ battery for our project. This was selected mainly because of its nice fit between covering 2 of our cases while still being affordable. More importantly, however, we will not be using the WiFi capabilities of the ESP32. Due to this, we will require less power than the absolute max and therefore are more than comfortable with our battery decision.

This decision is even further justified with our inclusion of a solar panel on the device. The solar panel is rated for an output of $530\,\mathrm{mAh}^{-1}$. To calculate the tolerances, we will look at several possible power outputs. The first that we will look at is the option in which the solar panel can provide the rated $530\,\mathrm{mA}$ but it is only able to provide this power one time, meaning that after it delivers $530\,\mathrm{mA}$ to the system, it does not contribute any more power. The total battery life for this case would then be $\frac{2500+530}{1380} = 2.2$ hours for our worst use case and optimal battery efficiency. The other option would be that the solar panel can deliver the $530\,\mathrm{mA}$ each hour. This would mean that for our worst case we would be able to gain an additional $530 \cdot 1.81 = 959.3\,\mathrm{mAh}$. This would bring our total time to $\frac{2500+959.3}{1380} = 2.5$ hours. Similarly, we can carry these calculations out for all possible scenarios and to determine the expected battery life based on the given constraints. The results of these calculations can be seen in Figure 5 and Figure 6 as well as Table 11 and Table 12.

As seen in Table 11 and Table 12 there is a large range in which our battery could potentially last. In the best of the best cases, we have the battery lasting 6.54 hours and in our absolute worst of the worst cases, we have the batter lasting 1.36 hours. Despite this, we are confident that our chosen battery size of $2500\,\mathrm{mAh}$ is enough to satisfy our need for an hour and a half of battery life. This is largely due to our predicted current usage will be equal to or less than the average current draw that we calculated. For every case of the average current draw, we were able to achieve the goal of an hour and a half. Our worst performing time with the average current draw was 1.82 hours which meets our requirement with room to spare.

