Cheap, Accurate, and Privacy Preserving Contact Tracing Chip

ECE 445 Design Document

Anshul Sanamvenkata, Abhinav Singh, Kapil Kanwar

Group 64

TA: Ali

1 Introduction

1.1 Problem and Solution Overview

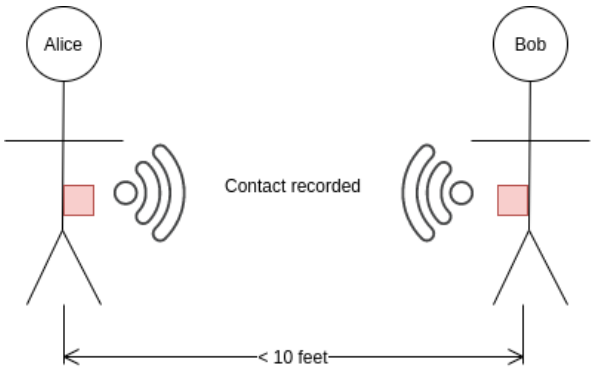
COVID is a deadly and highly infectious disease, and given the current trend of globalization and environmental destruction, such pandemics will only become more common. Testing and contact tracing are one of the best ways to fight a highly infectious disease while allowing people to maintain some semblance of a normal life. Contact tracing is rapidly becoming a heavily adopted way of fighting pandemics and is being used by the CDC in the United States [1].

Current contact tracing solutions either rely on manual effort, or mobile apps, which are both flawed. Manual methods typically involve calling someone who has tested positive and asking them to recall whom they met, which obviously is highly imperfect, since people oftentimes provide insufficient information, and many contact tracers must be hired [2]. Mobile contact tracing apps, although a great improvement over manual contact tracing, still have serious flaws. Apps that use GPS suffer from the fact that GPS is not always available and also quite inaccurate, not to mention the privacy concerns of mass surveillance of everyone's locations. Apps that use NFC, or Bluetooth, to address the privacy and availability concerns of GPS, still fall short. In the case of NFC, the range is far too small, and in the case of Bluetooth, the ability to measure distance accurately is sorely lacking, which inevitably leads to high false positive rates [3]. Finally, modern smartphones are simply too expensive in many parts of the world, and few people have sufficiently sophisticated smartphones that can perform effective contact tracing.

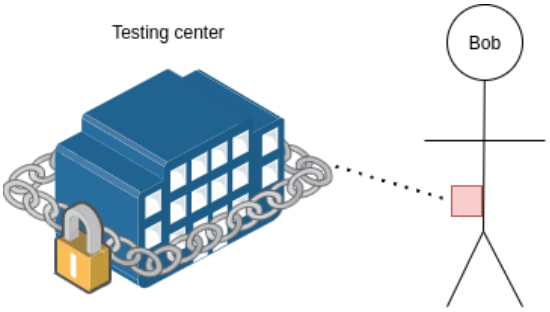
We propose a small, cheap chip that can be easily carried which will automatically communicate with other nearby chips over ultra-wideband (UWB) to perform contact tracing, detecting potential transmissions of up to 10 feet away (adjustable depending on the nature of the pandemic it is being used for). Testing centers will upload cryptographically signed COVID status messages onto people's devices to prevent malicious false status attacks, and finally, it must be regularly docked with a PC with Internet access to charge and upload contact information to a server. When the server identifies a potential contact, the PC receives a notification telling the user to quarantine.

1.2 Visual Aid

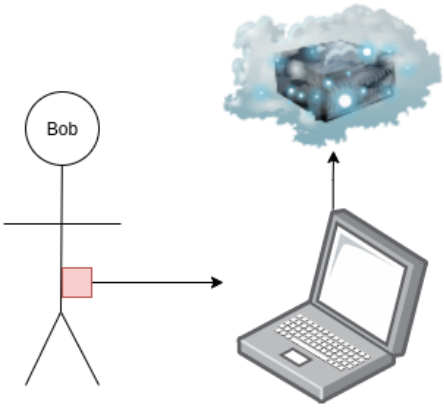
The following series of images demonstrates how the device is intended to be used, and how it can effectively detect and notify users of potential transmissions.



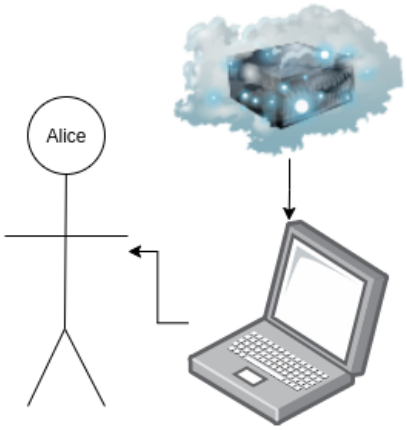
Alice and Bob's chips detect each other within 10 feet, and record each others' anonymous IDs in their internal storage.



Bob receives the bad news from a trusted testing center that he has tested positive. The testing center connects to Bob's chip and uploads a cryptographically signed message with his positive status.



Bob connects his device to his computer to charge it and to upload his positive status to a server.



Alice receives the bad news immediately from the server that she was in contact with someone who was positive. She must now follow whatever public health policies have been set by her local government.

