Cheap, Accurate, and Privacy-Preserving Contact Tracing Chip

By

Abhinav Singh

Anshul Sanamvenkata

Kapil Kanwar

Final Report for ECE 445, Senior Design, Spring 2021

TA: Ali Kourani

5 May 2021

Project No. 64

Abstract

With a current global pandemic spreading around the world, methods to fight off and limit deadly transmissions are crucial. With testing and contact tracing, a semblance of normal life is possible while only those who are potentially infected must quarantine. This method is used by governments all over the world and is highly recommended by the CDC in the United States [1].

Our solution is a simple contact tracing chip that users can carry during daily life. It uses ultra-wideband technology to detect and determine the distance other contact tracing chips to see if two users have come within a disease-specific threshold. It can be plugged into a PC to upload a list of contacts, as well as determine if the user has to quarantine. We plan to use cryptographic methods to ensure that the entity operating the contact tracing server does not have any identifying information on its users.

Contents

1. Introduction

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.

This project is a contact tracing chip using ultra-wideband (UWB). This chip will have several key capabilities. First it must be able to detect another chip within 10 feet. It must then store this detection in the memory of the chip via the microcontroller. Once connected to a PC, it will upload the list of contacts and update a central contact graph that shows which users came in contact. If the a user has been in contact with another user who has tested positive within the last two weeks, the user is be told to quarantine. Finally, the user can charge the chip via USB-C to a full charge that can last an entire day. The subsystems that make up this project are the Ultra-wideband, Microcontroller, Power, Software, and Server, as shown below.

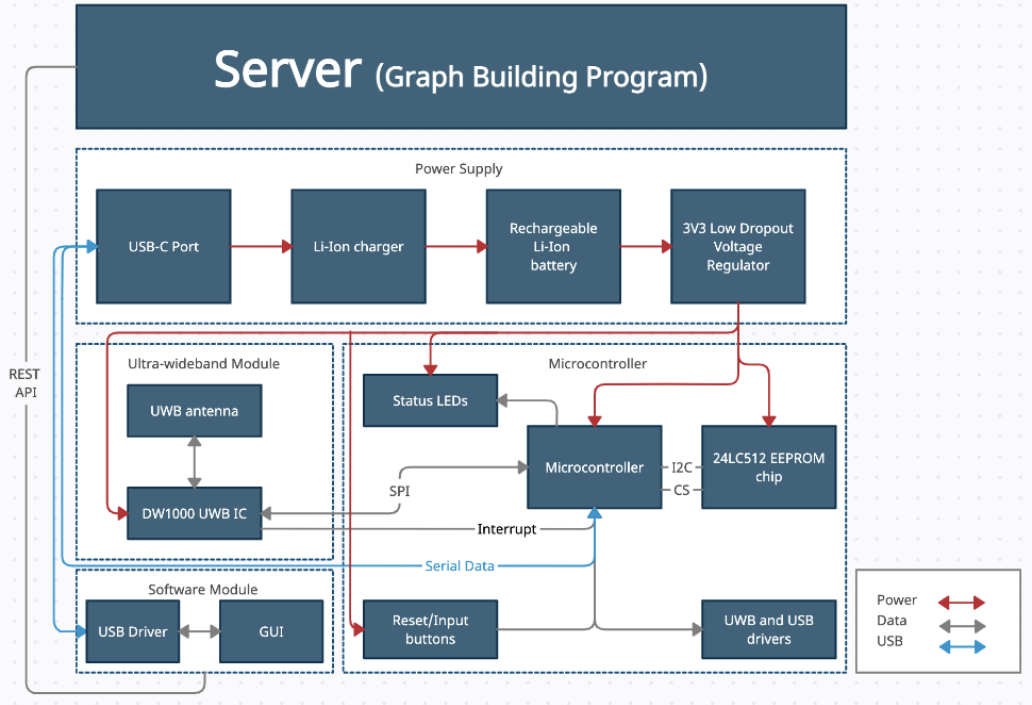


Figure 1: Block diagram of the entire system

The primary functionality of the device is in the microcontroller. The microcontroller is responsible for using the UWB subsystem to communicate with other contact tracing chips, as well as communicating with the software module to upload the list of contacts. The UWB subsystem is also crucial, since it gives the functionality to measure the time of flight between messages sent, which is how the chip determines distances between users. The software module, once contacts are loaded, sends the list of contacts to the server and asks the server whether or not the user has to quarantine, displaying the information from the server in an easy to use GUI. The server is the simplest module, simply creating a graph of contacts and checking if a user has been transitively in contact with anyone that has tested positive. Finally, of course, the power module is required to power all of the components of the system.

