Scotty Maps

Jeff Chen, Ifeanyi Ene, Weelie Guo

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—Scotty Maps will be a system capable of providing highly accurate indoor localization and navigation services to students at Carnegie Mellon University. This project will utilize a network of ultrawideband transceivers to localize the user within academic buildings on campus. Our web application will display the user's location, as well as provide navigational instructions to the best paths to the student's destination. We hope to create a novel, inexpensive, and scalable system that can be easily implemented in preexisting buildings on campus, streamlining the experience of new students exploring buildings on campus.**

***Index Terms*—Indoor Localization, Mapping, Navigation Systems, Time of Arrival, Ultrawideband**

I. Introduction

Navigation systems have become a ubiquitous part of our lives, revolutionizing how we move the world. GPS can be leveraged to provide directions in nearly any outdoor environment, allowing navigation systems to offer unparalleled convenience and precision. However, navigational systems do not work well indoors due to GPS signals being stopped by walls. Hence, an indoor navigation system would take our daily navigational needs even further, providing seamless guidance within complicated buildings.

For students, the benefit of indoor navigational systems is especially pronounced. New students at CMU may have particularly significant difficulties in finding rooms in the academic buildings on campus. Trying to find a room in an unfamiliar building can both stress students and waste their time. An indoor navigation system would assist students in confidently finding their destinations, minimizing the frustrations that come along with getting lost in a new place.

Scotty Maps aims to solve the problem of students becoming lost in CMU buildings by creating a localization system and navigation app to guide students directly to their desired rooms. All students will need is a tag they can put in their backpacks. Then, once they are in a building, they can use their phone to access our web application, which will provide them with directions to get to their destination.

Indoor navigation systems are not wholly a new idea. Several companies have solutions regarding implementing indoor localization systems. However, these systems are often enterprise-level and are designed for large-scale commercial applications, making them prohibitively expensive for a college campus. Instead, Scotty Maps will make a solution tailored specifically to the unique needs of the CMU campus by creating an affordable and scalable solution, focusing primarily on navigation within academic buildings to streamline the student experience.

II. Use-Case Requirements

Scotty Maps will be designed primarily to achieve excellent localization accuracy, along with providing highly useful directions for the user to follow. However, along with these goals, this project will also have an intent of being accessible, and unobtrusive, while also protecting the privacy rights of students who will be using the device.

Our project has several use case requirements, the most important being highly accurate localization of up to an accuracy of one meter. One meter is typically the width of most doorways, and we believe that this localization resolution is sufficient for preventing students from getting lost in buildings. Then, we would like the localization system to function responsively, having an update frequency for the student's location of at least 2 Hz. Assuming the student is moving at a walking pace, this frequency is sufficient to accurately maintain the real-time location of the student, as students will move less than a meter in the 500 ms.

There are also several physical characteristics of what the final product should consist of. We would like the portable tag the user is carrying to have a battery life of at least 4 hours, as we think 4 hours is approximately the maximum amount of time we expect a student might spend walking around indoors on a given day. Next, the size of the tag should not be excessively large either, as we would like the tag to fit in a user's backpack. Hence, the device size should be less than 2 liters, the size of a laptop. Additionally, the overall price of the tag should not be too expensive for students to purchase, at a cost of less than $100. Finally, the cost of installing the localization system inside a building should be reasonable, at around $50 a hallway, or approximately $300 for a floor of a typical building.

Due to the system constantly tracking the students as they are moving around indoors, respecting the privacy rights of the students becomes paramount. Security measures such as a login system will be implemented such that bad actors will be unable to access the system and view the locations of other students. Furthermore, our system should avoid storing historical data of student movements, as it should only use relevant information on the current location of the student to provide navigation information. Finally, we are also going to be using CMU building floorplans in the navigational view of our web application. This information typically requires an

AndrewID to access, so we think it would be prudent for our web application to only support usage from people with AndrewIDs.

III. Architecture and/or Principle of Operation

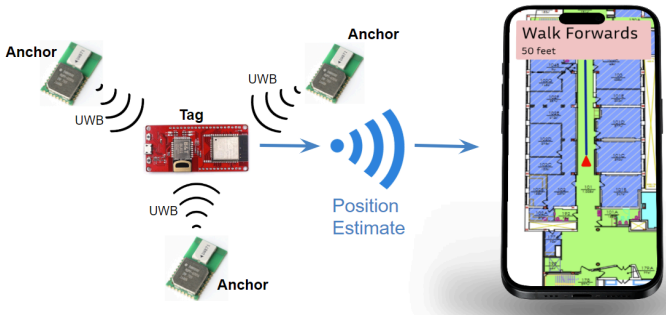


Fig. 1. A high-level view of our system implementation for finding the user's location and providing them with directions.

As seen in Figure 1, the high-level overview of our system consists of a network of UWB transceivers, which function as "anchors". These anchors will be fixed in various locations in a building. The student carries around a "tag". Once the tag is in the vicinity of the anchors, the tag will communicate with the anchors to figure out the distances between it and each of the anchors. With the distances, it will then run localization algorithms to determine the device's location. This information is communicated over Wi-Fi to our server, which will host our web application. Users will utilize a browser on their phones to access the server and view the webapp, which will display their current location in the building, as well as the directions they need to follow to get to their destination.

Our system block diagram is shown in Figure 2. An anchor is displayed at the very leftmost part of the image. In our final design, we aim to have at least 10 anchors fixed in various positions around a building for better connectivity throughout the area of the building. These anchors will be constantly listening to UWB packets, sent by the tag. Upon receiving these packets, the anchors will respond to initiate a communication protocol with UWB to determine the distance between it and the tag.

The tag will have the functionality to communicate with all of the anchors present in our network. It runs the localization algorithms necessary for localizing our device. Additionally, there is also an IMU attached to the tag, which will allow us to find the user's orientation, so we can know which direction they are currently facing within the building. The IMU can also be leveraged to help provide an estimation of the direction the user is headed in, which can be used in tandem with our UWB localization to better refine our location. The tag can send all the necessary localization information over Wi-Fi to our server.

