MiniTrace

A Contact-Tracing System to Slow the Spread of COVID-19

EECS 473
Fall 2020

Team members: **Samina Abdullah, Luke Cohen, Matthew French, Lexi Roberts, Stephanie Sheehan**

Since the beginning of 2020, our world has looked different due to a deadly virus known as COVID-19. Over 65 million people have already been infected by this virus worldwide, with over 1.5 million of those cases resulting in death.[1] As these numbers continue to rise, action must be taken now to help slow the spread of COVID-19. Our product, called MiniTrace, attempts to decrease the rise in the number of COVID-19 cases by tracking possible exposures through contact tracing. While companies such as Google and Apple have already attempted to implement contact tracing applications, these systems have raised privacy and attainability concerns, causing many to not use their application. MiniTrace successfully fills these gaps in the market through anonymously tracking interactions in local establishments such as schools and nursing homes, where an administrator is the only person that would have permission to the data. For these reasons, MiniTrace is a worthwhile solution that is able to not only slow the spread of COVID-19, but can also do so in a way that has not been done before.

Over the duration of a few months, we worked hard to conceptualize, design, implement, and test MiniTrace to the best of our ability. This process was especially interesting because most of this work was done remotely due to COVID-19, whereas in a normal semester, this work would have been carried out in person. As a group, we decided upon a design for MiniTrace that consisted of three parts: a wearable device that could track user interactions, a base station that could transmit and categorize interactions, and a database used to store all interactions over a certain time period. At the beginning of the implementation process, we were late in completing some of our more ambitious targets, but we were able to quickly make up for lost time. Eventually, we were able to achieve all of the set goals and expectations for MiniTrace, and were able to thoroughly test the design to ensure its functionality.

The final result of MiniTrace was a fully working system that can track exposures of COVID-19 through contact tracing. This was demonstrated at EECS 473's online Design Expo.[2] For the setup, we had two wearables that were both initialized with no known COVID-19 exposures, one base station, and a database which was run by one of our team members as an administrator. For the demonstration, the administrator entered into the database that one of the two wearables was infected with COVID-19. These wearables were then put into range with one another, and afterwards, connected to the base station. As expected, one device showed that it was infected, and the other showed that it was exposed. Additionally, during the Design Expo, we were able to successfully show that the wearable devices were able to switch screens, and also that they could measure temperature through a temperature sensor. Our team was overall very satisfied with the results, and feel that we proved that MiniTrace does not only enforce privacy and attainability, but can also help to slow the spread of COVID-19.

---

[1] https://www.worldometers.info/coronavirus/
[2] https://cse.engin.umich.edu/eecs-473-advanced-embedded-systems/

Project Description

MiniTrace is a product that not only solves current real world issues, but has the potential to help control future problems as they arise. Due to a great deal of research and a well-thought out concept, MiniTrace was fairly feasible and we achieved many of our goals.

Goals

The main goal and purpose for MiniTrace was to slow the spread of COVID-19. With the current situation that we are all living in today, it is very evident that our world could use some help to control the virus until a vaccine is available. The method that MiniTrace implemented, called contact tracing, allows for people to be notified of their possible COVID-19 exposures early on so that they can quarantine themselves and not further infect others (Figure 1). While contact tracing is normally implemented using a "contact tracer" who is responsible for asking an infected person about contacts and then notifying those people, a system such as MiniTrace could automate this process and more accurately catch all possible exposures in an environment.

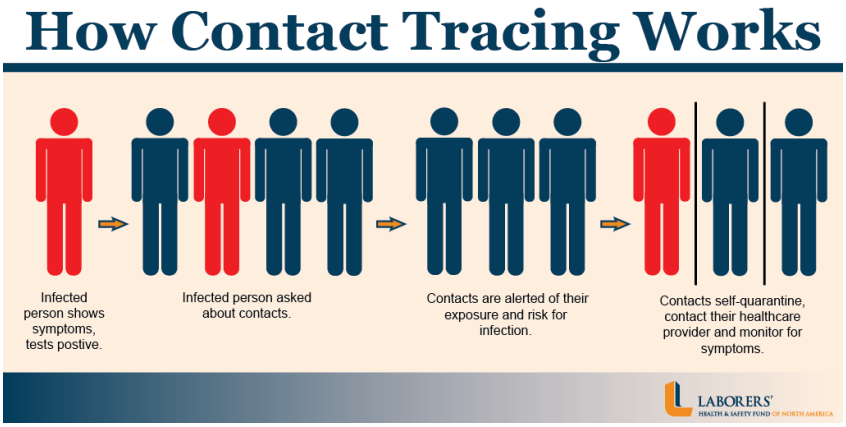

Figure 1. Diagram depicting the four main stages of contact tracing, showing those who have been infected with COVID-19 in red.[3]

Besides slowing the spread of COVID-19, MiniTrace also aimed to enforce privacy and attainability, which are current gaps in the market. While some contact tracing implementations such as Apple's and Google's API are a great start for fighting COVID-19, there are issues with these systems. For one, systems like the Apple/Google API only work on smartphones and tablets. This leaves out one in six Americans who do not own a smartphone, causing attainability issues.[4] In addition, half of all Americans are not willing to enroll in contact tracing mobile applications even if they have the necessary smartphone to take advantage of it.[5] This may be because large companies like Google are already collecting and using much data from users, causing privacy concerns for many Americans.[6] From these two statistics, we estimate that the upper bound for Americans that would enroll in the contact tracing

[3] https://www.lhsfna.org/index.cfm/lifelines/june-2020/covid-19-contact-tracing-explained-roles-for-employers- and-workers/
[4] https://en.wikipedia.org/wiki/List_of_countries_by_smartphone_penetration
[5] https://www.washingtonpost.com/technology/2020/04/29/most-americans-are-not-willing-or-able-use-an-app-tracking-coronavirus-infections-thats-problem-big-techs-plan-slow-pandemic/
[6] https://www.wired.com/story/google-tracks-you-privacy/

implementation from Apple and Google is 39.55%. We are confident that MiniTrace could cover the other 60.45% of the market missed by our competitors by further enforcing privacy and attainability.