1.3 High Level Requirements

- The chance of a false negative, which is defined as the device failing to record a contact despite two users being less than 10 feet apart, must be less than 25%. The chance of a false positive, which is defined as the device recording a contact despite two users being more than 10 feet apart, must be less than 25%.

- The device must be capable of operating for at least 12 hours without having to be charged.

- The device must fit within the volume of a wallet, which we define as 3.5” x 4.5” x 1.0”

2 Design

2.1 Block Diagram

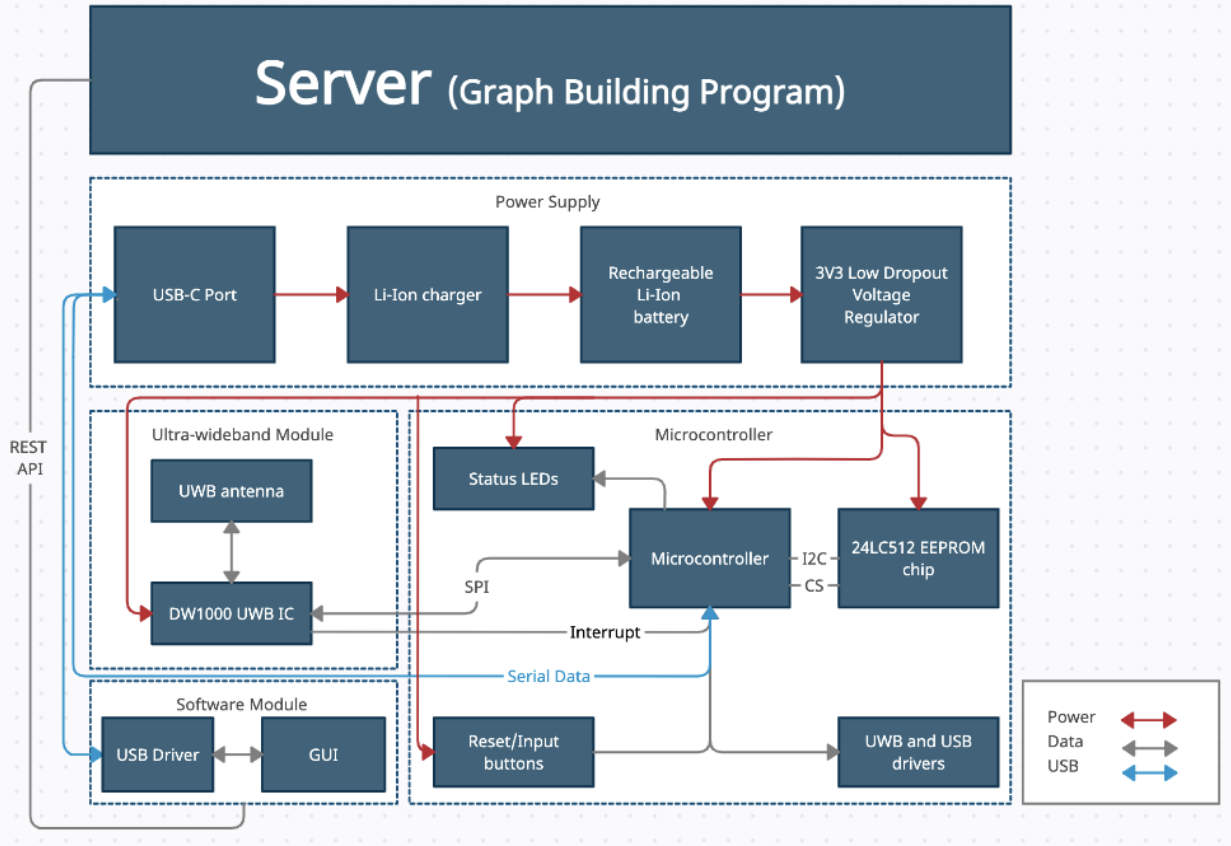


Figure 1: Block diagram of the entire system

The primary functionality of the device is in the microcontroller. Obviously, the microcontroller requires power, which is provided by the power module. We plan on using a 1300mAh battery and based on the datasheets, the UWB module should take ~70mA in its most power-intensive mode and the Arduino should take 11.3mA, which clearly fits within our 12 hour limit [4]. Then, the microcontroller communicates with the UWB module to talk to other contact tracing devices, and the UWB chip also provides the distance information required to decide whether or not a contact has occurred. Repeated measurements will ensure the first high level requirement of at most 25% false positives and negative is met. The EEPROM chip is used to store this contact information, and finally the microcontroller communicates this stored information with the software module over USB. To ultimately determine whether or not the user must quarantine, the software module communicates with the server, which is responsible for constructing a full contact graph and finding connected nodes. For the final high-level requirement, it is sufficient to just realize that Arduinos, the UWB module, and the Li-po battery pack do not take a lot of time.