1.1 Performance 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 Ultra-wideband

2.1.1 Design Procedure

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. There are a variety of approaches to determine distances using radio signals. Bluetooth is a commonly used option due to the prevalence of smartphones, but suffers from accuracy problems, and while NFC can be much more accurate, it has a very low range, which doesn't make it very useful for contact tracing. UWB, as an emerging technology, solves both of these problems, as it is both highly accurate and has a high range [4]. Since we need to accurately determine whether or not people came into contact within a certain threshold, choosing this technology was vital.

2.1.2 Design Details

The UWB subsystem interfaces with the microcontroller subsystem through SPI and is powered through the stable power supply from the power subsystem. The specific chip we used 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. It has a maximum range of 290 meters, which is of course sufficient, and it also has a precision of 10 cm. These are very reasonable values to be able to distinguish contacts on the order of 10 ft.

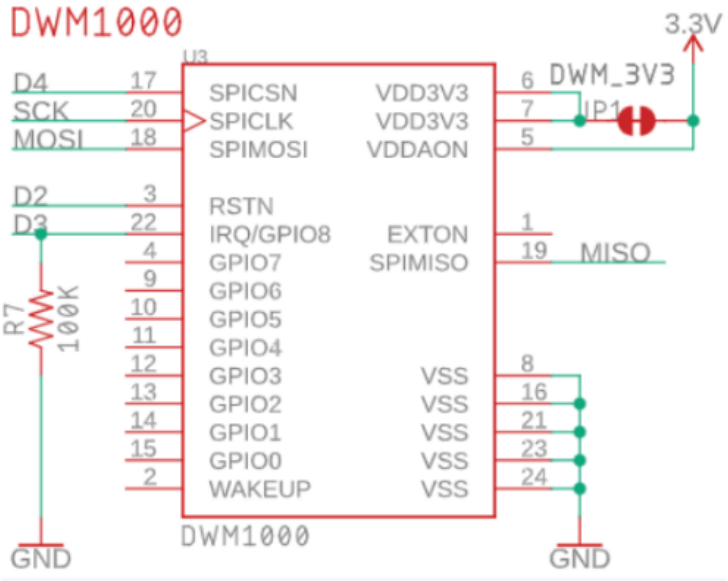


Figure 2: Schematic of the UWB subsystem

In order to be able to actually calculate distances using the ultra-wideband module, we need to devise a ranging protocol so that both devices can consistently get accurate distance values. The following diagram demonstrates the ranging protocol used:

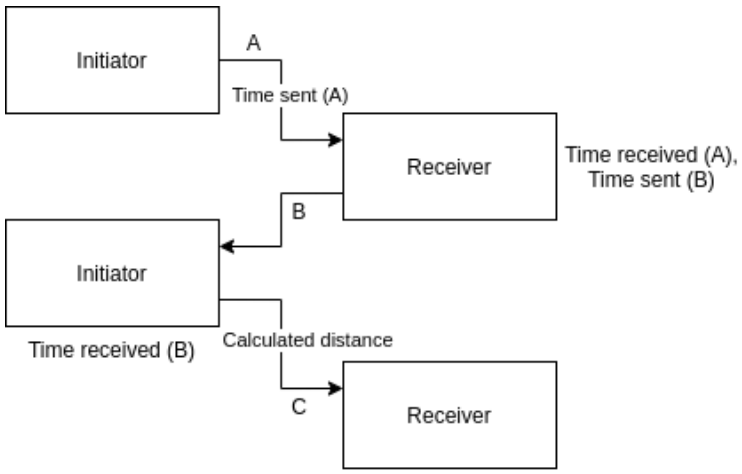


Figure 3: Ranging protocol summary

Every chips acts as an initiator, sending broadcast messages labelled A at random intervals along with its device ID. When a chip receives the initial message, it acts as a receiver, sending back the time it received message A as well as the time it will send the next message, which is a functionality provided by the UWB chip. Finally, when the initiator receives this message back, it can use the time it sent the initial message, the time it received the final message, and the times recorded by the receiver to calculate a very precise time of flight. We simply multiply the time differences by the speed of light and divide by 2, yielding a highly accurate value. After this, the initiator sends the calculated distance to the receiver, so that two chips always get the exact same distance values.