The last set of goals that MiniTrace attempted to achieve were technical goals, including size, power, and weight. While these constraints are further discussed later on, it is important to note that these constraints shaped our concept and design. We wanted to make sure that MiniTrace could be worn comfortably for an entire day without running out of power, getting too uncomfortable or inconvenient on one's wrist, or becoming too much weight to wear and lift all day.

System Concept

The concept that we decided upon is a private, centralized approach that would work best in localized environments where the people within the environment are regularly there, and not many new people come in/out very often. Additionally, an ideal environment for MiniTrace would have an administrator or other executive figure present to handle input into the database if a new COVID-19 case is reported. Locations such as schools and nursing homes are great examples of environments that could easily and efficiently use MiniTrace.

Within this centralized approach, there are three main components: the wearable devices, the base station, and the database. Each component serves its own purpose, but they all work together to create one automated system that can implement contact tracing (Figure 2).

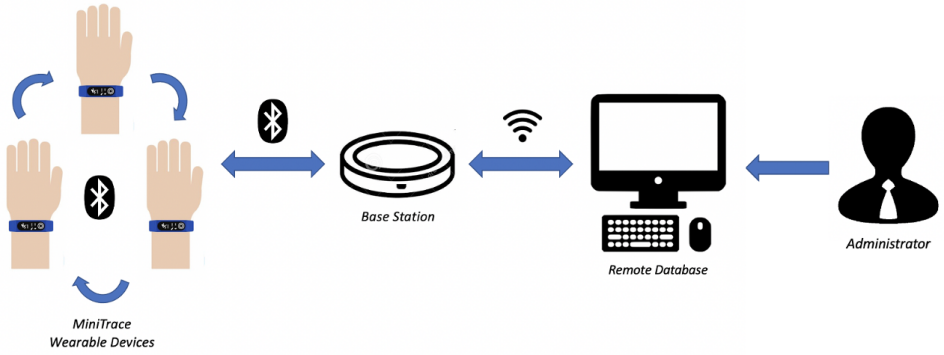

Figure 2. A diagram demonstrating how the different components of MiniTrace connect with one another. The types of communication interfaces used, including BLE and WiFi, are also shown.

Wearable Device

The wearable device is a small bracelet that ultimately tracks interactions between users. Since it is worn for an entire day, the device was designed to be small, light, and low power. The wearable device was also designed to reliably communicate with other devices in a short range, and therefore uses Bluetooth Low Energy (BLE) which is built-in to an STM32 processor on the device's PCB. Besides tracking interactions between users, the wearable device can measure a user's temperature and notify the user of a high temperature through a temperature sensor that the user would place on their forehead. This is an important feature, since one of the most prominent symptoms of COVID-19 is a

fever.[7] Additionally, the wearable device includes an OLED that can relay other information such as a MiniTrace splash screen, instructions, date and time, COVID status, and temperature reading through the push of a button. A protective, 3D-printed encloses the PCB and all of the components (Figure 3).

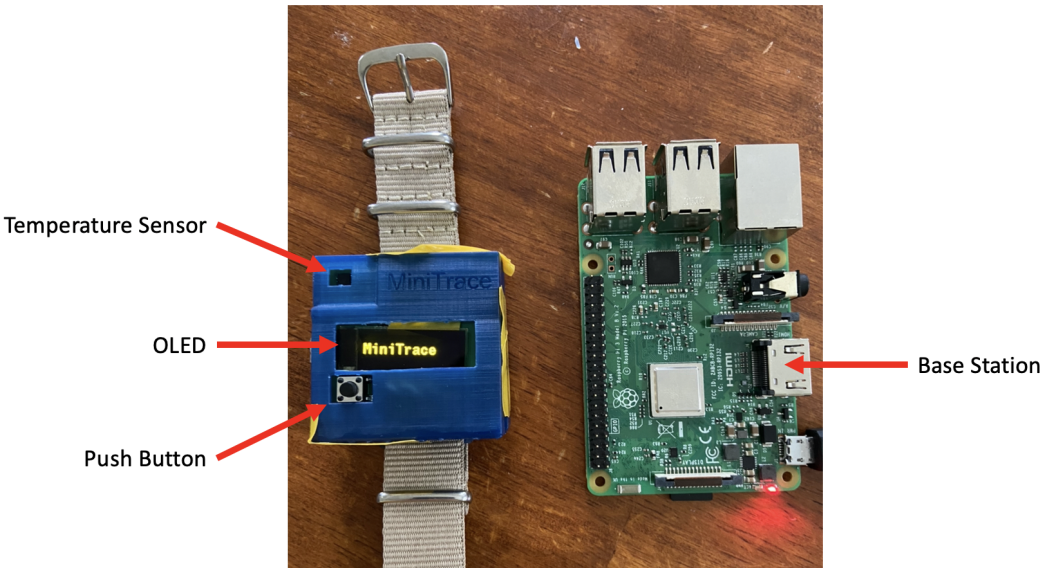

Figure 3. A picture of the wearable device (left) and the base station (right). The different components included within the wearable device are listed to the left of the wearable device for reference.