2.2 Physical Design

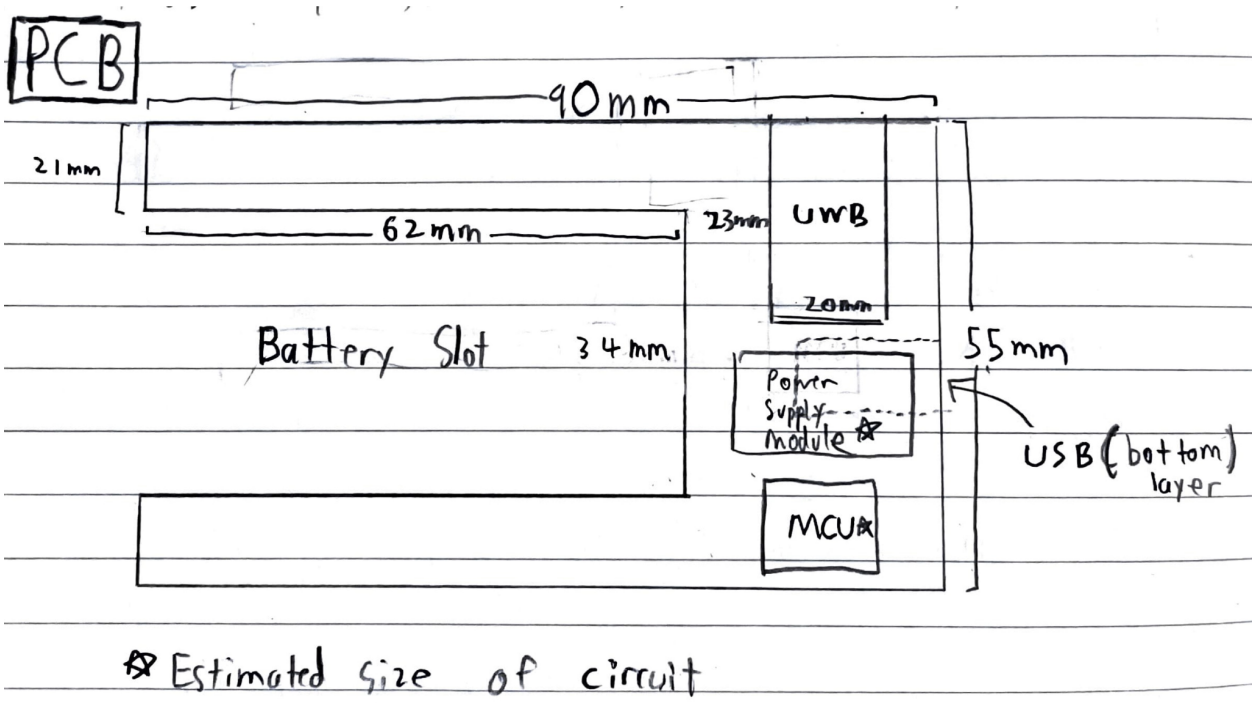


Figure 2: Physical design of PCB

We plan on putting all of the physical components on a single PCB as pictured above. Clearly, this fits within the requirements that we listed, and ideally we would also put this PCB within some sort of simple plastic casing.

2.3 Ultra-wideband Subsystem

This subsystem contributes to the overall system by allowing our contact tracing card to be able to communicate and find the distance between other contact tracing cards in the area. We chose to use UWB instead of other similar technologies because UWB is capable of more accurate distance measurements [5]. The UWB subsystem interfaces with the microcontroller subsystem through SPI and is powered through the stable power supply from the power subsystem. This subsystem is by far the most difficult to implement due to UWB being such a new technology with little documentation and support.

The specific chip we will use is the DWM1000 by Decawave. It has a power draw of 70mA while transmitting, and 30mA while receiving, which puts it well in the range of our 12 hour battery life [5]. It has a maximum range of 290 meters, which is of course sufficient, and it also has a precision of 10 cm [5]. These are very reasonable values to be able to distinguish contacts on the order of 10 ft. We will further detail this subsystem in the Tolerance Analysis section.

Requirement	Verification
Determine distance between two separate modules with a precision of at least 10cm so that false positives and false negatives are reduced.	Equipment: 2 Arduinos, yardstick 1. Connect two separate UWB modules to two separate Arduinos acting as mock objects 2. Upload test firmware to send simple data packets back and forth 3. Separate the two nodes by 2 feet using the yardstick to measure 4. Measure the distance according to the UWB modules 5. Verify that the error is less than 10cm, and repeat steps 3-5 by increments of 1 foot until reaching 15 feet This will be presented as a table of values with columns: distance, absolute error, and percent error

