Individual Final Report

Benjamin Williams

R11544055

13 November 2022

This report details the engineering process behind Project Lab III's Digital Communications project. The project is to retrieve three codewords from three notebooks hidden around Urbanovski Park by guiding a rover to coordinates through radio communication from a base station. No existing infrastructure, such as GPS, Internet, or Bluetooth, is allowed. The rover's location must continuously be tracked through the base station's Urbanovski Park map GUI. Credit will be given to the interceptor if another group intercepts a codeword. Given this description, the following tasks outline the engineering process. The first requirement was selecting a design for rover tracking and collecting materials for the build. Second, creating a method of intercepting other groups' codeword transmissions. Finally, using the materials ordered, create a prototype Adcock antenna array for radio direction finding to track the direction of the rover's incoming signals and test the prototype's resonance frequency using a network analyzer.

Initially, a method for finding the Angle of Arrival or direction of the rover from the base station was selected (Reference 5). The chosen method is the Watson Watt Adcock Antenna (Reference 17). This design includes five antennas. Four antennas are placed in a square facing toward specific directions: North, East, South, and West. The North and South Antennas are connected end to end to form a dipole, and the East and West antennas are connected end to end, forming a dipole. These two dipoles output relative magnitudes depending on the direction of the signal around the array. The bearing angle of the incoming signal is the inverse tangent of the East and West dipole's magnitude over the North and South dipole's magnitude. However, the inverse tangent can only describe an angle between -π/2 and π/2 radians. Therefore, a signal approaching from the South-East appears the same as a signal approaching from the North-West. A "sense" antenna is required to combat this uncertainty. This antenna is located in the center of the array to capture the phase angle of the incoming signal. This phase angle differentiates a South-East signal from a North-West signal or any other combination (Reference 15).

Figure 1: Watson Watt Adcock Antenna Diagram

Chart, radar chart

Description automatically generated

1 Watson Watt Radio Direction finding method. It depicts two dipole radiation patterns colored blue and green with a black line circling, representing an incoming signal direction. The magnitudes of each dipole are depicted on the right. A positive or negative sign is derived from a sense antenna and associated with each magnitude (not shown).

Next, a block diagram was used to outline the materials needed to create the Watson Watt Adcock design (Reference 7).Diagram, schematic

Description automatically generated

2 Watson-Watt System Block Diagram depicts the two dipole configurations and some general requirements for signal processing afterward, such as RF to IF modulators, signal mixers, and filters.

The components were selected using this diagram as a basis for requirements. A component is needed to handle the Radio Frequency (RF) to Intermediate Frequency (IF) conversion. Then, an Analog Digital Converter (ADC) capable of sampling an IF signal such that the Raspberry Pi or Arduino Uno would be able to interpret those signals for the bearing angle calculations.

For RF to IF conversion, I found a TB03 from Mini-Circuits (Reference 11). This device incorporated an RF to IF demodulator and an IF amplifier. This mixer will take a high-frequency radio signal of 915 MHz and step it down to an Intermediate frequency of 10-15 MHz. An image of this device is shown below in Figure 3 (Reference 12).

Figure 2: TB03 Mixer

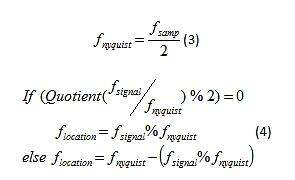
A close-up of a circuit board

Description automatically generated with medium confidence

3 This mixer will require a Local Oscillator (LO) that creates a steady sine wave of 900 MHz. When the 915 MHz signal is mixed with the 900 MHz from the LO, the output should step down to a 15 MHz IF signal.

Next, the build needs an Analog to Digital Converter (ADC) capable of sampling the 15MHz IF signal. According to Nyquist's Theorem, the ADC must be capable of sampling at least twice the input signal frequency to reproduce the analog signal digitally at some degree of accuracy. Ideally, more than twice the frequency of the sampled signal may be desired for a more accurate measurement. The Nyquist Theorem detailed above is shown in Equation 1 (Reference 13).