The overall idea is that a wearable device will be worn by the same person, every day, in the same environment. Each person in the environment will also follow these conditions, so that everyone in the environment is part of the MiniTrace system. Each time two users in the environment pass one another, they will exchange two encrypted tokens over BLE, notifying each other that they have interacted. Then, whenever a user passes by a base station (which are placed throughout the environment in hopes that a user will pass by at least once a day), the wearable device will dump all of its tokens to the base station over BLE, and after some analysis, the base station will update the COVID-19 status on the user's wearable device. Then, the user can push the button and traverse through the screens to see their new status. This status is either None (no known exposure of COVID-19), Exposed (a known interaction with an infected person), or Infected (user has COVID-19, which can only be generated if an administrator manually inputs this into the database).

Base Station

The base station serves as the liaison between the wearable devices and the database. It communicates with wearable devices over BLE and communicates with the database over WiFi. We chose a Raspberry Pi Model 3 B in order to have BLE and WiFi on board (Figure 3). Additionally, when the base station connects with a wearable device, it runs automated algorithms that can delete any old data from the database (interactions that happened longer than 2 weeks ago), enter the device's new interactions into the database, and update the user's COVID-19 status.

[7] https://www.cdc.gov/coronavirus/2019-ncov/symptoms-testing/symptoms.html

The core functionality of the base station is that it will be able to handle communication from the wearable, to the database, and back to the wearable. When a wearable device comes in range, the base station will connect with it over BLE and receive a number of encrypted tokens from the wearable device representing all of the interactions the user has had. The base station will then run an automated script to obtain the user's new COVID-19 status. The base station will then transmit this status back to the wearable device over BLE, where the user can see it. Since we use a database held on a cloud service, we can access the data from anywhere. Any base station we have will be able to connect to the database remotely over WiFi and they will all have access to the same data in real-time. This means that we can have any number of base stations, so we can easily scale up the size of our system for larger applications like schools and nursing homes.

Database

The database is used to store all of the interactions that have occurred in an environment as well as each user's COVID-19 status. MiniTrace uses an AWS instance of a MySQL database. While the database should ideally never be directly accessed, there is a Python script that an administrator can run that allows them to either input a new user into the database, update a current user's COVID-19 status, or delete a user from the database. This script also tells the administrator the total number of users, exposed users, and infected users in the environment.

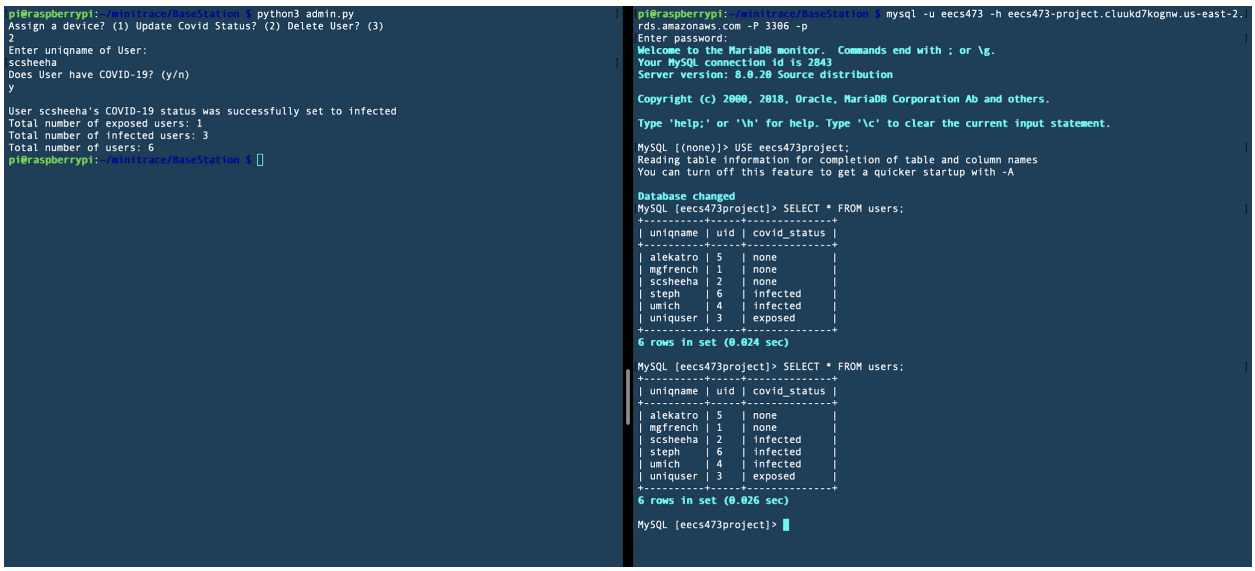

Figure 4. A screenshot of the administrator script (left) and the table of users and their COVID-19 statuses in the database (right). Note that in this example, the first table on the right shows that the user scsheeha is not infected. After the administrator runs the admin script and changes user scsheeha to infected (left), the second table on the right shows that this change has been noted in the database, as the user scsheeha now has a COVID-19 status of infected.