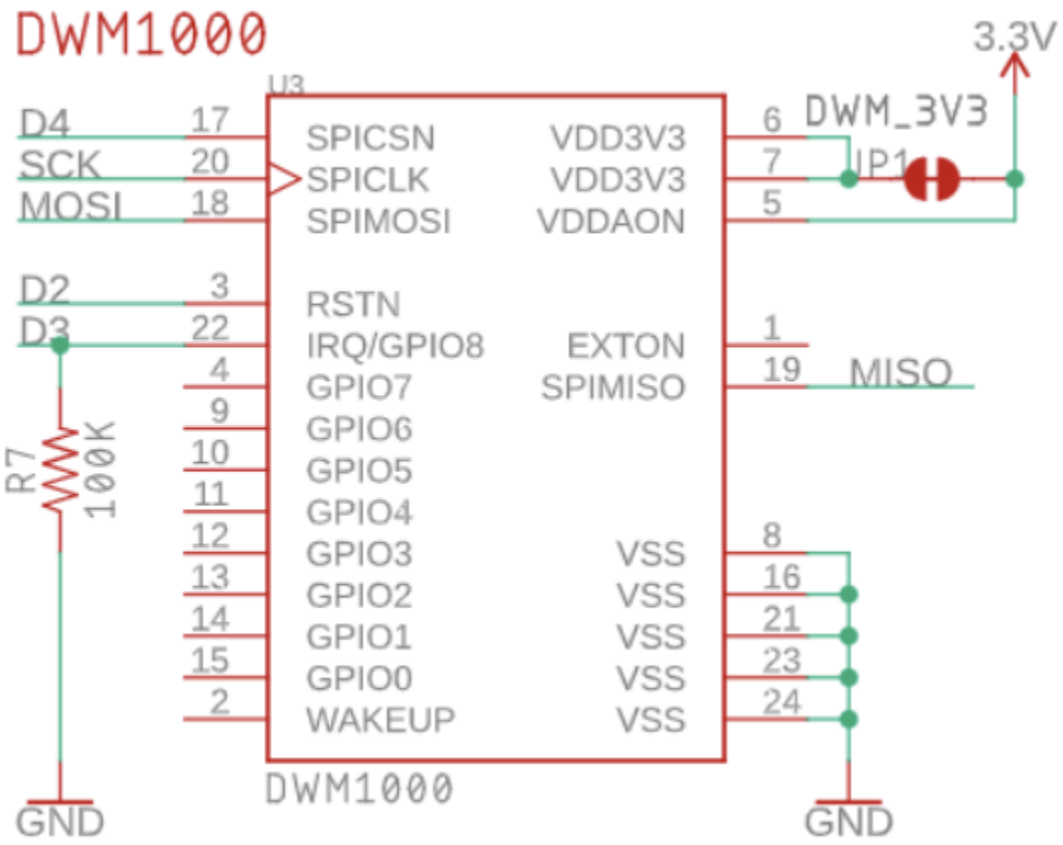


Figure 3: UWB subsystems schematic

2.4 Microcontroller Subsystem

This subsystem acts as the main processing of the whole contact tracing system and is responsible for initiating communication with other contact tracing systems, as well as communicating contact information with the software subsystem. It leverages the SPI interface to the UWB chip to scan and discover other chips in the vicinity and perform the appropriate cryptographic verification and storage of user information. It also uses the built in USB capability to communicate with the software. We are planning on using a ATmega32U4 as the microcontroller for this subsystem, and in addition, we will be using the 24LC512 EEPROM to persist contact data.

The ATmega32U4 has a 32 KB program memory size, which should be sufficient for all the chip needs to do: repeated UWB measurements, basic cryptographic verification, and storage of contact data. 2,560 bytes of SRAM are also included, which again should be sufficient, because the program itself is not very complex [6]. More importantly, it has SPI interfaces to be able to communicate with the UWB subsystem, and a USB module to communicate with the software subsystem [6]. As for the 24LC512 EEPROM, all we need to store is contact data. Assuming a person comes in contact with 5000 people in a day, which (we hope) is a gross overestimate, and a cryptographically signed positive message takes at most 100 B, which is also an overestimate, we arrive at 500 KB, which is still within the storage limit of 512 KB [7].

Requirement	Verification
Must determine whether or not a contact occurred between two chips (defined by a threshold of 10 ft) with 75% accuracy	Equipment: yardstick 1. Modify the software to blink a light if a contact has occurred 2. Separate two nodes by 2 feet 3. Turn the device on and allow it to check for contacts 4. Record whether or not a contact was detected 5. Repeat steps 2-4 ten times to get a percentage accuracy 6. Repeat steps 2-5 with different distance values by increments of 1 foot up to 18 feet This will be presented as a table of values with columns of distance, and percentage correct (contact or no contact)