2.1.3 Verification

The following table demonstrates the verification performed to ensure that the distance values calculated by the ranging protocol using the UWB chip are accurate.

Actual (ft)	Measured (ft)	Error (cm)	Actual (ft)	Measured (ft)	Error (cm)
1	1.14	4.3	8	7.85	4.6
2	2.02	0.6	9	8.70	9.1
3	2.85	4.6	10	9.68	9.8
4	4.30	9.1	11	11.7	21
5	4.98	0.6	12	12.2	6.1
6	5.87	4.0	13	13.1	3.0

7	6.75	7.6	14	14.4	12

We do observe that the values at 11 ft and 14 ft are off by more than 10 cm, but this error is still acceptable and within the limits required by our application.

2.2 Microcontroller

2.2.1 Design Procedure

The job of the microcontroller subsystem is to initiate communication with other contact tracing chips, store contact data to be transferred to the computer, and quickly and reliably transfer this data to the computer. Initially, we planned on using an ATmega32U4 for this subsystem, but decided against it for a couple of reasons. This chip only has 2,560 bytes of SRAM, which is not sufficient for all it has to do, and since the chip only has a 16 MHz clock speed, it may not be able to keep up when there are many contact tracing chips in the area [5]. Instead, we ended up using the 32-bit ARM ATSAMD21 microcontroller, which has a higher clock frequency of 48 MHz, and more importantly 32 KB of SRAM, which is much more suitable for this application [6].

2.2.2 Design Details

Most of the work on the microcontroller subsystem, other than large amount of soldering work required to get it working on the PCB in the first place, came down to the software to communicate with other chips, store contact data and upload it to the software subsystem over USB.