The overall idea of the database is that it is able to do contact tracing for us through storing interactions and COVID-19 statuses of each user in the environment. The database should only be updated either whenever a wearable device connects to a base station, or when an administrator runs the administrator script.

Feasibility

While the design and implementation of MiniTrace was challenging, it was feasible, and we were able to produce a successful product. We are especially proud that we were able to do all of this remotely, since that was no easy task. The two biggest aspects of our design that did not seem very feasible at first were the BLE communication and antenna design. Since we had three different types of BLE communication (wearable to wearable, wearable to base station, and base station to wearable) and not much experience with BLE, it was a large learning curve to understand how to write the software to do all of this. Additionally, since antennas are really tricky to work with and none of us had done antenna design before, this did not seem extremely feasible and was a difficult task. Through extensive research, seeking help from experts, and giving ourselves time to execute both the BLE communication and the antenna design, we were able to successfully implement both and prove that both tasks were feasible.

Design Constraints

MiniTrace was designed to meet many goals, with a few of these goals focusing on design constraints such as size, weight, power, and privacy. We had to take these constraints into consideration and prioritize them when implementing and designing the system. The analyses of how well we met each constraint in the end product are further described below.

Size

One of the most important constraints that we had to prioritize was size. We wanted to ensure that each device was big enough to handle all of its functionality, but not too big that it did not fit on one's wrist. Through research, we determined that a good length and width for each wearable device would be between 40-50mm and a reasonable height would be between 10-20mm. At the end of our project, our size analysis showed that each of our wearable devices had dimensions of 43mm x 44mm x 15mm. As these values all fall within the requirements we set for ourselves, we were able to successfully fulfill this constraint and create a functional wearable device that would fit comfortably on one's wrist.

Weight

Another constraint that we had to implement and build our design around was weight. We needed to make sure that the wearable devices contained all of the components needed to be functional, while ensuring that they were not too heavy for the user to wear for extensive periods of time. Through research, we determined our weight requirements for the wearable devices to be between 40g and 60g. Our weight analysis at the end of our project measured our wearable device to be 48.7g. Since this weight is well within our optimal range, we were able to accomplish our weight requirements and ensure that a user would be able to wear the device for a long period of time.

Power

Minitrace was also designed to be low-power. Because the user wearable is a watch, we wanted it to be wireless, with comparable battery life to other smartwatches. In order to accomplish this, we not only selected a 1000 mAh battery to provide lasting power to the device but also took care to limit the duty cycle of our OLED screen and Bluetooth communications. We proposed that Minitrace be able to

last at least eighteen hours on a single charge. In order to test this, we performed a power analysis on an operating wearable. The results of this test are shown below. Overall we found that after fifteen hours, Minitrace lost 4.38% of its battery capacity. From this, we expect that the Minitrace wearable is able to last for well beyond our eighteen-hour proposal, and possibly up to sixty hours if we extrapolate on our graph shown below. In fact, we may be able to use a reduced battery capacity.

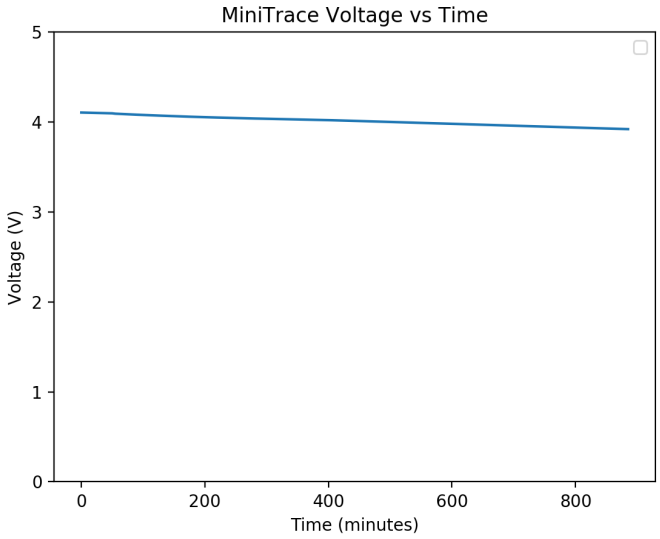

Figure 5. Plot of voltage over time of one MiniTrace wearable device while running our optimized code.

Privacy

Another constraint we prioritized was privacy. One disadvantage of existing systems is that users have concerns about the requirement or collection of personal user data. We designed Minitrace with the goal of minimizing required user data at every step. Since Minitrace is designed to be used in an environment with an authoritative body, the only user data that is entered into the Minitrace system is the COVID status of the users by the authority. This status is held only in a single table and is never spread throughout the network. In addition, the information that is transmitted by the user wearables over Bluetooth is never dependent on user information and is encrypted and randomized to prevent logging or tracking. More specifics on how this information is used is described in the communications section below. Compared to smartphones, Minitrace users can have confidence that the minimum user information is needed and used to leverage a powerful contact tracing network.

System Architecture

The Minitrace architecture involves two different devices—a wireless wearable and a Raspberry Pi 3 base station. This section will describe how these devices relate as a whole as well as the organization of both the wearable and base station.