Our server consists of a Django server that will be hosted on AWS. The server contains a Django web application, which will be the primary form of contact between the user and the localization device. The user will be able to input locations they wish to go to via the webapp, and then the webapp will use that information to find navigation instructions the user can use to get to their destination. While in navigation, the user will be able to input feedback into the system by their current position (provided to the server by the tag). Our server will be able to keep track of the location of the user and display that information for the user to see in the web application.

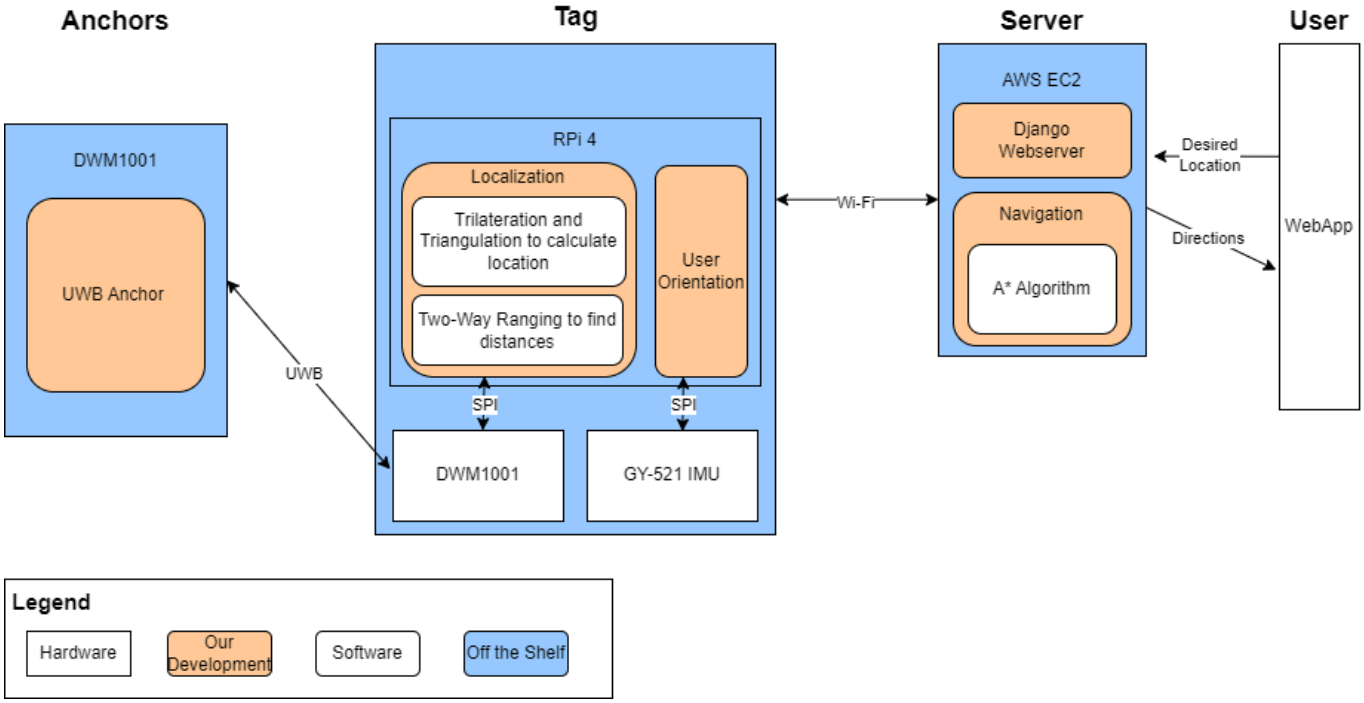


Fig. 2. System block diagram showing the overall layout of our anchors, tag, server, and user interface.

IV. Design Requirements

A. Ultrawideband Accuracy

The most important use case requirement we would like the design to focus on is localization accuracy. To achieve a high accuracy of less than one meter, we would like to use UWB and the TWR TOA protocol to determine the distances between our UWB devices. This protocol relies upon a tag sending a message to an anchor before the anchor responds with a reply. The DW1000 chip on the UWB board is then able to calculate the time of flight (TOF) using (1), where t_1 is the start time of the initiator, and t_2 is the time at which the initiator receives the response from the responder.

$$TOF = \frac{t_2 - t_1 - t_{reply}}{2} \quad (1)$$

The propagation speed of UWB signals through air is equal to the speed of light. Hence, we can solve for the resolution the DW1000 should have to be able to measure differences in location by using (2).

$$Distance = c\frac{(t_2 - t_1 - t_{reply})}{2} \quad (2)$$

We can assume that t_{reply} is a fixed constant based on the delay of the sum of the anchor's antenna and the anchor's processing speed. While solving for the propagation time of the signal, this can be safely ignored. By using a distance of 1 meter and a speed of light of 3.0*10^8, we find that the DW1000 should have a time resolution of at least 6.67 nanoseconds to be able to differentiate distances of one meter. Unsurprisingly for a device with accuracy claims of 0.1 meters, this is the case, as the DW1000 has a time resolution of approximately 15.65 picoseconds [1]. Our findings reveal that our UWB boards are capable of reaching the accuracy necessary for our application.

B. Responsive Tracking

We would like to be able to sample the user's location at a frequency of at least 2 Hz to be able to capture their location in real-time. To do this, our tag needs to obtain distances from the anchors as well as run the localization algorithms within 500 ms. It is likely the communication protocols could be the most time-consuming portion. First, we need to be able to communicate with multiple UWB devices simultaneously. To obtain accurate distances between the tags and the anchors, several samples of distances should be obtained between the same tag and anchor. The TWR TOA protocol takes approximately 1.2 ms between a tag and anchor when the tag is able to discover an anchor [1]. When considering a larger network of at least 10 anchors and assuming the tag will sample each anchor at least 5 times, we note that the overall localization time is reasonably quick at less than 100 ms and should not bottleneck our overall localization speed. The rest of the computation relies on our localization algorithms. These will be run on an RPi 4, which should be sufficiently powerful to run the algorithms well within the required time. Altogether, these results show we should be able to meet the timing requirements for our project.