Equation 1: Nyquist Theorem



For this reason, I've selected an ADS930E capable of a 30MHz sampling rate. Since the signal received is 915 MHz; however, ADCs get exponentially more expensive as you increase the max sampling rate assuming they maintain guaranteed accuracy. For this application, the ADS930E should suffice. The ADS930E device is shown below in Figure 4 (Reference 14).

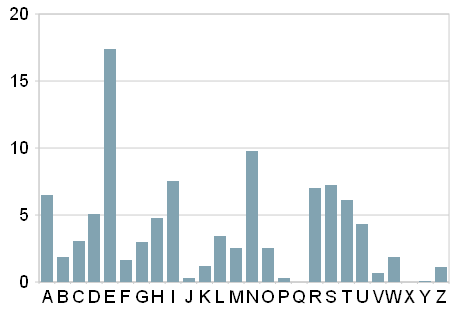
A picture containing electronics, circuit

Description automatically generated

4 ADS930E Analog to Digital converter capable of sampling 30 MHz—2 times the 15 MHz IF signal outputted from the RF to IF mixer, satisfying Nyquist's Theorem.

Next, a method of intercepting other groups' codeword transmissions was required. A python program was designed to be capable of cracking the Caesar Cipher encryption algorithm for this purpose. The Caesar Cipher encryption algorithm is the most basic encryption technique, which involves shifting the alphabet by a certain number of characters. For example, take the word "Hello" and use a Caesar Cipher key of five characters; the encrypted result would be "Mjqqt." As many as five groups intended to use this encryption method. Consequently, this program was designed to hard crack the Caesar Cipher by cycling through all twenty-six key combinations and finding the most likely character shift amount that results in an accurate English word. Theoretically, this method could work on different ciphers if they apply the same shift to each character.

The method comes from Robert Eisele (Reference 6), who implemented this algorithm in PHP, which was then reformatted to execute in python since a raspberry pi is set to control the base station. This program uses a weighted histogram array corresponding to the frequency that each letter in the English language shows up in a word. By comparing the frequency of each character in the encrypted message to the frequency of each character in the English language, the program can record the closest match over each iteration and then return the corresponding character shift amount used to create that ciphered message. The frequency chart or histogram is shown below in Figure 5 (Reference 6)



5 Histogram of how often each letter in the English language appears across all English words. Note the extreme difference in the letter 'e'—the most common letter—compared to q, x, or y—the least common letter.

First, the program takes these weight values into a NumPy array and then uses two empty arrays to store the count and the frequencies of characters in the ciphered message. Next, the program goes through a double nested for loop, which cycles through each letter of the alphabet twenty-six times to represent different character shift amounts. Each iteration counts the frequency of each letter in the message, then calculates the distance of that distribution versus the English distribution in Figure 5. Finally, the program checks to see if the distance is smaller than the currently recorded smallest distance, and if so, sets that shift amount as the new key guess. The best key guess is used to subtract the shift amount resulting in the original message.

Using object-oriented programming skills, all of Robert Eisele's algorithm was encapsulated within a class called 'RomanSenate.' Revealing the method behind decryption defeats the purpose of decrypting. Decryption also required another class called 'JuliusCaesar' since the decryption program requires a ciphered message at the start. The purpose behind consolidating the code into two separate classes was to emphasize that the deciphering algorithm has no outside information regarding the message it is tasked to decode. By separating these classes and only allowing a ciphered message to be passed into the 'JuliusCaesar’ class, there is no doubt that the decryption algorithm functions independently. The two classes described are shown below in Figure 6.

Text

Description automatically generated

6 Encapsulated Code JuliusCaesar.py and RomanSenate.py. Used to demonstrate independence between other groups’ ciphers and the deciphering algorithm.

