WiFi Localization

Authors: Txanton Bejos, Vrishab Commuri, Enock Maburi: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—In this report we present a method and a system implementation by which WiFi-transmitting devices can be passively located within a room without the need to analyze signal content.**

***Index Terms*—Beamforming, Localization, Software Defined Radio, WiFi**

1 INTRODUCTION

In many situations it is pertinent to be able to track and identify the locations of devices that are outputting signals of specific frequencies. This need can be seen in areas of security when sweeping a room to find devices that are linked to a network they should not be connected to. Furthermore, inspection protocols that require all devices to be accounted for in an area, such as in police investigations, need a way to identify all devices efficiently. In lesser serious cases companies can track one's device when in their building in order to collect data about their visitors. In this report we in aim to target devices that are transmitting WiFi since it operates in 2.4 and 5 GHz rather than signals like LTE, 4G, etc. which operate in many different frequencies which would be hard to target with one device. Since 3 billion devices ship annually with WiFi enabled it makes it a ubiquitous signal worth specifically analyzing.

This proposed device will implement a digital signal processing (DSP) algorithm called beamforming which will localize the devices by sweeping its Field of View (FOV) to determine the signal strength of all devices it captures to produce a heat map of their locations. It must update the heat map at 30Hz, be able to locate 100% of devices in the room, and locate devices at least 5ft apart based on its 15 degree squared lobe-width.

2 DESIGN REQUIREMENTS

In order to localize devices within 10 feet in a 20x20 foot room, we need an approximate localization resolution of $\arctan(10/20) \approx 30$ degrees. The room size was determined based on an average room size estimate in a study by NIST [5]. Previously, our goal was to locate signals to within 5 foot in a 20x20 room, but due to budget constraints we were not able to afford the antenna hardware to meet this specification. Nevertheless, the constraint of 10 feet is more than enough for most applications.

The localization resolution is determined by the main lobe width of our antenna array. The particular calculations for determining this main lobe width are performed in section 4.2.4.

We are also aiming for a processing latency of at least 30 updates to the localization map per second. This specification was determined because a refresh rate of 30Hz is common and the latency between successive frames is typically not noticeable to the human eye [6].

In summary, our quantitative requirements follow:

- Localize transmitting devices to within 10 feet in a 20x20 foot room.
- Antenna array main lobe width of 30 degrees squared (based on the localization constraint above).
- The system should have a refresh rate of 30Hz so as to appear fluid to the user.

3 ARCHITECTURE OVERVIEW

3.1 Signal Capture Pipeline

We have 4 antennas in a linear configuration. The spacing between each antenna is 6.25cm in order to sample 12.5cm wavelength WiFi signals without incurring artifacts from spatial aliasing.

WiFi signals are transmitted at a frequency of 2.4GHz. Additionally, WiFi signals are transmitted over 11 channels, each with a bandwidth of 20MHz. We have opted to process WiFi signals from channel 6, which in our testing using a WiFi signal analyzer is a very commonly-used channel. We downconvert (downmix) the channel 6 WiFi signals to baseband using an off-the-shelf downconverter module attached to each of our 4 antennas. The downconverted signals are then filtered and converted from analog to digital by an RTL-SDR. The RTL-SDRs are synchronized via their clock pins, and IQ (in-phase and quadrature) samples – essentially the raw signal data – are taken directly from each and fed into the computer via USB.

3.2 Computer Processing

The array of 4 signals coming from the RTL-SDRs are sent over USB to the computer for processing. These signals are sent to the **Xcorr** module, implemented in C and Python, which handles the correlation of signals in the time domain and determines lags between the incoming signals. These signals are sent to the **Beamforming** module which delays and sums the signals to produce a single signal output of received intensities. This output signal will be sent to the **intensity_evaluation** module which handles the calculation of the critical values used to produce the heatmap values. The first value is the maximum intensity value

which is the maximum signal intensity found by the beamformed signal. The second value is the azimuth value which is the estimated horizontal degree from source. The degree values are important to calculate the position from the specific chunk of the room which the antennas are currently looking at to the sources of the signals. These two values are then be sent to the **heatmap_value** module which will compute the specific value used to generate a pixel color in the heatmap as defined in 4.2.4. This value is then sent to an OpenCV routine which processes the heatmap overlay. Using the webcam as a background layer, once all heatmap values are sent to the function, the heatmap overlay will be generated and output to the mini-DisplayPort to be shown on a monitor. The heatmap will be updated at 30Hz refresh rate. The block diagram can be seen in Figure 7.