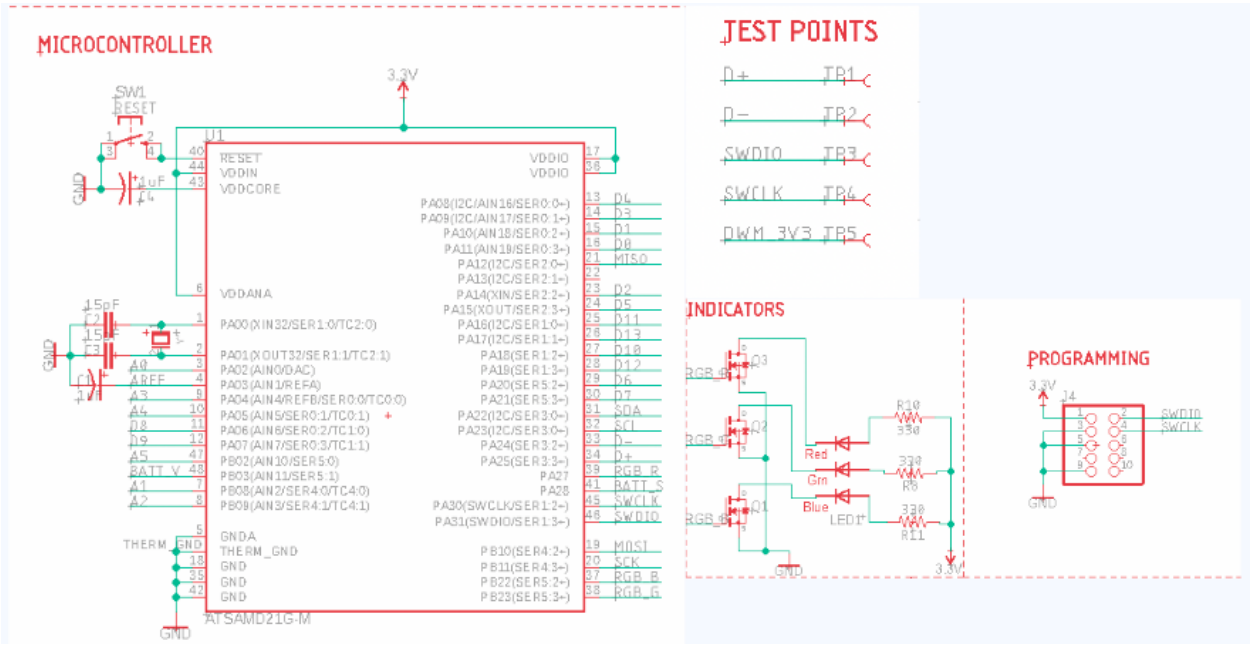


Figure 4: Schematic of the microcontroller subsystem

The exact details of the USB communication protocol will be detailed in the section for the software subsystem, since the two are so tightly coupled.

As far as ranging over UWB goes, we did some basic probability analysis to determine how many measurements would be required to match our high-level requirement that false positives and negatives not occur at a rate of higher than 25%. We found that the probability distribution of the range values at around 10 ft is approximately symmetric, which means that at exactly 10 ft, there would be a 50% chance of getting a value above or below. Therefore, to achieve the desired false positive / negative rate, we must find the value n such that $50\%^n < 25\%$, and the smallest value of n which achieves this is 3. Therefore, we require 3 consecutive measurements before deciding that a contact has either occurred or not occurred.

2.2.3 Verification

To verify that this subsystem does a proper job of deciding whether or not a contact has occurred, we simply placed the devices 8 ft apart, performed 5 measurements, and then placed the devices 12 ft apart, and performed 5 more measurements. All times, contacts were recorded at 8 ft and were not recorded at 12 ft, confirming that false negatives and false positives occur below the desired rate.

Figure 5: Checking for a contact at 8 ft

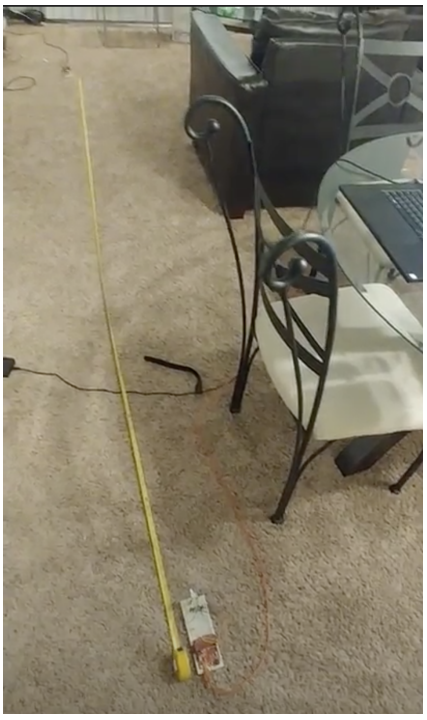


Figure 6: Checking for a contact at 12 ft

2.3 Power

2.3.1 Design Procedure

For the power subsystem there were a number of different approaches we could have taken. The strict requirement we had was to be able to supply a constant and ripple-free 3.3V to the whole subsystem. This is due to the fact that the operating voltage for all subsystems requires 3.3V. Another requirement was charging the onboard Li-Po battery. The power subsystem itself could have been broken into smaller subsystems, and that was our initial design consideration. We hoped to split the battery charging circuit and the low dropout voltage regulator circuit, but on further consideration we realized that by combining them we could put them on the same ground plane and get better thermal dissipation as well as performance.

2.3.2 Design Details

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.