C. Power Requirements

Our goal is for our tag to have a battery life of at least 4 hours, as we assume that this is approximately the maximum time a student will spend walking indoors in a given day. Assuming the power consumption of the DWM-1001 development board is approximately 82 mA at 3.5 V when it is sending data, and noting that the power consumption of the RPi is around 600 mA at 5.0 V, we can use (3) to estimate our average power consumption when constantly transmitting and processing data is around 2.8 watts.

$$p = iv \quad (3)$$

From these results, we find that our battery supply for the tag should be at least 12 watt-hours. Any power supply with approximately 2.5 amps of capacity at 5 volts should be sufficient for our device to maintain a full-day battery life.

D. Range of Ultrawideband Transceivers

We would like the installation of our UWB anchors to not be excessively expensive, at a cost of approximately $50 for a corner of a hallway. To do this, we would like our transceivers to span the distance of most hallways (with longer hallways such as those in Wean Hall being divided into two). Given this constraint, we note that many hallways in academic buildings at CMU have hallways that span approximately 25 meters before meeting an "intersection" or a wall. The lengths of hallways in academic buildings are all different and there are many exceptions to this rule; however, from our brief experiences measuring out the lengths of various hallways, we believe this distance is typically sufficient. Hence, to keep the costs of our ultrawideband receivers on the lower end, we would like our UWB transceivers to have 25 meters of range.

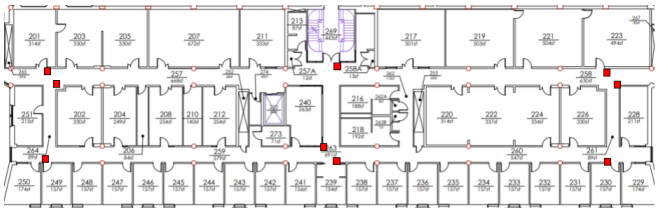


Fig. 3. Example locations of anchors, in red boxes, to map out floor 2 of ReH

Figure 3 displays an example of how anchors could be installed inside floor two of the Roberts Engineering Hall to map out the major walking areas, such that we have at least three anchors with direct line of sight in each hallway for trilateration. Despite the horizontal hallways spanning further than 25 meters, the edges of the main hallways are broken up by vertical hallways into lengths of less than 25 meters. By installing sets of anchors in these corners, we are able to map out the floor.

V. Design Trade Studies

A. Wireless Technology

One of the most important areas of contention when we came up with a solution was the technology we wanted to use for indoor localization. The choice for selecting a technology was motivated by our use case requirements, as we wanted to create a system with highly accurate tracking, while still keeping our system costs relatively low. The main wireless technologies we researched were Wi-Fi, BLE, and UWB, as these were all involved in previous research in indoor localization. We wanted to focus primarily on determining a technology's localization accuracy, then the range of the devices, before also considering the propensity of the signals to be subject to interference, and finally, cost.

Table I. Wireless Technologies Tradeoffs

Technology	Characteristics		
	Accuracy	*Indoor Range*	*Infrastructure Price*
Wi-Fi	3 m	50 m	$0
BLE 5.1	1 m	100 m	$42
UWB	0.3 m	100 m	$75

Table 1 summarizes some of our general findings regarding the characteristics of the technologies we were interested in exploring. We found the accuracy of technologies utilizing results from previous projects and research. Wi-Fi has the worst accuracy at approximately 3 meters [2], while Bluetooth has an accuracy of around 1 meter [3], with UWB having the highest accuracy of around 0.3 meters [4]. The indoor range of these technologies varies slightly with utilizing different devices or the assumption of the number of obstructions in an indoor environment, though we found BLE 5.1 and UWB to typically have higher ranges than Wi-Fi. Finally, we estimated the infrastructure prices of using these devices along a single 50-meter-long hallway, assuming we needed 3 anchors for BLE and UWB (to use trilateration) and 6 anchors for Wi-Fi (due to the difference in range).

Initially, we assumed Wi-Fi would be a good technology to use. Due to the ubiquitous nature of the internet, Wireless Access Points (WAPs) are already built into the infrastructure of buildings. WAPs typically have a high range (up to 50 meters indoors) and are also typically installed in a fashion such that there is some overlap between each of them, allowing us to triangulate the user. Utilizing these WAPs for distance localization could reduce the cost of implementing our system in a building. Additionally, there was a lot of published literature regarding utilizing Wi-Fi in indoor localization. However, our literature review showed the accuracy of Wi-Fi systems to be lacking accuracy; typically they struggled to become accurate to within 3 meters. Wi-Fi inherently falls short because the technology was not built to be an indoor localization technology. Wi-Fi finds the distance between devices with RSSI, which lacks resolution, as well as faltering in the face of obstructions, such as walls, rendering Wi-Fi subject to interference.

BLE was another technology we researched. In terms of localization, it functions quite similarly with respect to Wi-Fi, also utilizing RSSI for distance calculation. Our research found that the newer BLE 5.1 version was capable of localization accuracy to around a meter with a direct line of sight, which is a level of localization accuracy that falls slightly short of our goal of less than one meter. The range of BLE 5.1 is greater than that of Wi-Fi at around 100 meters; however, a localization system would require the installation of a large number of these devices. Assuming each device consists of a transceiver and a microcontroller, we find that each device costs around $14, or a cost of $42 for localization in a hallway. Finally, due to Bluetooth also relying on RSSI, it also suffers from being prone to interference from obstructions.

The third technology we researched was UWB. UWB seems to have significant advantages in many aspects that make it more accurate in localization. Importantly, UWB is different from narrow-band systems such as Wi-Fi and Bluetooth as its signals have better multipath resolution, such that UWB signals remain distinct while narrowband signals might overlap in a multipath environment [4]. Hence, when it comes to determining distance, UWB can rely on protocols other than RSSI, such as Time-of-Arrival, which ultimately leads to a significantly higher accuracy of less than 0.3 meters, which is sufficient for our localization system. Our investigation of the range of UWB transceivers showed they were capable of a range of around 100 meters, high enough to cover longer hallways. Additionally, due to UWB being transmitted over multiple frequency bands, it is also less prone to interference than either Wi-Fi or BLE (both of which transmit on the 2.4 GHz band). However, the cost of a UWB system appears to be an issue, primarily due to the newer nature of the technology, and fewer options for transceivers. These devices have a per-device cost of around $25, which makes localization within a hallway to be around $75. Although the prices for UWB devices are higher than other technologies, the advantages they have in accuracy and providing solid range make them the suitable technology for this project.