The only downside to this method is that its accuracy depends on the amount of data collected using the same key. This program requires around 3 simple codewords to guess with a degree of certainty unless the words are long and contain many characters. The more data provided is directly proportional to the accuracy of the decryption algorithm. For example, short words like “hello” is decrypted correctly unless more of the ciphered messages sent are recorded. Once the key is guessed, that key can be used to decrypt their previous codewords or watch for the next one. To capture the messages in the first place, an RTL-SDR (Software Defined Radio) will be required to intercept and interpret other groups' transmissions (Reference 8). This device will allow scanning across frequencies to record the data being sent. It can also be used for the direction-finding system possibly incorporating all the necessary mixers and filters in the design into a programmable software application. This device is shown below in Figure 9. (Reference 4)

Text, letter, whiteboard

Description automatically generated

9 RTL-SDR depiction. A tiny USB device with an SMA adapter and a multitude of filters, mixers, etc., all able to be logically controlled through software.

The figure shows that the device looks like a simple USB dongle with an SMA antenna connector. Inside this device are many filters and programmable logic (Reference 10), which I can use to manipulate data discovered over the radio waves. Additionally, the device can scan the majority of the Radio Frequency Spectrum, at least up to 1GHz (Reference 10), and tune into any selected signal. For starters, the device could look at different radio FM stations and tune into them using Wideband FM, allowing the user to choose a center frequency and adjust the bandwidth to capture more or less of any given signal that appears within the SDR’s frequency band. This device was tested by tapping into radio stations such as 102.5 MHz Kiss FM and outputted audio from the song Buddy Holly in the Lab. Then, the device also proved to capture signals from our 915 MHz transmitter.

The RTL-SDR will be valuable for this project through the use of SDR# or GNU radio which has syntax similar to C or C#, but designed for the purpose of interpreting and processing radio signals from the RTL-SDR. Essentially, allowing the user to directly code or create a flowchart controlling the signal processing of a receiving signal.

Finally, the prototype of the Wattson Watt Adcock antenna array was built for receiving 915 MHz. Recall that the theory behind this array is that by comparing the relative strength of each of the dipoles, the angle of arrival of the incoming signal can be determined by the arctangent function. To create this build, a diagram related to a fox hunting Adcock antenna was used as a reference. This theory diagram is shown in Figure 10.

Diagram

Description automatically generated

10: Build concept of Wattson Watt Adcock Antenna array. Featuring 4 1/8th λ antennas spaced 1/4th λ apart, dipoles connected with 75 and 50 ohm coax cables.

Before building a prototype based off this design, the value of lambda—wavelength–is required for each measurement in the diagram. To derive the wavelength of the desired 915 MHz signal, the equation below was used:

(Reference 4)

Using this equation with the constant c and a desired frequency of 915 MHz resulted in a measurement of λ equal to 0.32764 meters. The dimensions required for the build are 1/8th λ for the four antennas, each spaced 1/4th λ away from each other. Therefore, the prototype required four 4.1 centimeters long antennas spaced 8.2 centimeters apart.

With wavelength dimensions and a build diagram planned out, the next step involved acquiring the materials for the prototype build—not to be confused with the parts outlined above that will be required for processing the signals outputted from this prototype. The resultant parts list created consisted of parts ordered from Digi-key through the ECE stockroom (Reference 2). This parts list consisted of 18-gauge copper wire for the antennas, a Tee section SMA Female-Male-Female connector, three RG-179 50-Ohm coaxial cables, and two coaxial cable to BNC connectors (Reference 1 & 2). While these materials will be used for the final build, intermediate materials were acquired through the ECE stockroom to create a rough prototype. Therefore, the materials used in the prototype were 20 Gauge twisted copper wire which were used for the 1/8 λ antennas, a surplus of twisted pair coax cables, two coax to BNC connectors, and some PVC pipe that was already on hand from a previous project.

With all the intermediate materials gathered, an initial build of the Wattson Watt Adcock Antenna array was constructed (Reference 2). The resulting build is shown below in Figure 4.