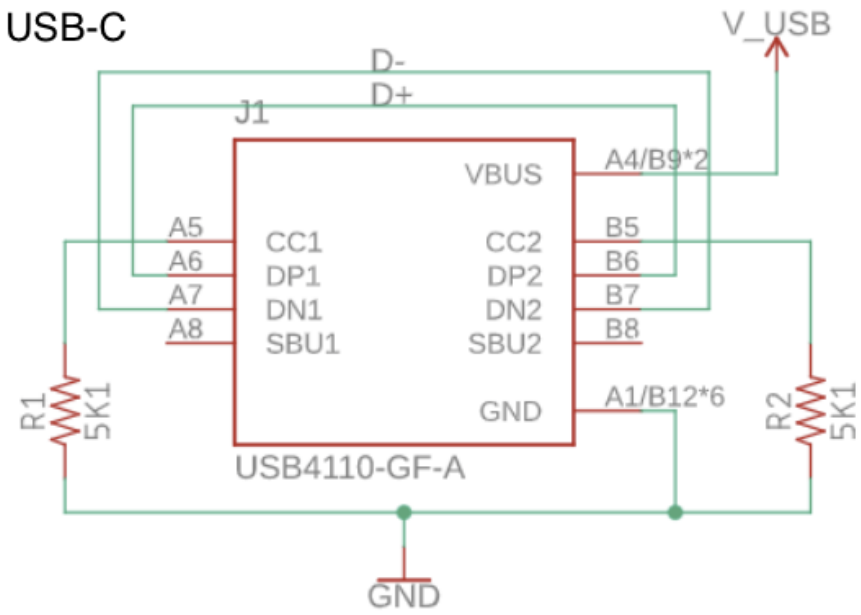


Figure 5: Schematic of USB Component in Power Subsystem

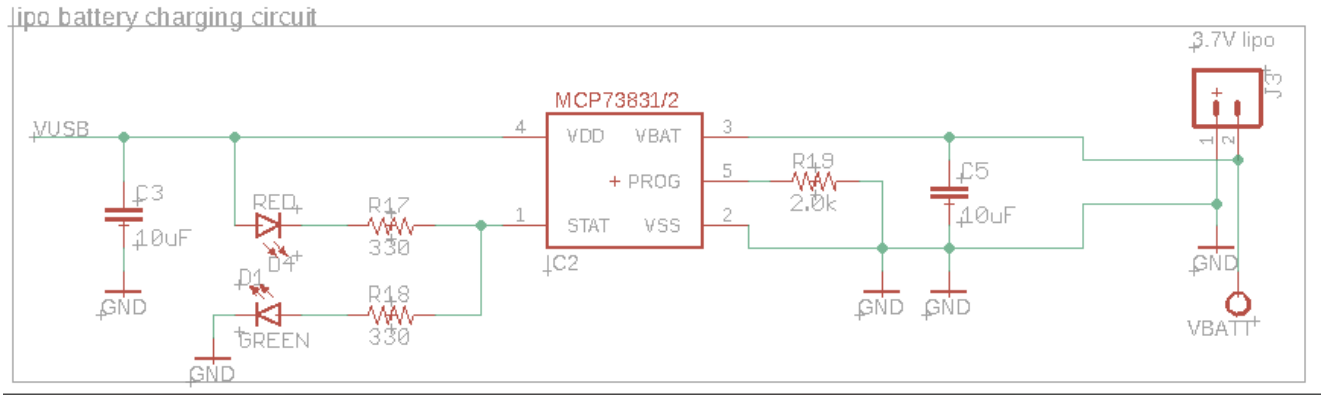


Figure 6: Schematic of Battery Charging Circuit in Power Subsystem

2.4 Software

2.4.1 Design Procedure

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 used serial over USB to have the chip interact with the PC. We are able to meet the requirement of transferring all contact data between the chip and the computer within at most 10 seconds. This is an important goal because if a shared computer is being used in a location where computers are not as abundant, many people can quickly upload their information.

2.4.2 Design Details

There are two primary pieces of this subsystem: the USB communication protocol, and the GUI application which allows users to view their contact data and whether or not they have to quarantine. First, we will describe the communication protocol. We have two types of messages: REQUESTs and NUMBLOCKs, and BLOCKs, where REQUESTs are sent by the software subsystem to the microcontroller, and NUMBLOCKs and BLOCKs are sent back. The format of REQUEST messages are as follows

0xAA
32-bit Requested 1-based block index #1 (or 0x00 for **ALL** blocks)
...
32-bit Requested block index #N
8-bit XOR checksum of the entire message
0x6162636465666768 sync string

We include a checksum so that the microcontroller will ignore any malformed messages, which are quite common when using serial over USB. We also include a sync string which the microcontroller can uniquely identify (and has a very low probability of randomly occurring in any of the other data) to know where the message ends. Then, the microcontroller responds with a NUMBLOCK message followed by the BLOCK messages corresponding to what was requested by the software subsystem. The NUMBLOCK and BLOCK messages have a similar format, sending the number of upcoming blocks, as well as the actual list of contacts corresponding to each block, respectively.