B. Ultrawideband Board

The choice of which UWB chip would best fit our application ended up being a rather simple choice due to the limited options available for cost-effective UWB solutions. Some manufacturers such as Microchip Technology only sell UWB chips. However, incorporating an UWB into a PCB was beyond our expertise, so we looked for developer boards with a UWB chip and antenna instead.

We examined UWB developer boards from NXP, SPARK Microsystems, and Qorvo, with the results summarized in Table 2. Boards from NXP and SPARK were relatively expensive, costing several hundred dollars for each board, so we opted towards Qorvo, which happens to have significantly cheaper developer boards.

TABLE II. UWB DEVELOPMENT BOARD COST

Device	Unit Price
SPARK SR1010	$999
NXP NCJ29D5	$129
Custom DWM1000 PCB	$27
DWM1001-Dev	$25

[a]

Qorvo sells two varieties of chips, the DW1000 and the DW3000. The latter is the newer, more expensive variant offering slightly better energy efficiency, though qualities such as the range or accuracy are rated as being the same as the DW1000. Qorvo sells multiple packages of their DW1000 chip. One is the DWM100, which simply consists of a PCB with a DW1000 as well as an UWB antenna. We looked into designing PCBs for housing a DWM1000, as well as a microcontroller to control the transceiver, and we found that this would have a unit price of around $27. However, Qorvo also sells DWM1001-Dev boards that encompass the functionality our custom PCB solution would resolve at a lower price of $25 per device. Due to these reasons, we ultimately settled on using the DWM1001-Dev development kits.

C. Choice of Gateway Device

The DWM1001-Dev board we selected as our UWB transceivers is controlled by an nrf52832 microcontroller, which does not inherently have Wi-Fi connectivity. Therefore, we need a way to convert the device into a gateway for the tag to post to the server to update the state of the user's position. Two options we explored to include this functionality are the ESP8266 and the RPi 4. The ESP8266 has the benefit of being a cheaper device at around $8, compared to the RPi 4's higher price of $62. The ESP8266 also has a smaller power consumption while being smaller than the RPi. However, the RPi 4 has the primary advantage of having a massively higher amount of computing power available. With the higher compute, we can be more confident the device can handle running our localization algorithms in the required amount of time in tandem with the extra processing needed to acquire data from our IMU and make location predictions with it.

D. Raspberry Pi to DWM1001-Dev Communication

There are basically two ways to connect RPi to DWM1001: SPI and USB. SPI has a master clock and data is transferred simultaneously in and out. USB uses differential NRZ signaling and the data transferring between RPi and DWM1001-Dev is not synchronized. The advantage of SPI is that it can transfer data in any direction at the same time, and it is slightly faster than USB because it is synchronized. However, we only need a one-way data transaction, and the USB connection is already fast enough (115,200 baud rate). The disadvantage of using SPI is that it will use a lot of GPIO ports, but we still need to connect the IMU to RPi with GPIO ports. The IMU position must reflect the position of the tag, so it must use GPIO ports, but RPi doesn't have enough ports to support both SPI and IMU connections. In this case, USB communication will save the GPIO ports for IMU. When using USB to connect between RPi and DWM1001-Dev, it is also easier for our team to adjust the position of DWM1001-Dev because it will collaborate with IMU to approximate the current tag position on the map. Based on the above reasons, we decided to select USB communication over SPI communication.

E. Handling Multiple Transceivers

There will be situations where the tag could receive signals from more than three different anchors. To deal with this situation, there are two different approaches. One is to change our algorithm from trilateration to multilateration, and the other one is to pick the anchors that are closest to the tag. The advantage of multilateral positioning is that it will create more precise positions of the tag in 3D space. However, since our project is mainly for solving this problem on a single floor, this advantage is not important. Also, the distances will be not accurate enough for anchors that are further away from the tag, especially in the case where the signal from the anchor is blocked by a wall. Based on the above reasons, we stick on implementing trilateration and pick the closest three anchors each time for the algorithm.

F. Frontend Choice

After deciding on using UWB as our localization technology and determining the components that would compose the device, another consideration was the most optimal frontend technology that could be used to communicate information effectively to the user. On this front, there were several viable options. We could design an app for mobile devices, a Python application, or a Django web application.

Implementing a mobile application is somewhat of a logical option, due to the plethora of other navigation apps that many people already use on their phones. Hence, this option could produce an application that would be fairly intuitive for users to use. However, the members of our group have fairly limited experience with mobile development, and we decided it would be excessively challenging to pursue this approach. Another option was to take advantage of the computing power of the RPi to host a graphical Python program. By adding a screen, we would be able to display directions while running the localization algorithms, which would keep all of the computing localized on one device. However, this option has the issue of leading to a worse user experience, as we would need to also come up with a user interface and work out an optimal method for considering user input.

We ultimately chose to use a Django webserver due to our previous expertise in developing Django apps. Django applications tend to be particularly flexible, allowing us to design an application that could be used on any mobile device or laptop. By having the webserver handle requests, we would be able to easily have a sufficient amount of connectivity between the user interface and their physical tracker.

VI. System Implementation

A. Anchor Implementation

From our block diagram, the anchors of our design will consist of the DWM1001-Dev boards. These boards have a DWM1000 module which has functionality as the UWB transceiver. We will configure these boards to function as anchors, such that they will always be listening for UWB messages, and be ready at any time to respond to requests from the tag to initiate an exchange of Two-Way Ranging (TWR) Time of Arrival (TOA) data to calculate the distance between the tag and the anchor. They will be set with a specific id such that the tag and the anchors will be in the same system network. Thus, the tag will be able to figure out the anchors that it can communicate with. Each of the anchors will have a DC jack to be plugged into a nearby wall socket, with the device itself being mountable on any wall with double-sided adhesive, the higher the better, as it gives more range than if placed on the floor. For easy mapping purposes, since all anchors are two-way communicators, the devices can also talk to each other to get their distances from one another. Such an array of distances could be used to create a scale-accurate 3D map of all the anchor points, which we need only to align with the map of the building to get the positions of all anchors relative to the building. Our code will set the initial positions of these anchors on the map, and then the anchors can adjust themselves automatically by measuring the distances with other anchors. This is a routine we will use to set up the system initially, after which the anchor positions will be remembered and used to locate the user when they perform trilateration on their distances from each of the anchors.