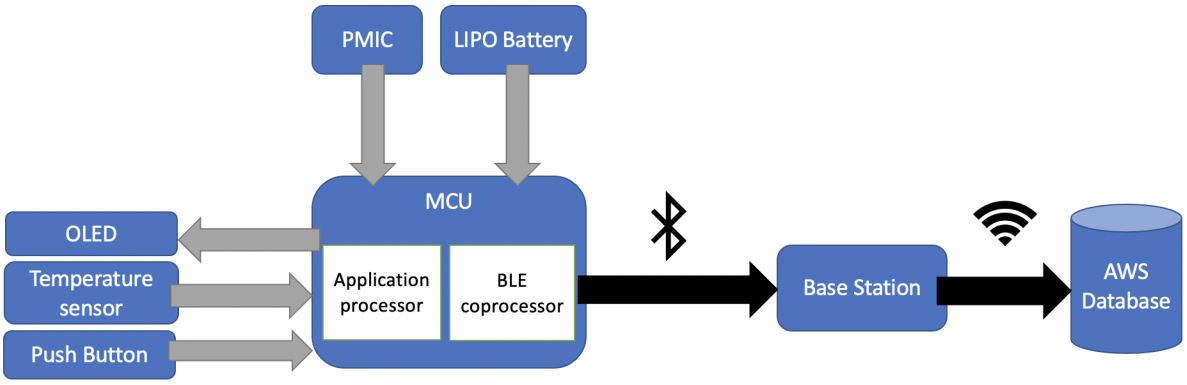

Figure 6. A block diagram representing all of the MiniTrace peripherals and interfaces.

Wearable Device & Peripherals

The wearable is a rechargeable device that supplies power through a LIPO battery and can be charged via USB. The charging operation is controlled by a PMIC and the power supplied to the MCU and its peripherals are regulated with an LDO. The wearable device has an MCU with an application processor that runs the application program and a BLE coprocessor that runs the Bluetooth LE stack. The BLE stack facilitates all device-to-device communication which is detailed in the next section. The application program controls an OLED screen on our device which displays the user's temperature, the date/time, and the user's COVID status. When the user turns on the device, they are prompted with the home screen and a navigation screen that describes these features. The user can switch between screens using the push button on the device. When entering the temperature screen, the user is given instructions for how to take their temperature which triggers a reading from the device's temperature sensor.

BLE Communication & Database Operation

To implement contact tracing, the wearable of each user transmits and stores information that can be used to determine which users have potentially been exposed to a virus. This information is sent among wearables over Bluetooth Low Energy (BLE) in the form of a 16-byte beacon broadcast we call tokens. About every ten minutes, each wearable device generates a unique token based on its serial number and the time of generation. This token is then encrypted with RSA-256 to ensure no device information is available over the air and then stored to be broadcasted later. The wearable broadcasts its most recently generated token in a non-connectable undirected advertisement with an unresolvable address every 250 milliseconds. Wearables are also scanning for tokens and servers for one second every five minutes. When one wearable device comes within the broadcast range of another wearable device, the token is received and analyzed. Based on the RSSI of the received data, only advertised tokens that are received from within around ten feet are stored in the wearable's memory. A wearable can store many tokens from multiple other devices at once.

Wearables also scan for low duty cycle connectable directed advertisements that are broadcasted from all base stations. Every base station hosts a GATT server which consists of a custom service and several data characteristics. When in broadcast range, the wearable can determine that the received advertisement is that of a base station from its advertised service UUID. The wearable then creates a point-to-point connection with the associated Bluetooth MAC address. While connected to a base station the wearable device will stop exchanging tokens with other wearables, and instead perform a data exchange with the GATT server.

The GATT server, shown in the figure below, running on the base station contains four characteristics with which the wearable interacts. When the wearable first connects with this server, it writes all of its saved tokens, along with its device ID, to the token characteristic. This includes all of its most recently generated and broadcasted tokens, as well as those it has received from other wearables. Based on the device ID, the tokens are stored in a dictionary until the transfer is complete. After all tokens have been written, all but the most recently generated tokens are cleared from the wearable's memory so that more can be received. To indicate that it has completed transferring tokens, the wearable will write its device ID to the server's flag characteristic. When this happens, the server will store the saved tokens in an AWS SQL database.

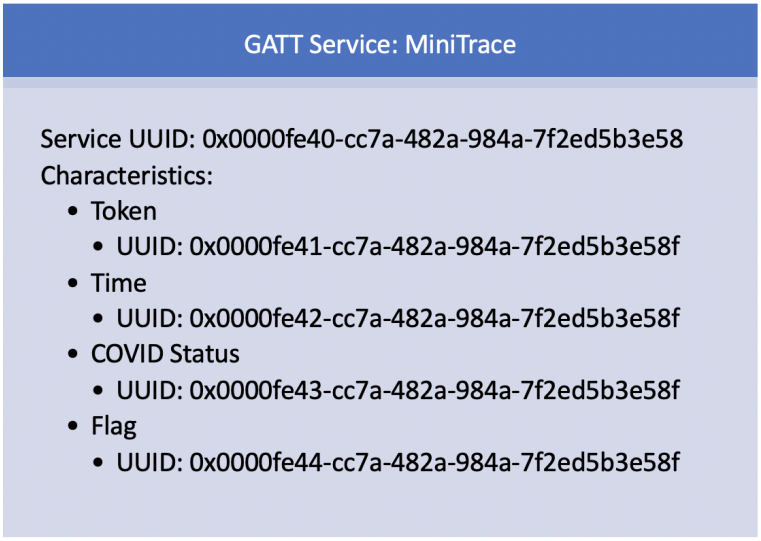

Figure 7. MiniTrace GATT Service

The AWS database is a SQL database containing three tables—one containing the COVID status of each of Minitrace's users, one storing which tokens were generated and sent from each device ID, and one storing which tokens were received by each device ID. An administrator can add users and their device IDs to this database and update their COVID status through an application program run on the base station.