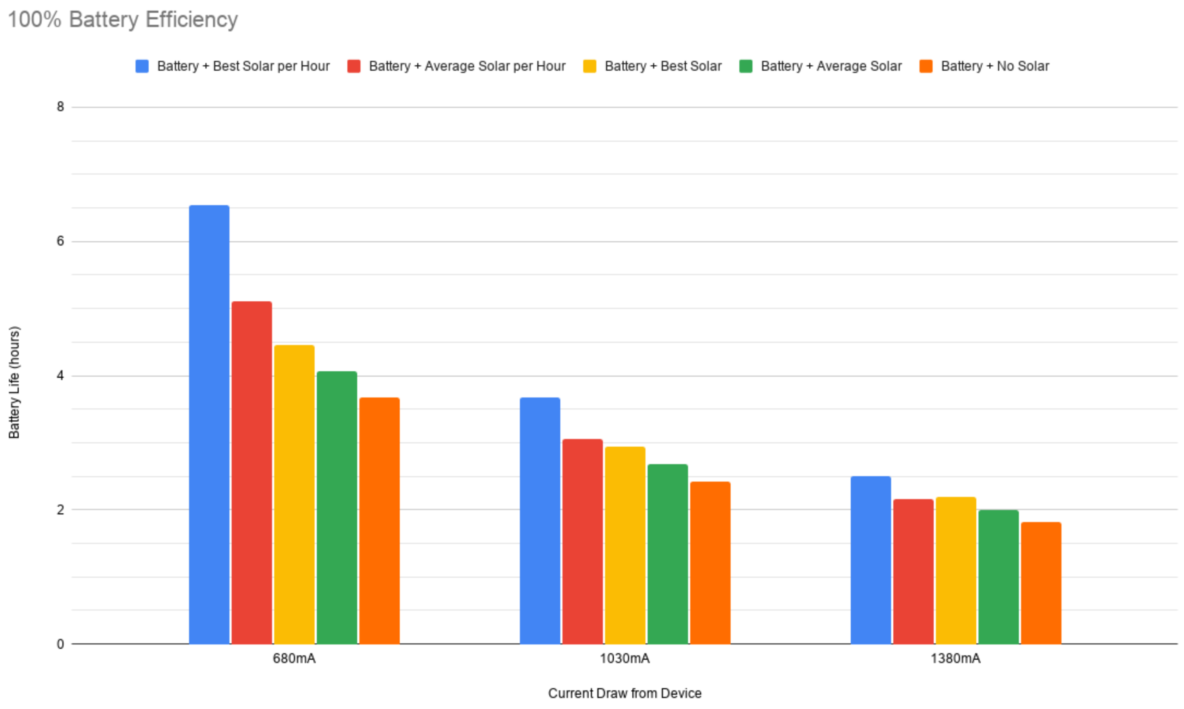

Figure 6: 100% Battery Efficiency

	Best Case (630 mA)	Average Case (1030 mA)	Worst Case (1380 mA)
Battery + Best Solar per Hour	6.54	3.68	2.51
Battery + Average Solar per Hour	5.11	3.05	2.16
Battery + Best Solar	4.46	2.94	2.20
Battery + Average Solar	4.07	2.68	2.00
Battery + No Solar	3.68	2.43	1.81

Table 2: 100 % Battery Efficiency Predicted Time Usage (values in hours)