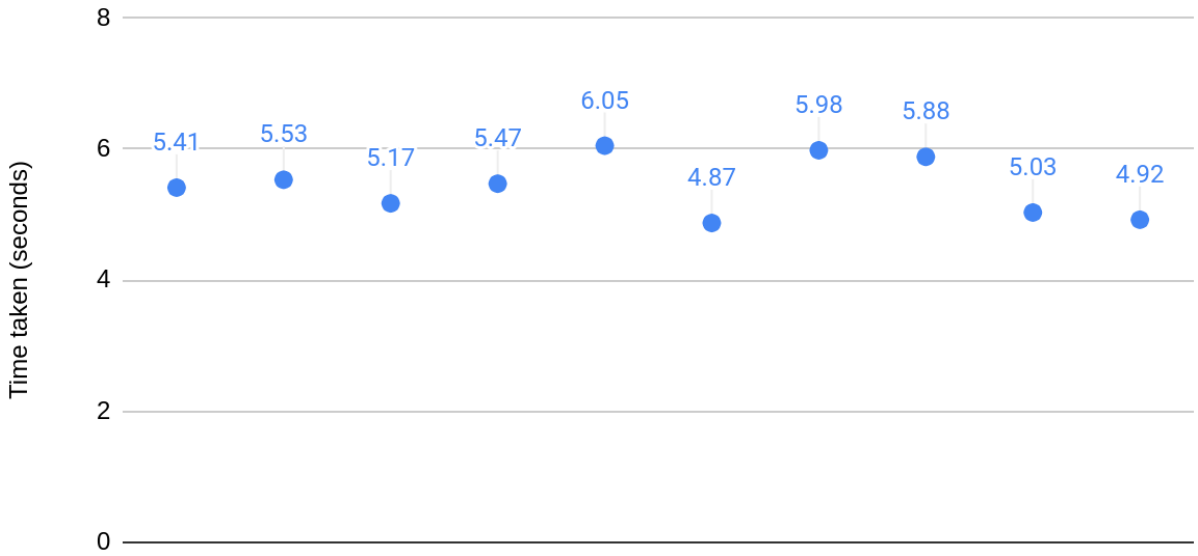
The computer attempts to do fuzzy matching of the synchronization string to increase the chance that a message isn't missed due to random errors using an edit distance based algorithm, and sends REQUESTs if there is any missing data.

2.4.3 Verification

The following is log output corresponding to requesting 16 blocks of data, where the microcontroller is programmed to insert a bit flip every 4 blocks.

```
Sending request for blocks
Got block 2
Got block 3
Got block 4
Got block 6
Got block 7
Got block 8
Got block 10
Got block 11
Got block 12
Got block 14
Got block 15
Got block 16
Re-requesting blocks 1, 5, 9, 13
Sending request for blocks 1, 5, 9, 13
Got block 5
Got block 9
Got block 13
Re-requesting blocks 1
Sending request for blocks 1
Sending request for blocks 1
Got block 1
```

As is clear, all the blocks were received despite failures. Additionally, in order to ensure that we are able to upload contacts, quickly enough, we measured the amount of time it takes to upload 5000 contacts several times, resulting in the following data.



2.5 Server

2.5.1 Design Procedure

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. A backend system was necessary in order to store all these connections as storing all this info in memory would have not been feasible. It was determined that a graph was the best possible option since it allows for the simplest and fastest way to access relevant neighbors.

2.5.2 Design Details

The code consists of two primary classes, the user nodes and the graph itself. The user nodes have an ID, status, and a set of neighbors. The graph is used to set and view connections between user nodes. Within the system users find connected nodes via a DFS algorithm. With this the user goes through connected nodes and sets quarantine status to True if a positive node is found in the same set. It interfaces with the software subsystem via a simple REST API.

3. Requirements & Verification Procedures

3.1 Ultra-wideband

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

3.2 Microcontroller

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)

3.3 Power

Requirement	Verification
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.