After storing tokens in the received and own token tables in the database, tokens that were received more than two weeks from the present time are discarded, and a check is performed to determine if the connected user's device has been exposed to COVID. The received tokens for a given

device ID are cross-referenced with the sent tokens from other devices. If it is found that a token sent from a user with an “infected” status was received by the connected device, the connected user’s status is made “exposed”. The connected user’s COVID status is represented as a character “n”, “e”, or “i” and written to the COVID status characteristic. At this time, the server also updates the time characteristic with the current time and day. After all of these tasks are completed, the server will update the flag for the connected device ID to indicate that the data is ready to be read.

Meanwhile, after the wearable has written to the flag characteristic, it will repeatedly poll the flag characteristic until the server indicates that the time and COVID status are valid. When the server updates the flag, the wearable reads the time and COVID status characteristics and stores the respective information to be displayed.

Milestones, Schedule, and Budget

The original budget we proposed for our project was $534.30. Our total cost in the end without shipping was $613.80 for all of the components needed. We went over budget by the second milestone due to buying wristbands for our actual finished product, extra PCBs in our first PCB order, and extra parts for the PCB. We had to purchase a second order of PCBs that was $41.50 and some miscellaneous parts so in total we spent about $841.08 with shipping included. We went over budget because we didn’t allocate enough money to the PCB order and the fact that we might have had two orders. We also didn’t account for some of the extra parts we’d need for the PCB, such as the crystal oscillators. We purchased a lot of extra components as well since a lot of the components were very small and difficult to solder. We found out through research later that the Raspberry Pi’s that came in the lab kits did have Bluetooth capabilities, so we didn’t need the Raspberry Pi 4’s that we purchased. As for the components that actually went onto the wearable device, most of the budget went to shipping, two PCB orders, and components that aren’t needed for the end product. Therefore, the devices themselves would be fairly cheap to mass produce, which was our main goal when planning out our budget.

Our original timeline for the project was to have everything finished by Thanksgiving break and just have to work on the final deliverables and present afterwards. We ended up putting the finishing touches on our project the night before the Design Expo, so we ran over our original timeline by about a week and a half. In the beginning we were able to get all of our parts ordered and start prototyping on schedule. We didn’t receive some of our parts as early as we would’ve hoped so we were unable to meet some of our targets for Milestone 1. We also mainly focused on the mechanical design, Bluetooth communication, and sensor interface from Milestone 1 onward, so we got behind schedule for PCB design. We did not start our PCB design by October 8th, but we were able to meet our targets for the PCB and our manufactured casing for Milestone 2 so we caught back up to the schedule in that regard. Bluetooth communication took from day 1 of the project to right before Thanksgiving break, so we seriously underestimated how long that would take to get fully functional. Once we had our first PCB ordered we were somewhat back on schedule because all of our other software was tested and debugged on the prototype and we had the database setup with the code for it as well. So by the end we were mainly behind schedule due to underestimating the Bluetooth code, getting our PCB triple checked

since we had to include an antenna, and then having to order a second round of PCB's. We had to put in a new PCB order, test them, and assemble them all over Thanksgiving break, so while that was being worked on, we integrated all of our code and tested it on the prototype over the same week. Things generally took longer than we had expected, but we were able to get our system working well in the end.

Milestone 1	Milestone 2
Demo 1: Wearable prototype able to send tokens to each other and print on screen	**Demo 1:** PCB is assembled
Demo 2: Temperature sensor can read user's temperature correctly and display measurement on screen	**Demo 2:** Casing for wearable is manufactured
Target 1: Wearable prototype can send data base station prototype	**Target 1:** Wearable integrated with PCB can send and receive data to/from other wearables
Target 2: PCB design is finalized	**Target 2:** Wearable integrated with PCB can communicate with base station
	Target 3: Hardware interfaces are properly implemented, tested, and debugged

Table 1: Our original milestone demos and targets.

As seen in Table 1, Milestone 1 consisted of two demos: show that the wearable prototypes are able to communicate over Bluetooth, and show that the temperature sensor and OLED display are working. Our targets for the first milestone were to have the wearable prototype be able to send data to the base station, and have the PCB design finalized. Our goals for Milestone 1 were very ambitious because we didn't finish the Bluetooth communication protocol fully until the week before Thanksgiving break. Thus, Demo 1 and Target 1 for Milestone 1 were not hit as well as Target 1 and 2 for Milestone 2. Although we did not finish the Bluetooth communications between the wearable devices and the base station until November, we were able to see a development board broadcasting data over Bluetooth as well as the base station via a scanner application on our phones. Underestimating the timeline for the Bluetooth communication is one major reason why we seemed to be behind in our progress reports. For Milestone 1 we were able to successfully complete Demo 2. We did not hit Target 2 because we were still gathering all of our resources for the PCB design and working on the mechanical design of the casing. The mechanical design was particularly important for our wearable because our temperature sensor was a really small surface mount sensor, so we had to carefully think about how we would be able to have that exposed outside of the casing as well as the push button and OLED display.