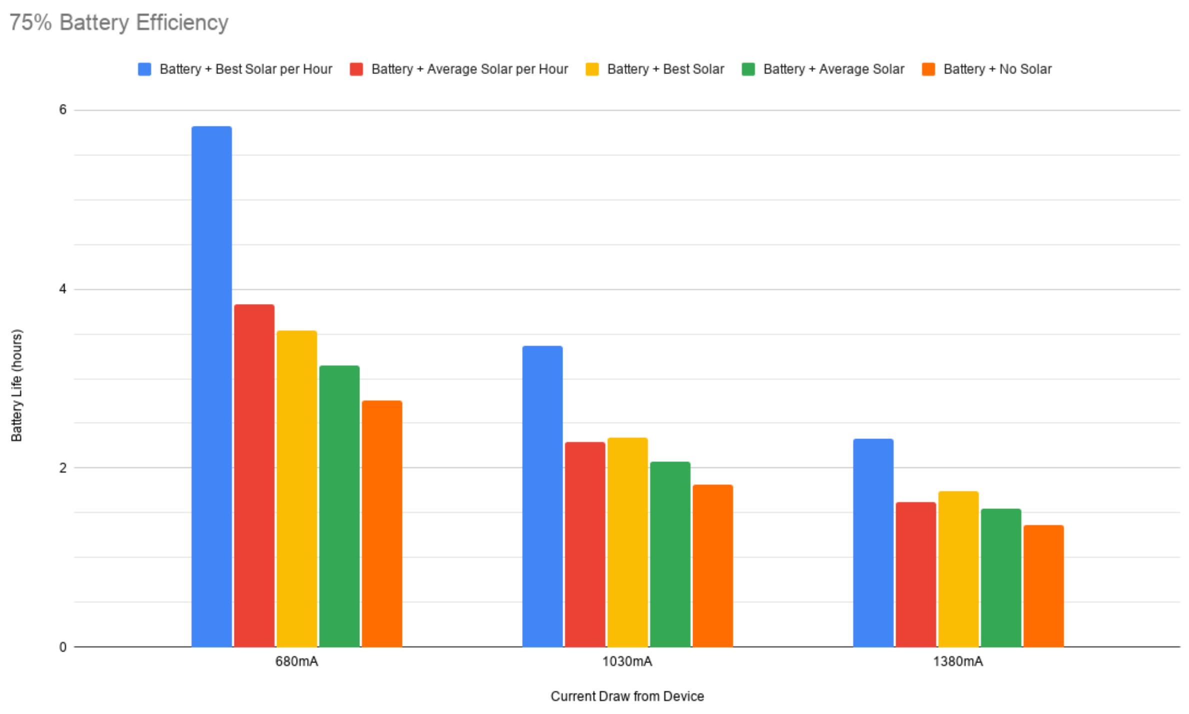

Figure 7: 75% Battery Efficiency

	Best Case (630 mA)	Average Case (1030 mA)	Worst Case (1380 mA)
Battery + Best Solar per Hour	5.83	3.36	2.33
Battery + Average Solar per Hour	3.83	2.29	1.62
Battery + Best Solar	3.54	2.33	1.74
Battery + Average Solar	3.15	2.08	1.55
Battery + No Solar	2.76	1.82	1.36

Table 3: 75% Battery Efficiency Predicted Time Usage (values in hours)

3 Cost and Schedule

3.1 Cost Analysis

3.1.1 Labor Cost

According to UIUC, the average starting salaries for graduates in Computer Engineering and Electrical Engineering are $\$96,992$ and $\$79,714$ respectively [7]. Hence, the average salary of the ECE graduate would be $\frac{\$96,992+\$79,714}{2} = \$88,353$. Assuming that the average engineer works 40 hours per week for 50 weeks per year, one would be making $\frac{\$88,353}{40 \cdot 50} = \$44.18$ per hour. One person will spend approximately 100 hours on the project; hence, the total cost of labor would be $\$44.18 \cdot 100$ hours $\cdot\, 3$ people $\cdot\, 2.5 = \$33,135.00$.

3.1.2 Component Cost

Part	Seller	Link	Quantity	Unit Price ($)	Total ($)
ESP32-WROOM Development Board	Amazon	Part, Guide	2	7.50	15.00
ESP32-WROOM Microcontroller	Adafruit	Part, Datasheet	1	8.95	8.95
CP2102 USB-UART TTL Module	Amazon	Part, Datasheet	2	3.25	6.50
SWD JTAG	Adafruit	Part	1	4.90	4.90
Solar Panel	Adafruit	Part, Datasheet	1	45.00	45.00
BQ24074 USB/DC-Battery Charger	Adafruit	Part, Datasheet	1	9.95	9.95
Li–Ion Battery	Adafruit	Part, Datasheet	1	14.95	14.95
LM1117 Voltage Regulator	Amazon	Part, Datasheet	10	0.80	8.00
KAE01SGGT Switch	Digi-Key	Part, Datasheet	1	1.00	1.00
APT1608ZGC LED	Digi-Key	Part, Datasheet	2	0.75	1.50
1020-15-003-011 Vibration Motor	Digi-Key	Part, Datasheet	4	1.20	4.80

Table 4: Component Cost

3.1.3 Total Cost

Section	Cost
Labor Cost	\$33,135.00
Part Cost	\$120.55
Total Cost	\$33,255.55

Table 5: Total Cost

3.2 Schedule

Week	Kevin	Soo Min	Stephen
3/1	Continue working on the design document and start Mapbox API research for a navigation software app.	Search for components acceptable for the design. Design the power and controller unit for PCB.	Work with Soo Min to find components the will work together with the tolerances needed.
3/8	Finish implementing Mapbox API into our android app and test Bluetooth connection between phone and wearable.	Check actual tolerance of each component and its voltage and current requirement. Determine the value of resistors and capacitors.	Work on programming the ESP32 and unit testing it for the different requirements that it needs.
3/15	Write all functions for sending data to and from the mobile app and ESP32 chips and start prototyping PCB.	Finalize the PCB design to submit. Add additional components on the board if required.	Continue working on the programming of the ESP32. Also, help Soo Min implement the circuit.
3/22	Integrate and debug software made with Soo Min and Stephen's hardware components.	Test if the PCB is properly designed by powering and running simple programs. Debug and update the design if necessary.	Work with Kevin to integrate the hardware in with the software.
3/29	Add any features needed in software that we had previously missed or recently thought of.	Test the power supply unit of the device and record the voltage supply difference of the solar panel under different conditions.	Continue to help debug the hardware and software.
4/5	Assist with any hardware and software debugging that is still needed.	Debug the device and start test-running the device outside.	Finalize project for the final presentation and begin working on the final report.
4/12	Finalize all progress with Soo Min and Stephen and do final tests of our product.	Gather all obtained data to write the final report.	Finish working on the final report.