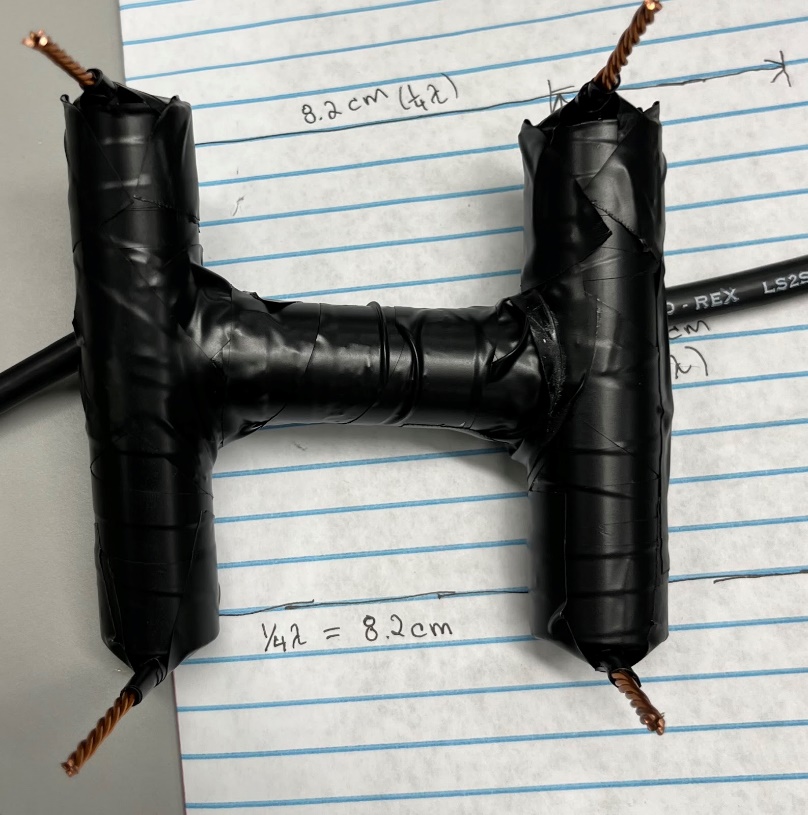


Figure 7: First Build of Wattson Watt Adcock Antenna array using Twisted copper wires, PVC pipes, and Twisted pair Coaxial cables. Spacing dimensions corresponding to wavelength measurements labeled underneath.

Next, this design was tested by connecting each coax to BNC Connector to a network analyzer and identifying the resonate frequencies of each dipole (Reference 1). The network analyzer test resulted in two frequencies similar to the harmonics of the desired 915 MHz signal: 220 MHz and 409 MHz. The 220 MHz signal is only 8.75 MHz off the exact 4th harmonic of the 915 MHz signal, which is 228.75 MHz, while the 409 MHz signal was 48.5 MHz off of the exact 2nd harmonic of 457.5 MHz. Evaluating the harmonics was necessary since the network analyzer is scaled only up to 500 MHz. So by analyzing the 2nd and 4th harmonics, the 1st harmonic can be revealed and tuned into. The resonate frequencies displayed on the network analyzer below in Figure 8.

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence

8 Prototype Watson Watt Adcock build connected to a network analyzer. The network analyzer displays the resonate frequencies fundamental to the physical dipole connected to the analyzer. The left image shows the 4th harmonic of 220 MHz, and the right image shows the 2nd harmonic of 409 MHz.

This Adcock antenna design is heavily dependent on the antenna element spacing. Therefore, the resonate frequency can be increased or decreased by manipulating the distance between the two antennas associated in each dipole. Therefore, the prototype will be tested again while manipulating the element spacing until each dipole shows the desired resonate frequency harmonics corresponding to 915 MHz. Measurements are to be taken for use in the final design.