B. Tag Implementation

The tag also consists of a DWM1001-Dev board, except it is also considered a gateway device. The DWM1001-Dev board is controlled by an nrf52832 microcontroller, which does not have built-in Wi-Fi, hence, we will connect the development board to a RPi 4 over USB. The RPi will constantly ping UWB devices by communicating over USB by requesting readings from DWM1001-Dev. This DWM1000 chip will be running multiple threads, each one performing Two-Way Ranging (TWR) with an anchor to get the distance between itself and the anchor. Once it has established these distances between itself and the various anchors, it will then transfer the distance data to RPi through serial port. The RPi will then run localization algorithms such as trilateration for the approximation of current location relative to all the anchors and the centroid of the triangle to find the more accurate location. The tag will also be connected to an IMU. By communicating with the IMU over SPI, the tag will be able to use algorithms to find the orientation of the user. Then, the RPi will send all of the localization information to the webserver by utilizing the Python requests library to post information to the server's database. The tag device will be powered by a 5V power bank, which will be plugged into the RPi, while the DWM1001-Dev board will receive power from its USB connection to the RPi.

C. Localization Algorithms Implementation

The most important algorithm that will be used for localization is the trilateration algorithm. Trilateration is the use of distances (or "ranges") for determining the unknown position coordinates of a point of interest. [5] Based on measurement of the times of arrival (TOAs) between anchors and the tag, we could get the distances between the tag and the surrounding anchors. A point (x, y) on the Cartesian plane lies on a circle of radius r_1 centered at (cx, cy) if and only if is a solution to (4).

$$(x - cx)^2 + (y - cy)^2 = r_1^2 \tag{4}$$

With the same reasoning, we can derive equations for the circles generated by the three different anchors. Since each one has its own position, we can express their positions with (x_1, y_1), (x_2, y_2), (x_3, y_3). The radius of the circles the form can be expressed as r_1, r_2, r_3.The problem of trilateration is solved mathematically by finding the point $P = (x, y)$ that simultaneously satisfies these equations of these circles.

$$(x - x_1)^2 + (y - y_1)^2 = r_1^2 \tag{5}$$

$$(x - x_2)^2 + (y - y_2)^2 = r_2^2 \tag{6}$$

$$(x - x_3)^2 + (y - y_3)^2 = r_3^2 \tag{7}$$

Equations (5), (6), and (7) are based on the assumptions that the circles that the anchors generate meet at a single point, and the signals of the anchors can form perfect circles. However, in real life, there could always be cases when the circles don't meet at a single point, and the signals can form perfect circles. Thus, we need to find the point that best approximates the tag position. Given a position X, we can estimate how well it replaces the ideal precise location P. We can do this simply by calculating its distance from each anchor. If those distances perfectly match with their respective distances, then X is indeed P. The more X deviates from these distances, the further it is deviated from P. In this case, we can treat trilateration as an optimization problem. The goal is to find the point X that minimizes the error from X to P. Suppose the deviation from the ideal distances d_1, d_2, d_3 with the actual distances between the tag and the anchors are e_1, e_2, e_3, and the anchor's locations are L_1, L_2, L_3. We need to find the position X that minimize the following equation sets.

$$e1 = d_1 - dist(X - L_1) \tag{8}$$

$$e2 = d_2 - dist(X - L_2) \tag{9}$$

$$e3 = d_3 - dist(X - L_3) \tag{10}$$

We can use the mean squared error to express the overall deviations from the point X to the ideal location of the current position.

$$\frac{\Sigma(d_i - dist(X - L_i))^2}{N} \tag{11}$$

To minimize the error, we will use algorithms that get the centroid of the triangle to find the point X. The area

overlapped by the three circles will form three vertices v_1, v_2, v_3. By using perspective correct interpolation, we can find the centroid of this triangle for the final decision.

$$\frac{\Sigma V_i}{N} \tag{12}$$

Figure 4 is a demonstration of how the general trilateration algorithm works.

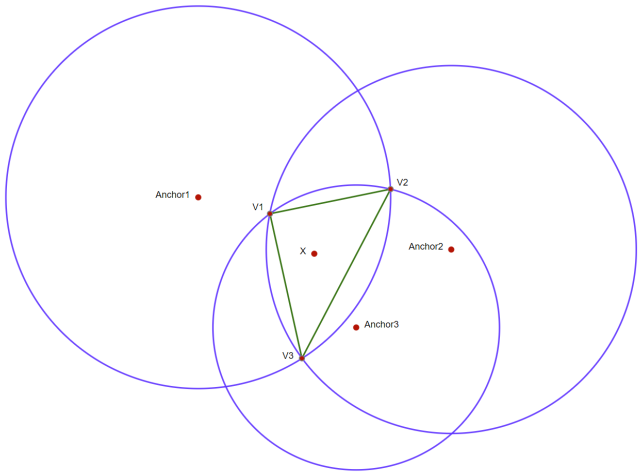


Fig. 4. Trilateration algorithm demonstration

D. *Web Application Implementation*

The user is able to interface with the webserver, accessing the webapp using the browser on their phone or laptop. The home page of the application displays a map centered over the CMU campus, along with some pins the user can click to access for further information of each of the academic buildings. The user will be able to pan around the map and view different portions of the campus as necessary. A mockup of how this would look like on a phone is displayed in Figure 5.

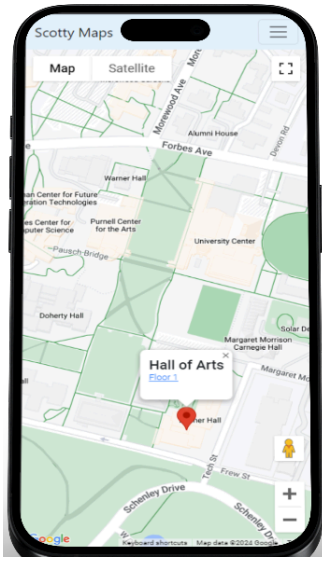


Fig. 5. Mockup of the home screen of our application.