Table 6: Schedule

4 Ethics and Safety

In our final product, we will have circuits in our wearable, so it is necessary for us to address some of the safety concerns that might arise to protect our customers as well as our product itself. During use, we will have a battery and solar panel attached to our product. One safety concern could be the possibility of these components overheating and without proper cooling, we could run into trouble of burning the customer. Therefore, to mitigate this we will have an automatic shut off sequence in place where if the unit gets too hot, the product will automatically shut off to ensure no one is injured while using this product. This specifically will address IEEE ethic code II.9 [8].

Aside from the power units, we also will have vibration motors attached to the product to give haptic feedback to the users. However, this could also potentially be a safety concern as vibrations that are too intense could easily cause injuries or massive discomfort. Therefore, we will ensure to test our product intensively in order to ensure the maximum vibrations will not be too intense and also give users an option in our mobile app to let users manually adjust intensities as well.

We will test our final design intensively and ensure to prevent any potential safety hazards and use of dangerous components. We will also abide by IEEE ethic code I.5, by taking into consideration the feedback from others and using them to improve our design safety, we are certain we can produce a product both safe and functional [8]. Lastly, we will treat all of our members and testers with respect and never engage in any harassment or discrimination of others.

References

[1] Winnie Hu. (2020). "A Surge in Biking to Avoid Crowded Trains in N.Y.C.," [Online]. Available: https://www.nytimes.com/2020/03/14/nyregion/coronavirus-nyc-bike-commute.html. [Accessed: 4-Mar-2021].

[2] H. Staples. (2013). "Bike Accidents and Phone use - Is There a Link?" [Online]. Available: https://www.holland-cycling.com/blog/291-bike-accidents-and-phone-use-is-there-a-link#:~:text=The\%20group\%20of\%20cyclists\%20most,into\%20a\%20tree\%20or\%20lamppost. [Accessed: 02-Mar-2021].

[3] Texas Instrument. (2019). "BQ2407x Standalone 1-Cell 1.5-A Linear Battery Charger with Power-Path," [Online]. Available: https://www.ti.com/lit/ds/symlink/bq24074.pdf?ts=1614809395816&ref_url=https\%253A\%252F\%252Fwww.ti.com\%252Fdocument-viewer\%252FBQ24074\%252Fdatasheet\%252Ffeatures-slus8103327. [Accessed: 01-Mar-2021].

[4] Texas Instrument. (2020). "LM1117 800-mA, Low-Dropout Linear Regulator," [Online]. Available: https://www.ti.com/lit/ds/symlink/lm1117.pdf?ts=1614853300118&ref_url=https\%253A\%252F\%252Fwww.ti.com\%252Fstore\%252Fti\%252Fen\%252Fp\%252Fproduct\%252F\%253Fp\%253DLM1117T-2.5\%252FNOPB. [Accessed: 02-Mar-2021].

[5] Espressif Systems. (2021). "ESP32-WROOM-32," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. [Accessed: 01-Mar-2021].

[6] UIUC. (2019). "Salary Averages," [Online]. Available: https://ece.illinois.edu/admissions/why-ece/salary-averages. [Accessed: 28-Feb-2021].

[7] IEEE. (2020). "IEEE Code of Ethics," [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8. [Accessed: 15-Feb-2021].

[8] Banggood. (2020). "The Secret Marketing Trick behind Powerbank Capacity," [Online]. Available: https://blog.banggood.com/the-secret-marketing-trick-behind-powerbank-capacity-29982.html. [Accessed: 4-Mar-2021].