Since the design is centered around ultra-high frequencies, this Adcock antenna design heavily depends on the precision of the antenna element spacing (Reference 3). Furthermore, the resonance frequency can be increased or decreased by manipulating the distance between the two antennas in one dipole (Reference 2). The initial design did not reach the desired resonance frequency of 915 MHz; the 2nd and 4th harmonic resonances were off by 8.75 – 43.7 MHz, respectively (Reference 1). Therefore, the previous design was taken apart to manipulate the antenna spacing to reach the desired frequency. Many tests were conducted using varying length antennae designed for either 915 MHz or 433 MHz since another LoRa transceiver has been ordered in case 433 MHz proves more suitable for this application.

The first test was done immediately after the initial prototype’s disassembly. This test was designed for 915 MHz using 1/8th wavelength, 4.1 cm, long twisted copper antennas spaced 1/4th wavelength, 8.2 cm, apart. This test resulted in about a -30 dB magnitude response at 915 MHz. The results of this test under a network analyzer are shown below in Figure 1.

A picture containing text, electronics, display

Description automatically generatedA picture containing indoor, wall, person

Description automatically generated

Figure 8: First resonance test for 915 MHz. Using twisted pair Coaxial cable, two 4.1 cm twisted copper antennas are connected in a dipole at the BNC connector. The left image shows the resonance correspondence to 915 MHz as desired, and the right image shows the spacing used to create the resonance.

Another test was done for the desired resonance of 433 MHz. This test used two 1/8th wavelengths long, 8.6 cm, twisted copper antennas spaced 1/4th wavelength, 17.2 cm, apart. This test resulted in about a -40 dB magnitude response at 433 MHz. The result of this test under a network analyzer is shown below in Figure 2.

A picture containing text, electronics, display

Description automatically generatedA picture containing person, electronics

Description automatically generated

Figure 9: Second resonance test under a network analyzer designed for 433 MHz. Using twisted pair Coaxial cable, two 8.65 cm twisted copper antennas, connected in a dipole at the BNC connector, antennas spaced ¼ wavelength—17.3 cm apart. The left image shows the results on the Network analyzer, and the right image shows the setup and spacing used for the test.

The results of these two tests were desirable. However, the results do not reflect the intended design since the copper antennas were never physically connected to the coax cables due to their exterior insulation (Reference 4). The exterior insulation was burned off at the coax connecting end, and solder connected each twisted copper wire at the opposite end to fix the antenna connections. Further network analysis testing continuously failed to reach any desired resonance independent of how long the antennas were or how far apart they were spaced. On the other hand, using only the twisted pair within the coax cable in the previous tests without connection to the copper antennas, the desired resonance frequencies were all acquired satisfactorily. Therefore, the copper antennas can be discarded from the design to thread the coax cable pair through a baseplate measured precisely ¼ wavelength apart.

The basis for this new design is almost complete. A 20 cm x 20 cm wooden baseplate was prepared this week, and precise measurements for the spacing will allow the coax antennas to remain in place to receive the desired frequency consistently. A preliminary layout is shown below in Figure 3—disregard the copper antennas, which will be removed.



Figure 10: Preliminary Adcock Antenna array build using a 20 cm x 20 cm Wooden baseplate, two BNC to SMA adapters, and a tee section adapter.

Next, the coax cables will be spaced according to the first resonance test results—8.2 cm apart. Then by using the LoRa transceiver to transmit signals at different directions relative to the array, the resulting North-South and East-West dipole magnitudes can be evaluated through a spectrum analyzer. This test should result in incoming signal direction discovery (Reference 5).

Next it was decided that a transition from the Adcock antenna Angle of Arrival Radio Direction Finding technique to the more common trilateration using Time of Flight translations from distances to relay nodes was necessary. Time constraints influenced the reasoning behind this transition at this late point in the project; with only weeks before demo day, a more reliable method for finding the rover's position became necessary. Therefore, details hereafter regard the final work done at the end of the week for the Watson-watt Adcock antenna and then cover the work done regarding the Time of Flight at the beginning of the following week.