Once the student arrives in a building, their browser will try to use geolocation to find which building they are in, which will cause the web application to open up a map of the building. A mockup of this view is displayed in Figure 6. The student will be able to confirm they are indeed at the building, and then will be able to enter the room number they would like to go to as their destination. The webserver processes this information and runs the navigation algorithm, using the A* pathfinding algorithm for finding the closest path between the student and their destination, as well as providing them with written directions to aid them with their travel. As the student moves, their position (as displayed on the webapp) will be constantly updated by using Javascript AJAX calls to the webserver to ask for data, showing the student's current progress. Javascript will also be used to draw the desired path the student should take to move forward.

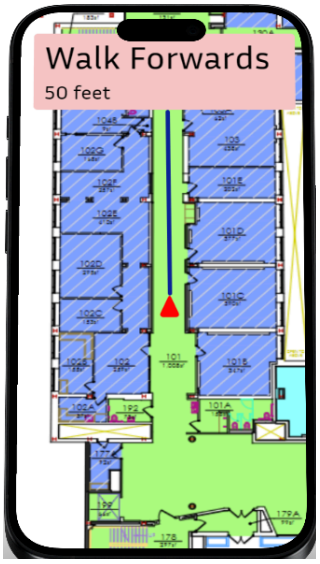


Fig. 6. Mockup of the navigation view of our application.

E. *Directions Implementation*

Once we know the location of the user (via trilateration) and the direction they're facing (via IMU), as well as which way they need to go (via the pathfinding algorithm), we can give them directions for what to do next in order to reach their destination. When a user is walking down a long hallway, we will direct them to "Walk Forward", with additional mention of how long they must walk until they need to make the next turn, whether that be 50ft, 100ft, etc. Once the user is within 10ft of the turn, we will instruct them to "Turn Right"/"Turn Left". Once their rotation confirms that they have completed this turn, we will go back to instructing them to walk forward. When they are within 5 feet of the door of the room they asked to navigate to and facing the door, we will inform them that "You have reached your destination!" If at any point the user is facing opposite the direction of intended travel, we will simply direct them to "Turn Around". To compute the direction of intended travel, we will use the A* gradient of each grid cell of our map. That is, we will compute for each cell the direction of reduced travel distance to the destination (easily computed by taking the average of the gradient of

distance from the target across all neighboring cells).

F. Mapping Process

As mentioned previously in the "Anchor Implementation" section, we will develop a routine where all anchors communicate with each other to find their relative positions to one another. This works by each pair using TWR to get their distance between each other, followed by sending these distances to a laptop, on which we can solve a simultaneous equation of all their distances to get their relative positions in 3D space. From there, we just need to align this map of anchor points with our map of the building (which we will obtain from CMU's online repository of building plans [6]). To do this, we can stand at exactly 3 points (at a minimum; 4 or more would likely increase the robustness and accuracy of the alignment) and mark where we are currently standing on the map. These marked points can then be employed, along with the corresponding points given by our tracking system, to solve for a location and rotation transformation that fits the anchor points to the map. This way, for any point provided by our tracking system, we can obtain the corresponding point on the map. From there, we can divide the map into a grid of cells with a size of 10 cm, which creates a grid detailed enough to fit slanted or circular hallways. Using this grid, we can run the A* pathfinding algorithm, which requires a space to be discretely tiled to provide navigation across it. Additionally, we would need to demarcate the grid with walkable zones and obstacles, which the A* pathfinding algorithm would also need to provide accurate navigation that steers around rooms and other obstructions. For the purposes of our demo, we are not mapping stairs or elevators, as we will treat every floor as a separate self-contained map.

VII. Test, Verification and Validation

A. DWM1001-Dev Range

To ensure we have sufficient connectivity inside of a building, we need to find the maximum distance at which two DWM1001 boards can communicate with each other. We can use a tape to measure the distance between the two devices. We need to make sure the maximum range is at least 25 meters to meet the range discussed in the design requirements. Furthermore, we will test to make sure that the distance remains accurate even through walls to ensure a useful indoor range.

B. Localization Accuracy

To test the accuracy of our localization system, we will stand at random positions within the hallways of the building and use a measuring tape to measure our distance from two walls to find our location in 2 dimensions. We will then obtain an estimated location from our localization system. Then our physical location will be compared to the result from our localization system by finding the difference. After taking a few dozens of these measurements, we will take an average of the measured errors. To pass this test, our average error should be less than 1m, and we will consider that our system is accurate enough to measure the position of the user within a margin of error the size of most building features. We will create situations where anchors and the tag are blocked by the walls and repeat the above process.

C. Navigation Optimality

To ensure the optimality of the travel paths proffered by our navigation system, we will use the A* pathfinding algorithm on a map of the building floor plan that has been divided into a grid and has had rooms and other obstructions marked as untraversable areas. The A* pathfinding algorithm mathematically guarantees an optimal travel path, so we need only verify that our implementation of the A* pathfinding algorithm is correct, which we can do by benchmarking our code against a library of test cases, which we will obtain from coding challenge websites such as HackerRank and LeetCode.

D. Latency of Localization Updates

Testing the latency of our localization system in obtaining its position is important to ensure we can have a high enough update frequency. We need to measure the total latency of our localization system, from the communication protocols in getting distances between the tag and the anchors to the latency of the algorithms for localization. The overall latency should be less than 500 ms.

Additionally, we also need to make sure that updates to the user position in real life are updated on the user interface in a timely manner. Upon finding a new update for the user's location, we need to timestamp this message before sending it to the webserver. Then, we can compare this timestamp to the time the Javascript in the browser receives the data, taking the difference to find the latency present. We would like this latency to be within 2 seconds.