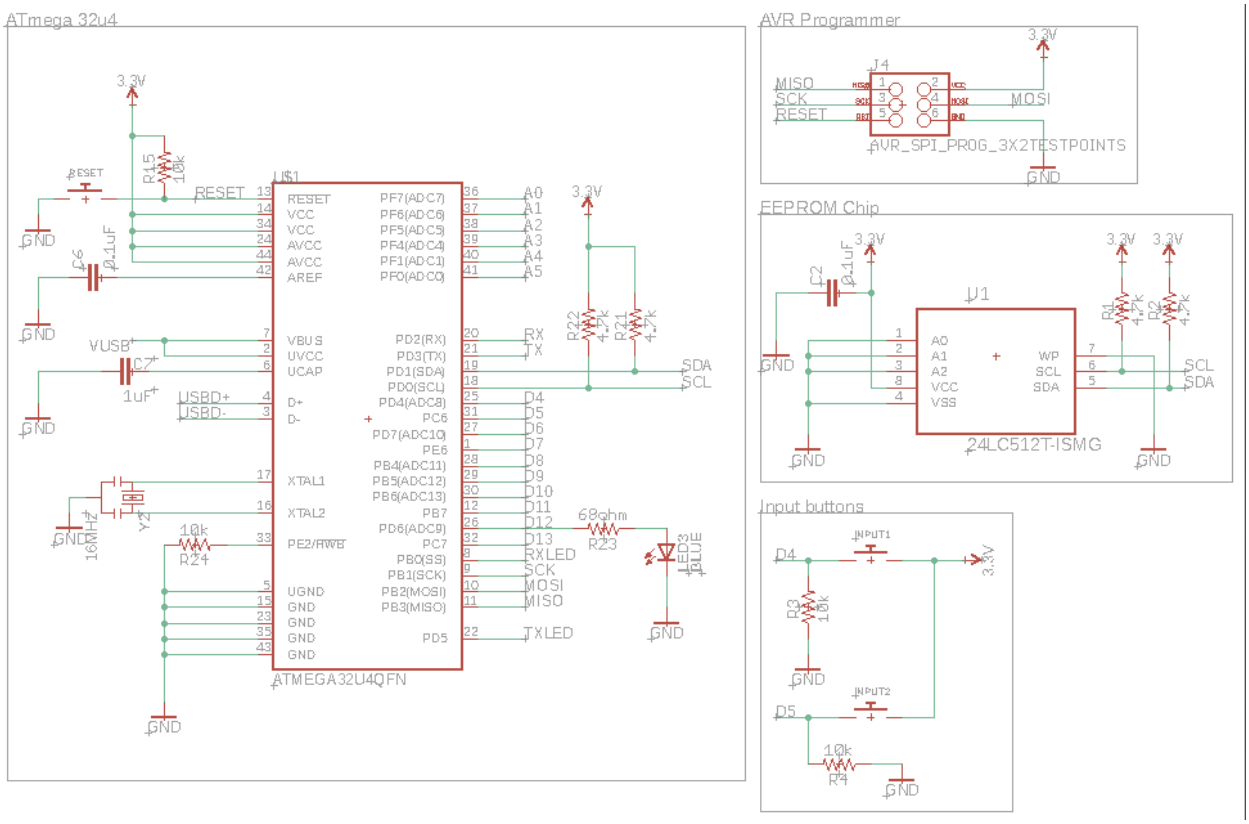


Figure 4: Microcontroller subsystem schematic

2.4 Power subsystem

This subsystem is responsible for managing the battery and utilizing USB power to safely recharge the lithium battery. It uses the concept of balance charging. This process will check the voltages of each individual cell in the battery and ensure they all have the same voltage ensuring battery health and safe recharging by charging in parallel. This is important due to the volatility of lithium batteries. We expect a roughly full day of usage with our system. The power subsystem will be using small rechargeable li-ion batteries with an average capacity of 1500mAh. We see from the UWB datasheet that the nominal power consumption is 70mA and the nominal power consumption of an Arduino type microcontroller is approximately 11.3mA. With these calculations we can estimate nearly 18 and a half hours of power at a time.

Must operate normally for up to 12 hours	Equipment: eyes 1. Charge the battery fully as described below 2. Turn the device on and wait 12 hours 3. Check that the device does not power off This will be recorded as a simple success / failure

Must safely charge to full 4.2V capacity within 3 hours	Equipment: LED, any device capable of providing power over USB 1. Discharge lithium ion battery to 3.7V 2. Charge the battery from MCP73833 Li-Ion charging IC with USB input 3. Ensure that the battery is charged at the end of 3 hours by connecting the LED to the pin which indicates that the battery is fully charged We will record this as a simple success / failure.
The output of the voltage regulator maintains a constant 3.3V with a tolerance of 0.1V for the course of the 12 hours that the device should be able to operate	Equipment: voltmeter 1. Fully charge the battery as detailed above 2. Connect the voltmeter to the output of the voltage regulator while the device is powered on 3. Record the value of the voltmeter every 20 minutes over a 12 hour period 4. Ensure that the value stays within the allowed range This will be presented as a line graph of the recorded voltage values over the 12 hour period with horizontal bars around 3.3V indicating the 0.1V tolerance.