Next, a continuation of resonance testing with the Adcock antenna took place (Reference 2). Since last week's discovery, once the copper twisted antennas were soldered together and properly connected, the desired ultra-high frequency range, 915 MHz, was unable to reach (Reference 4). The copper antennas were not ideal for the resonance frequency desired. Therefore, the copper antennas were removed in favor of the default coax cables, which resonated as desired for 433 MHz and 915 MHz (Reference 3). However, without the antennas, no change was observed on the network analyzer as previously seen when the spacing or parallel length of the cables was manipulated as shown below in Figure 1.

A picture containing text, electronics, display

Description automatically generatedA picture containing indoor, wall, person

Description automatically generated

Figure 11: First resonance test for 915 MHz. Using twisted pair Coaxial cable, two 4.1 cm twisted copper antennas are connected in a dipole at the BNC connector. The left image shows the resonance correspondence to 915 MHz as desired, and the right image shows the spacing used to create the resonance.

Manipulating the resonance with the antennas became uncertain as the connection between the cable and copper came into question. The coax cables would need to be used alone to move on with the antenna build. However, the spacing and length relationship to resonance appeared to no longer apply when manipulating the coax cables alone (Reference 1). This uncertainty and a shortening time constraint led to the side-lining of this direction-finding method in favor of trilateration nodes for this project.

For creating nodes, a decision between the frequency of the radio signals used for communication was necessary before ordering the extra transmitters for each relay. Therefore, a test was conducted to determine the frequency most dependable for on-campus radio transmissions. Two 433 MHz LoRa transmitters were tested to determine the relationship between radio frequency and range. The test showed that, even over a shorter distance than the 915 MHz range testing, 433 MHz could only receive a signal during direct line-of-sight while immobile. A diagram of the path taken for this range test is shown below in Figure 2.

Map

Description automatically generated

Figure 12: Range testing for 433 MHz Lora Transceivers. This image shows the route taken and the maximum range of the 433 MHz transmitters while the RSSI, Relative Signal Strength Index, appeared to be half that of the 915 MHz signal RSSI.

The results of this test were unreliable due to the nature of the LoRa chipset and software, which has been observed to produce unreliable results over various configurations throughout weekly presentations. However, further tests may be conducted using Signal-to-Noise ratios or other more reliable measurements for this transceiver before ordering the transmitters for the relay nodes.

The following steps for the relays are to expand distance and collect Time of Flight data to calculate the rover's position. Since the main communications work was already underway, the next important step was redesigning the GUI for the base station to fit with the new map location for the design. The new map shortens the maximum range required by our rover and relays; therefore, the map is shown below in Figure 3.

A picture containing map

Description automatically generated

Figure 3: New map for Radio Direction Finding project lab that negates most of the campus obstructed by many buildings allowing for more feasibility through varying designs. However, the enormous scale of United Supermarkets Arena directly in the center continues to demonstrate the effects of buildings on radio waves and the blind spots created by smaller buildings in the area.

With the new map configuration and approach, a new base station interface became necessary. Using the familiar "customtkinter" GUI python library, a new GUI was designed as a baseline this week with high expectations for the final design due to previous experience using this library in the professional context of an internship. The baseline interface compared to what the final result is expected to become is shown below in Figure 4 (Reference 5).

A map of a city

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

Figure 13: Baseline Map GUI using the 'Custom TKinter' library in python with comparison from previous knowledge/experience with this library demonstrating the potential for this program.

Next, to remodel the base station’s Graphical User Interface with a new map using Custom TKinter in Python and tracking the rover’s location after the time of flight values are calculated and passed through serial communication from the base station’s Arduino Uno. The program uses four classes to create instances of objects for the G.U.I. application, relays, and notebooks. The base station’s interface allows the user to view an interactive campus map, read distances coordinates anywhere on the map and to the base station, place icons for notebooks, relays, the rover, or arbitrary markers, a dark-mode switch, and a function for drawing time of flight distances around each relay.