E. Tag Device Convenience

To verify that our tag's battery life lasts long enough to meet our requirements, we will start a navigation session with the device, and leave it running. If the device is still powered on after 4 hours, then we will have confirmed that our battery life lasts long enough to aid a student throughout the day as they are walking indoors. We will also measure the dimensions of the final package of the tag to ensure that the volume is less than 2 liters in size, ensuring it is not too much of a burden to carry around.

F. Quality of Directions & User Experience

To validate that the directions that we deliver to the user are understandable and that our user interface is easy to use and successfully guides the user to their destination quickly and easily, we will pilot the app with several testers. These tests will be carried out whilst testing our navigation system. We will give our clients the tag device, and tell them a room in the building to navigate to. They will then have to enter this destination into the app and follow the directions it gives until they reach this room. After each trial, we will ask them for qualitative feedback and a rating out of 5 based on the user experience of the app, as well as the quality of the instructions provided. Using their feedback, we can work on

improvements to any features the clients had issues with. We plan to execute this feedback until we get at least 4 out of 5 on average across a round of 5 test users.

VIII. Project Management

A. Schedule

Our schedule is displayed in Figure 7. We will work on the localization system and the navigation system in parallel, before reconvening later on in the semester to make sure that the two subsystems can properly work together. By the mid-point demo milestone, we would like to have accomplished localization in a hallway, along with some basic connectivity to the user's phone to display their position. Two weeks afterward, we have our navigation system milestone, where we would like to have successfully been able to test the device from within an entire floor in a building. The final milestone is the final demo. At this point, we would have liked to troubleshoot our design to improve the localization and the navigation system, resolving issues coming about our testing and validation.

B. Team Member Responsibilities

Jeff Chen will be responsible for the web application portion of the project. He will set up the web server and build the app to run in the browser of a mobile phone. He will also be responsible for designing the entire user experience, including destination settings, updating the real-time map (with the user's position marked on it), and providing navigation directions. He will work with Weelie to build the communication endpoints to support the tag's communication with the server.

Ifeanyi Ene will be in charge of the anchor devices. This includes designing and building the anchor devices, as well as programming them to provide the right data to the tag device as it is needed. He will also be responsible for everything to do with the mapping side of the project. This includes the actual mapping process as well as implementing the pathfinding algorithm.

Weelie Guo will handle the tag device. He will be responsible for designing and building the device that will pair with the user's smartphone to aid in navigation. He will implement the localization routine that will get the distances from the anchors and use them to estimate the user's position. He will also write the code for the tag device to provide this information to the smartphone.

C. Bill of Materials and Budget

See Table III for the full Bill of Materials.

D. Risk Mitigation Plans

There are several risks associated with our project that could hamper our progress. The primary issue is that we have limited experience with development on the DWM1001 development boards. These boards are controlled by the nrf52832, which our group has no prior experience programming. Hence, if there are any issues coming up regarding the chip's performance or capabilities, we might need to pivot towards using another microcontroller that could better suit our needs, such as the ATMega328p or the ESP32.

Additionally, it could simply be that the range of the DWM1001 or the accuracy in a small indoor environment, such as a hallway, could prove to be insufficient, or the accuracy is lacking. If these factors arise, we could consider pivoting to a different technology such as Wi-Fi.

IX. Related Work

The indoor localization aspect of our project is similar to the Inexpensive Sports RTLS System capstone project that was done in Spring 2020. This team also used DWM1001 development boards to do indoor localization. Instead of using TWR TOA for localization purposes, this group used TDOA instead.

Additionally, in Spring 2023, a capstone team developed a project called "WiSpider", which also used wireless signals to localize devices within buildings. However, they used Wi-Fi signals rather than UWB. Furthermore, they used pre-installed CMU Wi-Fi access points rather than building their own anchor devices.

We are designing a navigation system interface heavily inspired by many extremely popular navigation apps such as Google Maps, Apple Maps, and Waze. However, instead of a focus on mapping the globe, we will instead focus on mapping out CMU academic buildings.

X. Summary

In conclusion, the project we are proposing seeks to change the way students navigate campus, by providing reliable and accurate information about exactly where they are and how to get to anywhere they might want to go. No longer will a student who has an event in a room they have never visited before have to settle for using Google Maps, which will help direct them to the building, but no further. No longer will a student who is not sure how they got somewhere be stuck or lost, either having to ask for directions or wander aimlessly looking for an exit if no one is around. Instead, any student who wants to be anywhere, as long as they have this device, can simply pull out their phone, open up the web app, type in the room they are looking for, and almost instantly receive directions to their desired destination.

To achieve this vision and this impact, we must find a way to assemble a fully working device for under $100, that can communicate with both the user's smartphone as well as our network of UWB access points. These access points must provide constant and accurate measures of their distance from the user. We must be able to use all this distance data to localize the user to the building map and calculate the path they must take from where they are to their destination, all of this computed in real-time. All this must also be done efficiently enough to give the user a full day's worth of battery life. And finally, all this must be collected into an interface that the user can easily operate and understand, to get them easily, reliably, and quickly to their destination.