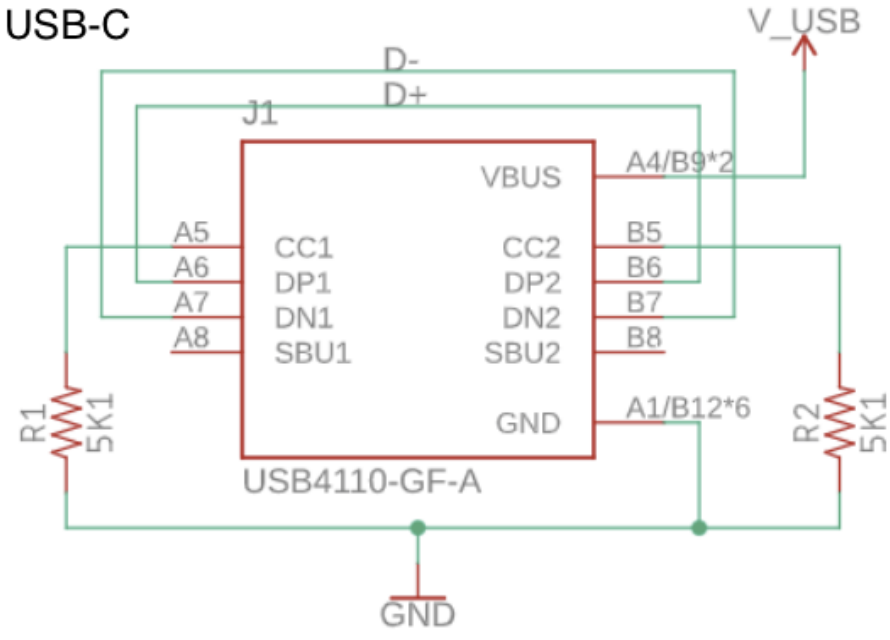


Figure 5: Schematic of USB Component in Power Subsystem

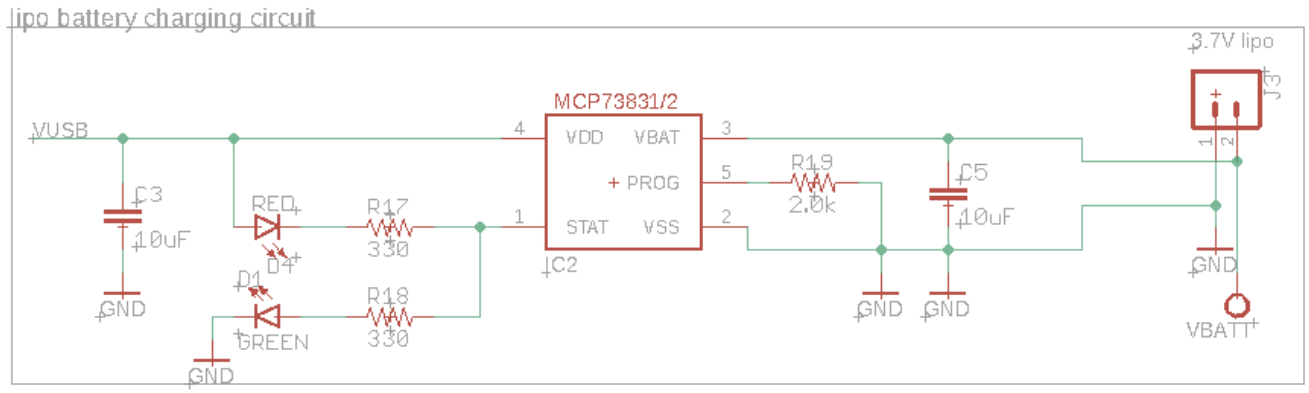


Figure 6: Schematic of Battery Charging Circuit in Power Subsystem

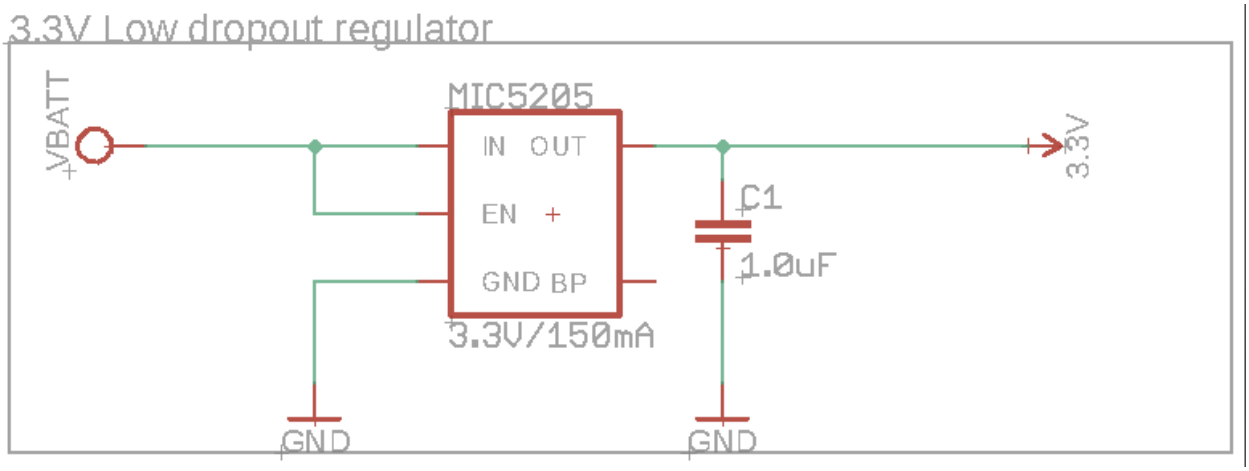


Figure 7: Schematic of 3V3 Voltage Regulator Circuit in Power Subsystem

2.5 Software subsystem

This subsystem is responsible for interfacing with the chip over USB, loading contact data, uploading it to the server, and receiving notification of potential transmissions from the server. As part of this subsystem, we will have to design a device driver to interface with the chip over USB. We require that this driver will be able to transfer all contact data between the chip and the computer within 1 second, after the OS-specific device initialization has occurred, which we require must take at most 10 seconds. This is so that if a shared computer is being used in a location where computers are not as abundant, many people can quickly upload their information.

Additionally, we will create a simple GUI application that users can use to view potential contacts and upload contact data. In order to interface with the server, a simple REST API will be used. This piece is the key component that feeds the server with the edges of the contact

graph.