Milestone 2 consisted of two demos: showing our assembled PCB as well as the manufactured casing for the wearable. Our targets for Milestone 2 were to have the integrated Bluetooth and hardware interface code working on the PCB, and have the integrated code fully implemented and tested. We completed Demo 1 and Demo 2 for this Milestone, however, the PCB we assembled ended up having a

problem that we couldn't solve. We had to order another round of PCBs after finding that problem, we needed a pin to be pulled low and couldn't fix it without changing the PCB. Even though we had to redesign and reorder our PCB, we caught up a bit in our timeline by Milestone 2 because we had the main parts of our PCB design down and caught any errors we had to fix. In between Milestone 1 and Milestone 2 it took a long time to get the PCB design finalized because we had to check the antenna design with multiple people and make sure that we were doing that correctly. That was an extremely important part of the PCB because without it we would not be able to use Bluetooth on our PCB. Instead of getting the Bluetooth and hardware interface code fully integrated and tested, we were working on the database during this period. We created the instance of the MySQL database on Amazon AWS, wrote all of the python scripts to access it, and tested these scripts so that we were ready to test the entire system once we got Bluetooth working. We also had the OLED, temperature sensor, and push button code fully implemented, tested and debugged so we hit Target 3. We were able to display the real-time clock and date, put the temperature sensor into shutdown mode to save power, and write the algorithm for turning off the OLED display to save power as well. At this point we were making good progress, so we took a week or two longer to finish the project then we had originally planned due to double checking everything on our PCB and testing it fully, and because we underestimated how long the Bluetooth communication would take.

Lessons Learned

Software Integration

Throughout the software integration process, we dealt with issues due to different priorities between our timer interrupts, GPIO interrupts, and scheduled tasks. Ensuring that our timing was correct for the timer and how often we were doing the scheduled tasks was key in reducing the interference between the Bluetooth code and the hardware interface.

BLE

BLE communication was undoubtedly complicated to understand and implement. As for technical material, we learned basics of wireless communication, different types of communication possible with Bluetooth, how to communicate between devices with advertisements and point to point connections, BLE ACI commands that facilitate low level communication, and BLE event flow handling. We did not expect BLE to be as complicated as it was, so the beginning of our project was held back significantly due to our learning curve. If we were to do this project again, we would consider using Zigbee instead of BLE as it is easier to use and we would likely choose a different board. Different boards, like from nRF for example, seemed to have more resources for development of wireless communication systems. As another consideration, we would look for boards for which Bluetooth/Zigbee libraries have been written, so that we could write code at a higher level without having to get into the depths of BLE implementation which was not necessary to understand to implement our project.

PCB Design

The PCB design and manufacturing process was proven to be particularly difficult. Due to the expense and moderately long lead time, the board had to be as correct as possible on the first iteration. Unfortunately, there were some design flaws that led us to have to revamp our design and order a second round of boards. Throughout this PCB design process, we learned that attention to detail is paramount and that even overlooking something as small as pulling an innocuous “BOOT0” pin low can lead to an unusable board. Because of the architecture of our dual processor chip and our need for wireless communication, we were required to incorporate an onboard antenna into our PCB design. While the bulk of the design was generally outlined for us in various specifications and datasheets, we ended up with a moderate amount of knowledge about antenna theory as it relates to layout considerations on the PCB as well as criteria that allowed us to maximize the chance that it worked. Overall, the PCB design yielded a bounty of lessons throughout the process that will be helpful for projects or work in the future.

Casing

The casing design was relatively simple once the PCB had been designed well as a 3D model of the board could be imported from Eagle into Fusion360, but the manufacturing of the case led to a few challenges. We found that 3D printing was quite imperfect and that we needed to run through multiple iterations in order to make sure the tolerances between our parts were correct and that our electrical parts could fit inside of the body of the design. After these iterations, our case successfully held all of our components snugly and snapped together well, but did not expose one of our sensors well enough to interface with it. Because of this, future work on this project could be to either improve the casing to accommodate the size of the sensor better, or find a sensor that is better exposed and design an updated casing module around that.

Cost of Manufacture

An initial assessment of our manufacturing cost (per unit board) from Circuit Hub[8] showed that when manufactured in mass quantity (>10000 units), each would cost $21.52 to manufacture, assemble and flash with firmware. An additional cost of external components including our OLED screen, bulk priced at $3.67 for 1000 units[9] and battery, bulk priced at $3.30[10], would increase our per unit cost to $28.49. While a quote for the mechanical casing is more difficult to find, it can be assumed that it would add a negligible amount of cost per unit to our design.

Addressing the application of this project as a product for the real world, we originally wanted to create a system that could be cheap and simple enough to be viable for use in environments like schools and nursing homes. Given that a typical manufacturer to retail markup for electronics is between 150 - 400%[11], this would make each wearable device cost between $40 and $120 and each base station, being $35 for us to obtain, cost between $52 and $140.[12] Given that our system can be implemented in many environments, our market is large. Any person who goes to an institution regularly would find our device useful for tracking COVID exposure. These institutions include workplaces, educational facilities, healthcare facilities, prisons, etc. Most people living in the developed world are part of one or more of these institutions. Using this rationale, our market size could go up to 1.3 billion users, the population of the developed world.[13]

[8] https://circuithub.com/

[9] https://www.crystalfontz.com/product/cfal4864a071bw-48x64-71inch-oled-display?kw=&origin=pla&gclid=CjwKCAjwh7H7BRBBEiwAPXjadhDTgtAT7XGhniOb1UDpoGLHENYcRyOM_09cywDLEjmsiPBBM9gcfhoCKu4QAvD_BwE

[10] https://www.alibaba.com/product-detail/Low-price-1000mah-lithium-polymer-3_62553698407.html?spm=a2700.7724857.0.0.632414d1nc7Fl2&s=p&fullFirstScreen=true