Glossary of Acronyms

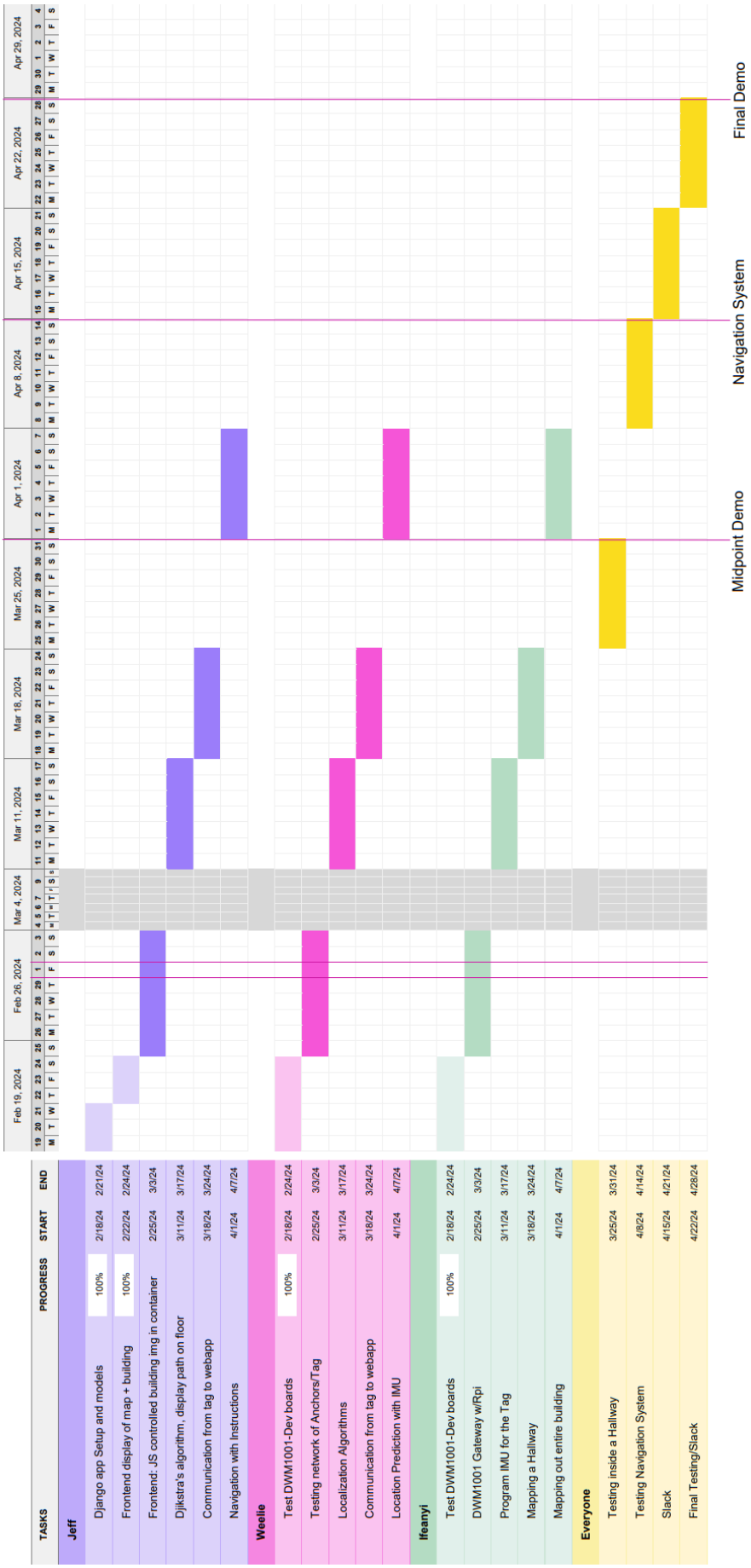
BLE – Bluetooth Low Energy
RPi – Raspberry Pi
RSSI – Received Signal Strength Indicator
TDOA – Time Difference of Arrival
TOA – Time of Arrival
TWR – Two-Way Ranging
UWB – Ultrawideband

References

[1] DecaWave. 2014. The implementation of two-way ranging with the DW1000.
[2] Yuntian Brian Bai, Suqin Wu, Guenther Retscher, Allison Kealy, Lucas Holden, Martin Tomko, Aekarin Borriak, Bin Hu, Hong Ren Wu, and Kefei Zhang. 2014. “A new method for improving Wi-Fi-based indoor positioning accuracy”, Journal of Location Based Services, 8:3, 135-147, DOI: 10.1080/17489725.2014.977362
[3] Ramirez, Ramiro, Chien-Yi Huang, Che-An Liao, Po-Ting Lin, Hsin-Wei Lin, and Shu-Hao Liang. 2021. "A Practice of BLE RSSI Measurement for Indoor Positioning" Sensors 21, no. 15: 5181. https://doi.org/10.3390/s21155181
[4] Mazhar, F., Khan, M.G. & Sällberg, B. Precise Indoor Positioning Using UWB: A Review of Methods, Algorithms and Implementations. Wireless Pers Commun 97, 4467–4491 (2017). https://doi.org/10.1007/s11277-017-4734-x
[5] F. Thomas and L. Ros, "Revisiting trilateration for robot localization," in IEEE Transactions on Robotics, vol. 21, no. 1, pp. 93-101, Feb. 2005, doi: 10.1109/TRO.2004.833793.
[6] Building Floor Plans and Room Information. Carnegie Mellon University. (n.d.). https://www.cmu.edu/finance/property-space/floorplan-room/index.html

TABLE III. BILL OF MATERIALS

Description	***Model Number***	***Manufacturer***	***Quantity***	***Cost***
Raspberry Pi	4	Raspberry Pi Foundation	1	$62.00
IMU	GY-521	HiLetGo	1	$3.33
UWB Development boards	DWM1001-Dev	Qorvo	12	$25.00
Li-Ion Batteries	16340	CWUU	11	$2.50
Battery Pack	5000mAh	YUMGOOD	1	$10.00
Total Cost				**$402.83**



TASKS	PROGRESS	START	END
Jeff			
Django app Setup and models	100%	2/18/24	2/21/24
Frontend display of map + building	100%	2/22/24	2/24/24
Frontend: JS controlled building img in container		2/25/24	3/3/24
Dijkstra's algorithm, display path on floor		3/11/24	3/17/24
Communication from tag to webapp		3/18/24	3/24/24
Navigation with Instructions		4/1/24	4/7/24
Weelie			
Test DWM1001-Dev boards	100%	2/18/24	2/24/24
Testing network of Anchors/Tag		2/25/24	3/3/24
Localization Algorithms		3/11/24	3/17/24
Communication from tag to webapp		3/18/24	3/24/24
Location Prediction with IMU		4/1/24	4/7/24
Ifeanyi			
Test DWM1001-Dev boards	100%	2/18/24	2/24/24
DWM1001 Gateway w/Rpi		2/25/24	3/3/24
Program IMU for the Tag		3/11/24	3/17/24
Mapping a Hallway		3/18/24	3/24/24
Mapping out entire building		4/1/24	4/7/24
Everyone			
Testing inside a Hallway		3/25/24	3/31/24
Testing Navigation System		4/8/24	4/14/24
Slack		4/15/24	4/21/24
Final Testing/Slack		4/22/24	4/28/24

Fig. 7. Gantt chart with our planned schedule.