Requirement	Verification
Download and upload contact data via USB with at most 10 seconds of device initialization, and at most 1 second of time spent uploading	Equipment: computer and USB cable 1. Initialize a chip with a couple of fake contacts 2. Connect the chip to a computer 3. Successfully pass through OS-specific device initialization in at most ten seconds 4. Load contact data via the GUI application and ensure that it completes within one second 5. Ensure that the correct data is loaded We will record the amount of time it takes to pass through device initialization and the amount of time it takes to load the contact data. Additionally, we will record whether or not the correct information is uploaded

2.5 Server subsystem

The server is responsible for getting the edges of the contact graph from the software subsystem, as well as positive COVID statuses. Given this information, it finds all nodes that are connected to positive users and sends a notification to their PCs via the software subsystem. This subsystem is fairly simple and does not have any really non-trivial quantitative requirements, other than those which would arise at scale, which is outside of the scope of this project anyway.

Requirement	Verification
Maintain a graph of anonymous contacts and send messages to users when in contact with an infected user. Must not send a message to user unless they came into contact with an infected user.	Equipment: 3 contact tracing chips 1. Mark two of the three chips as negative, and one as positive 2. Bring the two negative chips within 10 ft of each other and ensure that a contact is recorded 3. Upload all chips' contact data to the server and ensure that no message is sent to the two negative users 4. Bring one negative chip within 10 ft of the positive chip and ensure that a contact is recorded 5. Upload all chips' contact data once more and ensure that all users receive a message within 10 seconds

	We will record all the binary yes / no checks in the steps above, as well as the amount of time it takes for the users to receive a message.

2.6 Tolerance Analysis

In order to ensure that we keep the chance of false negatives and false positives below 25%, we must perform some statistical analysis. Treating the UWB chip as a black box which can return an estimated distance value based on some probability distribution, we can determine the number of repeated measurements necessary to achieve the desired rate of false reports. In order to minimize power consumption to reach our battery life requirement, it is critical that we keep the number of repeated measurements as low as possible.

Define $P_d(x)$ as the probability density function over the estimated distance value x given an actual distance of d. Define T as the contact threshold, which in this case is 10 feet. Define n as the number of measurements we must make before deciding whether a contact occurred or not. Finally, define ρ as the desired maximum failure probability, which is 25%.

We can make the following reasonable simplifying assumptions on the probability distribution, given, without loss of generality, a < b

Assumption 1 - If $x_1 > b$

$$\int_{x_1}^{x_2} P_a(x)\,dx < \int_{x_1}^{x_2} P_b(x)\,dx$$

Assumption 2 - If $x_2 < a$

$$\int_{x_1}^{x_2} P_a(x)\,dx > \int_{x_1}^{x_2} P_b(x)\,dx$$

In plain English, this means that if the actual distance goes up from a to b, the probability density for the estimated distance also shifts upwards, as well as the converse, respectively. This assumption is very basic and likely holds even for inferior alternatives to UWB like Bluetooth, NFC, and audio.

We can state our requirements for false negatives and positives as the following, respectively

- For all $d < T$

$$\left(\int_{T}^{\infty} P_d(x)\,dx\right)^n < \rho$$

- For all $d > T$

$$\left(\int_{0}^{T} P_d(x)\,dx\right)^n < \rho$$

Using **Assumption 1** and **Assumption 2**, respectively, we can conclude the following

$$\forall d < T : \left(\int_T^{\infty} P_d(x)\, dx \right)^n < \left(\int_T^{\infty} P_T(x)\, dx \right)^n$$

$$\forall d > T : \left(\int_0^{T} P_d(x)\, dx \right)^n < \left(\int_0^{T} P_T(x)\, dx \right)^n$$

This means that as long as we can find a value n such that the false positive or false negative rate at $d = T$ stays below ρ, by transitivity, we know that this holds for all d. All that remains to be done, then, is to perform measurements to characterize the probability distribution $P_T(x)$.

Assuming a symmetric distribution (and assuming the variance is small enough that the tail can be effectively ignored), the integral of $P_T(x)$ from 0 to T and from T to ∞ is 50%. Therefore, to achieve a false positive / negative rate of 25%, we simply have to find the smallest integer n such that $50\%^n < 25\%$, which gives a value of exactly $n = 2$. Although the real probability distribution is likely not symmetric, the final result will probably not be far from this estimate.

3 Cost and Schedule

3.1 Cost Analysis

We will have three members working on this project, at an estimated 40 dollars per hour and 10 hours a week occurring over approximately 10 weeks.

3 × $40/hr × 10 hrs / week × 10 weeks × 2.5 = $30000

Part	Cost (prototype)	Cost (bulk)
DWM1000 (UWB)	$17.90	$15
MIC5205 (3V3 Regulator)	$0.45	$0.336
Female USB-C port	$1.34	$0.60
ATmega 32u4 (Microcontrolelr)	$4.39	$3.65
24LC512 (EEPROM)	$1.46	$1.34
MCP73831 (Li-Po charger)	$3.10	$2.80
Resistors, capacitors, crystals, sockets, switches (Digikey; est.)	$8	$4

Li-Po Battery	$10.95	$10.95
PCB	$10	$1
Total	$57.59	$39.68

With three total chips this will lead to a final development cost of 30172.77. Our server costs will be approximately 3.36 dollars for our limited development environment using Amazon EC2 spot servers.

3.2 Schedule

This is an estimate of how we will split our time and effort, but if any part ends up being more difficult than expected, and another part ends up being easier, we will move our collective attention to the more difficult portions.