[11] https://ceklog.kindel.com/2012/08/30/retail-pricing-markup-and-margins/

[12] https://www.adafruit.com/product/3055?src=raspberrypi

[13] https://www.consultancy.uk/news/2191/97-percent-of-population-growth-to-be-in-developing-world

Parts

Category	Component	Cost Per Part	Quantity	Vendor	Total
Dev Kits	STM32WB Dev Kit	$42.88	4	STM	$171.52
	MAX30205 Dev Kit	$9.14	2	ProtoCentral	$18.28
	OLED Breakout	$6.80	1	CrystalFontz	$6.80
	LIPO Charger	$6.95	2	Adafruit	$13.90
	RaspberryPi 4	$30.00	2	Adafruit	$60.00
PCB Components	STM32WB	$9.43	7	Mouser	$66.01
	PCB Connectors	N/A	N/A	Multiple	$9.53
	PCB Power	N/A	N/A	Multiple	$21.255
	PCB Passives	N/A	N/A	Multiple	$73.67
	PCB RF	N/A	N/A	Multiple	$17.025
	LIPO Battery	$8.19	4	Amazon	$32.76
	OLED	$5.16	4	CrystalFontz	$20.64
	MAX30205	$2.44	5	Mouser	$12.20
	32.768KHz Osc	$0.73	15	DigiKey	$10.97
	32.00MHz Osc	$0.48	15	DigiKey	$7.25
PCB	PCB Iter1	$2.60	10	ALLPCB	$26.00
	PCB Stencil	$20.00	1	ALLPCB	$20.00
	PCB Iter2	$2.60	10	ALLPCB	$26.00
Total					$613.80

Table 2: MiniTrace Parts Cost

PCB

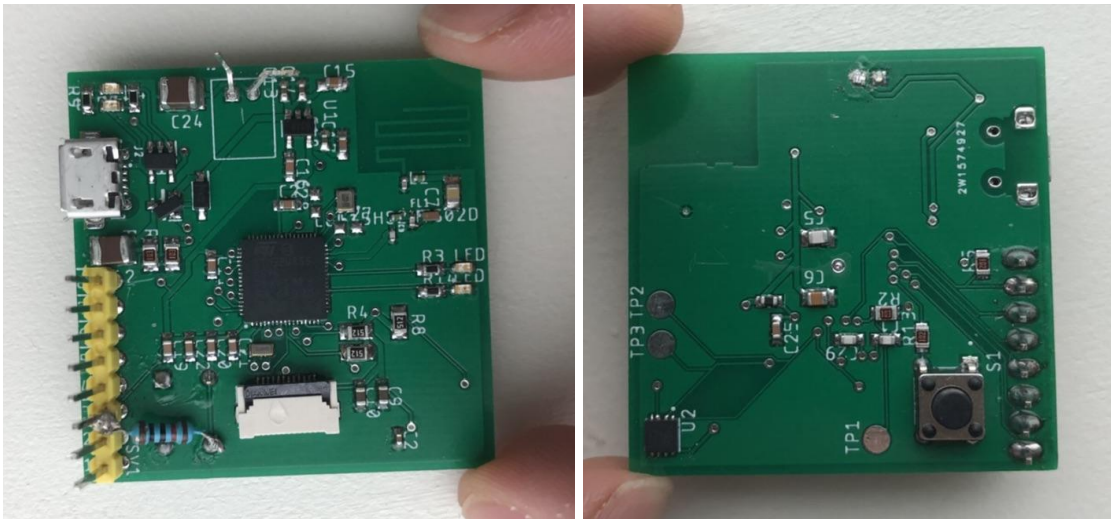

Figure 8. The back of the MiniTrace PCB (left) and the front (right).

References and Citations

References

Base for OLED Code: https://github.com/afiskon/stm32-ssd1306/tree/master/ssd1306

Our wearable project is based on the example projects, BLE_p2pClient and BLE_Beacon, from the STM32WB GitHub repository at https://github.com/STMicroelectronics/STM32CubeWB. We significantly modified the BLE_p2pClient project and incorporated the Eddystone beacon functionality we needed from the BLE_Beacon project.

Our base station GATT server was taken from the project at https://github.com/PunchThrough/espresso-ble. This project provided a python interface and examples for controlling the Bluetooth device on our RaspberryPi and a sample GATT server. We modified it by adding custom characteristics.

Our RSA implementation which is used to hash generated tokens was used from the project at https://github.com/B-Con/crypto-algorithms.

PCB design references:
https://www.st.com/resource/en/user_manual/dm00517423-bluetooth-low-energy-and-802154-nucleo-pack-based-on-stm32wb-series-microcontrollers-stmicroelectronics.pdf

https://www.st.com/content/ccc/resource/technical/layouts_and_diagrams/schematic_pack/group0/e3/a0/c8/d8/8d/da/4f/43/mb1355_schematics/files/MB1355-WB55RGV-C02_schematic.pdf/jcr:content/translations/en.MB1355-WB55RGV-C02_schematic.pdf

Antenna design reference:
https://www.st.com/resource/en/application_note/dm00470410-low-cost-pcb-antenna-for-24ghz-radio-meander-design-for-stm32wb-series-stmicroelectronics.pdf

Thank you Steve the PCB guy for helping us check the design of our PCB, Professor Alanson Sample for helping us with antenna placement on our PCB, and Fiona from ALLPCB for answering our emails at 4am.