This program uses four classes ‘MapItem,’ ‘Relay,’ ‘Notebook,’ and ‘App.’ MapItem is a super-class of Relay and Notebook, and this class has methods to get or set the icon’s position and attach it to the widget object created within App. Relay and Notebook inherit those methods from MapItem, but each has specific attributes for codewords or time of flight and methods for manipulating these values. (Reference 2)

The interactive map uses the Canvas widget to provide helpful information using binds for mouse hovering or clicks (Reference 3). A terminal is placed below the map, updating its text as the user performs different actions. When the mouse cursor hovers over any spot on the map, the terminal displays the distance coordinates and direct distance relative to the base station. If the mouse hovers over one of the function buttons, then the terminal also provides a brief description of the function bound to the button. When the user clicks any location on the map, a pop-up window appears at that location with a dropdown menu allowing the user to select between a relay, Notebook, marker, or rover object to instantiate and place on the map. The main interface described is shown below in Figure 1 (Reference 1).

A screenshot of a map

Description automatically generated

Figure 14: CustomTKinter G.U.I. Interface with an interactive map, functional buttons, text feedback terminal, and various icons to mark the locations of various objects.

The “Calculate ToF” button creates a new window with combo boxes to choose three rovers to draw squares, sized according to the time of flight attribute, around each relay object (Reference 3 & 4). The “create\_rectangle” method of the Canvas widget requires the coordinates of the bottom left vertex and top right vertex. Therefore, to draw the rectangles around the relays, the bottom anchor coordinate must be set to the relay position – the time of flight distance, and the top must be set to the relay position + the time of flight distance (Reference 3). The small box created in between the three relays reveals the approximate rover location. Once time of flight values are calculated through the Arduino Uno, this module will incorporate the PySerial library to update the time of flight values and rectangles periodically. The pop-up window and the updated map canvas are shown below in Figure 2.

A screenshot of a computer

Description automatically generated with medium confidence A picture containing diagram

Description automatically generated

*Figure 2: Time of Flight Module window on the right of the updated map canvas showing drawn rectangles corresponding to Time of Flight (ToF) magnitudes. The relay name is on the right of the module, the current ToF value is shown in the middle, and an entry box allows the user to change the ToF value.*

With testing, it became clear that the LoRa modules would not have enough range to cover Urbanovski park. Therefore, we switched to RYLR998 UART modules for everything. The base station used a USB to UART TTL converter to directly send messages to the rover, and the same was done for the rover. The relays were rebuilt using an Arduino, a battery bank, and a logic level shifter. These relays would simply wait for a message once it sees a message reply with the number of the node, along with its RSSI and SNR values attached.

Finally, the GUI was finished, but the ToF window was scratched in favor of automatic updates through pySerial and using circle intersections instead of rectangles. The final GUI is shown below in the figure. This GUI features a terminal for communication with the rover, a map of the intended area for use, and a useful tip bar at the bottom to describe each feature available.

A screenshot of a computer

Description automatically generated with medium confidence

This interface provides all necessary base station functions: it allows the user to place relays, notebooks, and markers by a simple click and selection. The three main relays for tracking are set at the start since those points will need to be fixed at known locations. Communication works through the terminal. First, the RYLR998 UART [20] module’s parameters are set up with AT Commands [19]. This is done through multiple serial prints to the UART module: "AT+BAND=915000000", "AT+ADDRESS=48", "AT+NETWORKID=12", "AT+IPR=9600". Once two modules were initialized, their range was tested across Urbanovski Park. The results of these tests were incredibly desirable. The UART Modules provided consistent communication everywhere around the park with minimal packet loss [21].

Entering a message and pressing enter will send the program through a script that inserts “AT+SEND=” to the message as well as the message length and address. This ensures that the UART module knows how to handle the message through its AT commands [19]. To send a message the UART module must receive “AT + SEND = {addresss},{ message Length},{message}\r\n” plus a carriage return and newline at the end.