Week	Abhinav	Anshul	Kapil
3/15	Research and try out communicating with the UWB chip with the microcontroller	Begin work on the PCB design	Research USB protocols and devise communication protocol between device and computer
3/22	Create user GUI to upload and view contacts	Complete the PCB design and get it past audit	Use tolerance analysis to reliably get UWB distance measurements
3/29	Design API between server and GUI	Complete the power subsystem	Implement USB communication on the chip
4/5	Implement server subsystem	Design and begin implementing the protocol the chips use to talk to each other	Implement Linux USB device driver to talk with the chip
4/12	Integrate USB driver with the GUI application	Finish implementing the protocol the chips use to talk to each other	Test and debug any USB issues and ensure reliability
4/19	Prepare Demonstration / Final debugging	Prepare Demonstration / Final debugging	Prepare Demonstration / Final debugging
4/26	Prepare Final Paper	Prepare Final Paper	Prepare Final Paper
5/3	Presentation/Finalize Paper	Presentation/Finalize Paper	Presentation/Finalize Paper

4 Ethics and Safety

A project of this nature has various ethics and safety concerns. Starting with ethics, the primary concern is user privacy and the storage of personal data. As #1 from the IEEE Code of Ethics states, we must “hold paramount the safety, health, and welfare of the public… [and] protect the privacy of others” [8]. In this case, our project aims to satisfy both of these apparently competing goals. Typical contact tracing solutions sacrifice individual privacy to protect the “safety, health and welfare” of others. Our solution, however, stores data using completely anonymous and randomly generated IDs, and so protects user privacy.

Additionally, #9 from the IEEE Code of Ethics states that we must “avoid injuring others, their property, reputation, or employment” [8]. In this case, our solution has the potential to damage others’ employment and personal happiness by forcing them to quarantine or by giving them a false sense of confidence. In order to minimize these risks while maximizing public safety, we have established a minimum probability of false contacts to balance the two competing interests consistently and ethically. We also will prevent the possibility of malicious actors with forged COVID statuses by requiring that COVID statuses be cryptographically signed by trusted testing centers.

Our project also has a few, albeit relatively minor, safety concerns. Since we are using lithium-ion batteries, there is a possibility of fire or explosion under a couple of circumstances: physical damage to the battery, high temperatures above 130°F, and below freezing temperatures during charging [9]. According to OSHA recommendations, in order to minimize the risk of fire or explosions, we must store the batteries in cool, dry locations, avoid physical damage, stop using upon any sign of bulging or high temperature, and remove batteries from the charger once they are fully charged [9]. Additionally, if there is ever a fire or explosion, we must evacuate immediately and contact the fire department [9]. In any case, since we are using a chip that will regulate the charging speed based on the measured temperature as well as set a maximum voltage, the risk of fire or explosion should be very low to begin with [10].

5 Citations

[1] “Contact Tracing Steps - Infographic.” Centers for Disease Control and Prevention, Centers for Disease Control and Prevention, 26 Feb. 2021, www.cdc.gov/coronavirus/2019-ncov/daily-life-coping/contact-tracing-infographic.html.

[2] D. Lewis, “Why many countries failed at COVID contact-tracing - but some got it right,” *Nature News*, 14-Dec-2020. [Online]. Available: https://www.nature.com/articles/d41586-020-03518-4. [Accessed: 01-Mar-2021].

[3] R. Faragher, “The Hidden Trade-Offs Inside Contact-Tracing Apps,” Forbes, 22-Apr-2020. [Online]. Available: https://www.forbes.com/sites/ramseyfaragher/2020/04/21/the-hidden-trade-offs-inside-contact-tracing-apps/?sh=dd085eeea07a. [Accessed: 01-Mar-2021].

[4] D. Christopher, “Guide to reduce the Arduino Power Consumption,” *DIYI0T*, 14-Jan-2021. [Online]. Available: https://diyi0t.com/arduino-reduce-power-consumption/. [Accessed: 05-Mar-2021].

[5] “DW1000 Radio IC,” *Decawave*, 18-Dec-2020. [Online]. Available: https://www.decawave.com/product/dw1000-radio-ic/. [Accessed: 05-Mar-2021].

[6] “ATmega32U4,” ATmega32U4 - 8-bit Microcontrollers. [Online]. Available: https://www.microchip.com/wwwproducts/en/ATmega32u4. [Accessed: 05-Mar-2021].

[7] “24LC512,” *24LC512 - Memory*. [Online]. Available: https://www.microchip.com/wwwproducts/en/24LC512. [Accessed: 05-Mar-2021].

[8] “IEEE Code of Ethics,” *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 01-Mar-2021].

[9] “UNITED STATES DEPARTMENT OF LABOR,” *Safety and Health Information Bulletins | Preventing Fire and/or Explosion Injury from Small and Wearable Lithium Battery Powered Devices | Occupational Safety and Health Administration*. [Online]. Available: https://www.osha.gov/dts/shib/shib011819.html. [Accessed: 01-Mar-2021].

[10] “MCP73831,” *MCP73831 - Battery Management and Fuel Gauges - Battery Management and Fuel Gauges - Battery Chargers*. [Online]. Available: https://www.microchip.com/wwwproducts/en/en024903. [Accessed: 05-Mar-2021].