3.4 Software

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

3.5 Server

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.

4. Costs

4.1 Parts

Table Parts Costs

Part	Manufacturer	Retail Cost Unit ($)	Bulk Purchase Cost Unit ($)	Actual Cost ($)
ATSAMD21G	Microchip	$3.65	$3.04	$14.605
MCP73832	Microchip	$0.59	$0.493	$2.36
DWM1000	Qorvo	$17.90	$15.60	$53.70
CM7V-T1A-32.768k-12.5pF-10PPM-TA-QA	Micro Crystal	$0.50	$0.221	$2.00
15PF-0402-50V	Yageo	$0.10	$0.008	$0.80
EMK107BJ105KA-T	Taiyo Yuden	$0.10	$0.009	$1.50
MIC5219-3.3	Microchip	$0.98	$0.735	$3.92
2N7002	ON Semiconductor	$0.38	$0.073	$3.80
USB4105-GF-A	GCT	$1.57	$0.76285	$6.28
SSSS810701	Alps Alpine	$0.96	$0.671	$3.84
0603X106M100CT	Walsin	$0.13	$0.091	$1.56
GRM0335C1E471FA01D	Murata Electronics	$0.10	$0.006	$0.40
RB161MM-20TR	ROHM Semiconductor	$0.33	$0.076	$1.32
IN-PI55TATPRPGPB	Inolux	$0.49	$0.127	$1.96
CRCW06035K10FKEAC	Vishay	$0.10	$0.004	$1.00
CPF0603F100KC1	TE Connectivity	$0.15	$0.016	$2.25
CPF0603F2K0C1	TE Connectivity	$0.14	$0.017	$0.56
KTR03EZPF3000	ROHM Semiconductor	$0.13	$0.012	$1.30
EVQ-Q2S03W	Panasonic	$0.28	$0.116	$1.12
Lithium Ion Battery - 3.7v 2000mAh	Adafruit	$12.5	$11.25	$37.00
Total				**$103.97**

[7]

4.2 Labor

We 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

5. Conclusion

5.1 Successes and Challenges

We were able to hit all the thresholds for all of the requirement and verification tables. From the very first round we were able to create a fully functional pcb design. This pcb is what we used for the entirety of our project and it was able to hit all the specifications we needed. Our project was able to have two chips detect each other within a certain distance and report this detection. It would then store this into memory in the microcontroller unit. This could be stored into a fully functional server/backend when attached into a PC and properly inform a user whether or not to quarantine. Here we were also able to update testing results, albeit manually. Finally, we were able to have this chip charge via USB-C in the appropriate time measures we set out.

We had several issues to overcome in order to complete this project. To begin, we had many issues with our serial port/USB-C due to soldering issues. As a result of inadequate equipment, our parts were not functional for quite some time and took valuable time to get up and running. Our second PCB was only functional for some time and later on we had to use development boards for testing along with our first PCB. On the software side, we also had several issues. For one the wrong SPI clock speed was giving us inaccurate measurements in calculating distance and took lots of calculation to manually correct. Unaligned memory access and porting code between platforms were also issues that took time and effort to correct.

5.2 Ethical considerations

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].

5.3 Future work

There are several improvements we would make given more time and resources to work on this project. First of all we would like to Improve privacy by using stronger cryptographic standards as well as moving away from a centralized server solution. This would allow testing centers to be able to upload results and maintain confidence in keeping user data safe. It would stop malicious attacks from outside sources as well. We would also like to construct a more rigid and to scale 3D printed enclosure to protect the board from different elements. In addition we would update the server for in scale use cases. For example, we would update the date functionality so that only contacts within two weeks would cause the user to quarantine. This ensures that we are only creating relevant quarantines and not spreading to too many people. Finally, enable support for Bluetooth so that the chip can also interoperate with modern smartphones. This would slightly increase the cost of the chip, but allow for people without this chip to take part in contact tracing.

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

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

[6] Microchip.com. 2021. ATSAMD21G18 - 32-bit Microcontrollers. [online] Available at: <https://www.microchip.com/wwwproducts/en/ATsamd21g18> [Accessed 6 May 2021].

[7] Microchip technology / Atmel: Mouser. (n.d.).

[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].