A new tracking algorithm was implemented using Fang’s Method [25], which lets one point be an anchor at (0, 0), then the next point lies along the same axis (x, 0), and the final can be anywhere at (x,y). This reduces the intersecting circle equations, as shown below in the figure [18].

Graphical user interface, text, application

Description automatically generated

Figure 15 Simplifying Intersection equations using zeros [18]

This simplifies the x and y equations into two easy equations to plug in known values and those coming through pySerial. However, we still needed to convert the RSSI values to distance, so after interviewing Brennan and Klay, we implemented their RSSI to distance linear equation of -0.1457(RSSI) – 55.5 to create relevant distances on the GUI [22 & 23]. This equation was derived through RF Soaking which is the process of gathering data from RF transmitters and averaging down to more realistic and measurable values. We had a solid baseline for demo day with all of this together. The GUI and Rover were seamlessly able to communicate. However, the sending of the RSSI values from the rover to Base station encountered errors. Thus, preventing our group from achieving real-time tracking.

Citations

1. Derek A., Johnston, Office Hours, 11 October 2022.

2. Klay Adams, ECE Stockroom, Materials

3. Introduction to Radio Direction Finding, <https://www.alarisantennas.com/blog/an-introduction-to-radio-direction-finding/>

4. Equation and background info on frequency and wavelength relation: <https://www.omnicalculator.com/physics/wavelength>

5. Background Angle of Arrival Information, <https://en.wikipedia.org/wiki/Angle_of_arrival>

6. Robert Eisele, <https://www.xarg.org/2010/05/cracking-a-caesar-cipher/>

7. Read, W. (1989, May). Review of conventional tactical radio direction finding systems. Retrieved September 13, 2022, from <https://apps.dtic.mil/sti/pdfs/ADA212747.pdf>

8. <https://www.rtl-sdr.com/buy-rtl-sdr-dvb-t-dongles/>

9. <https://www.rtl-sdr.com/wp-content/uploads/2013/04/SDRBlogImage_1_Side-150x150.jpg>

10. <https://www.rtl-sdr.com/wp-content/uploads/2018/02/RTL-SDR-Blog-V3-Datasheet.pdf>

11. TB03 Mixer : <https://www.amazon.com/5-1900Mhz-Frequency-Conversion-Passive-RMS-11/dp/B088BR1ZZQ>

13. Nyquist Theorem: <https://www.planetanalog.com/wp-content/uploads/images-common-planetanalog-2014-12-563771-Image-3.jpg>

14. <https://www.ti.com/lit/ds/symlink/ads930.pdf>

15. [https://apps.dtic.mil/sti/pdfs/ADA212747.pdf](https://apps.dtic.mil/sti/pdfs/ADA212747.pdf%20)

16. Jim Williams, Electrical Engineer at Vorago Technologies, Microsoft Teams Meeting.

17. http://www.rdfproducts.com/an006\_apl\_01.pdf

18. <https://journals.sagepub.com/doi/full/10.5772/63246#fig1-63246>

19. <https://reyax.com//upload/products_download/download_file/LoRa_AT_Command_RYLR998_RYLR498_EN.pdf>

20. <https://reyax.com/products/rylr998/>

21. <https://www.instructables.com/LoRa-Distance-Testing-With-RYLR998-in-Open-Field-A/>

22. Quick, Brennan, Interview

23. Adams, Klay, Interview

24. [RadioHead Library to use RFM95](https://learn.adafruit.com/adafruit-rfm69hcw-and-rfm96-rfm95-rfm98-lora-packet-padio-breakouts/using-the-rfm69-radio)

25. Fang’s Method: Fang B.T., Simple solution for hyperbolic and related position fixes, IEEE Transactions on Aerospace and Electronic Systems, Vol. 26 (No. 5) (1990) 748–753.