4 DESIGN TRADE STUDIES

4.1 Antenna Connectivity

Our original design had us creating custom Printed Circuit Boards (PCB) for each antenna in order to get the data needed from each of the antennas. The reason for this was while receivers for WiFi are very common, most of them do not give access to the signal properties for the data sent, and instead only the encoded data. This meant that we did not feel like we would be able to sufficiently apply our beamforming algorithm to the signals. The other alternative was to use Software Defined Radios (SDR) as these are programmable receivers. Unfortunately, since WiFi is in the 2.4 GHz range many of the SDRs that support that frequency were outside of our budget (the most affordable we could find was the HackRF One at $300) meaning while we may be able to buy one or two we again could not purchase enough to perform accurate beamforming. Due to these concerns we designed our own schematic, however, the amount of time it took us to create this schematic and select components was long than anticipated. After receiving feedback from the Professors we decided that this would not be a good approach for our team since it was already taking longer than we had hoped. We also did not feel like we had ample time to both get the devices fabricated and shipped to us, and then still have time to make a revision should it not work correctly the first time. Since RF Circuitry is so specific and detailed the design is already complicated and meant that any revisions would require more time than we could afford.

As a compromise to this we decided that we would instead invest our budget into getting down mixing components so that we could instead get the signals to work with more affordable SDRs such as the RTL-SDR. This down mixing circuitry is necessary since WiFi signals are outside of the frequency range of these budget SDRs. This approach ends up being slightly more costly than if we made a custom design, however it would reduce the amount of labor we would need to spend on assembling a custom PCB and also significantly increases the likelihood of our system working.

4.2 Signal Processing

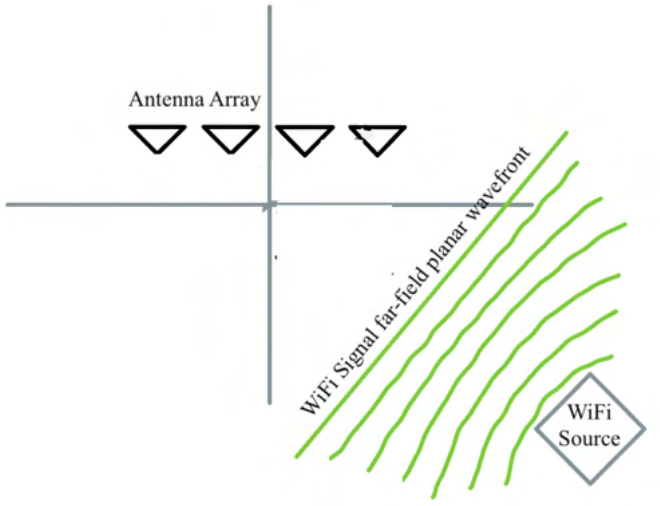

Figure 1: Simple depiction of our 1x4 antenna array with a WiFi-transmitting device shown in the lower-right corner of the image. The WiFi signals emanating from the device appear curved when nearby, but from far away the WiFi signals appear to be plane waves. This is called the far-field assumption, and we will leverage it in the following derivations.

The derivations we present follow, to a degree, the ideas presented in [1] and [2]. However, the former presents a discussion for continuous-time signals, whereas ours are discrete, and the latter does not provide particularly detailed calculations or a beamformer lobe width analysis. Our WiFi source is located at an azimuthal angle ϕ with respect to our antenna array which is roughly shown in Figure 1. When the emitting source is far away, the WiFi signals appear to the receiver as a sequence of plane waves; this is called the far-field assumption.

We want to compute the time delay between when a WiFi signal is received at an arbitrary antenna, located at (x, y) and when it is received at the center of the array. This is depicted in Figure 5.

4.2.1 Single Element Phase Difference

The distance that a WiFi signal must travel between an array element at location (c, r) and the origin can be computed by examining the vector diagram in Figure 5. In particular, the delay $d_{c,r}$ is given by $d_{c,r} = m \cdot w = m_x w_x + m_y w_y$. Since our array is linear, all antenna elements are on the same axis, so we drop the $m_y w_y$ term: $d_{c,r} = m \cdot w = m_x w_x$.

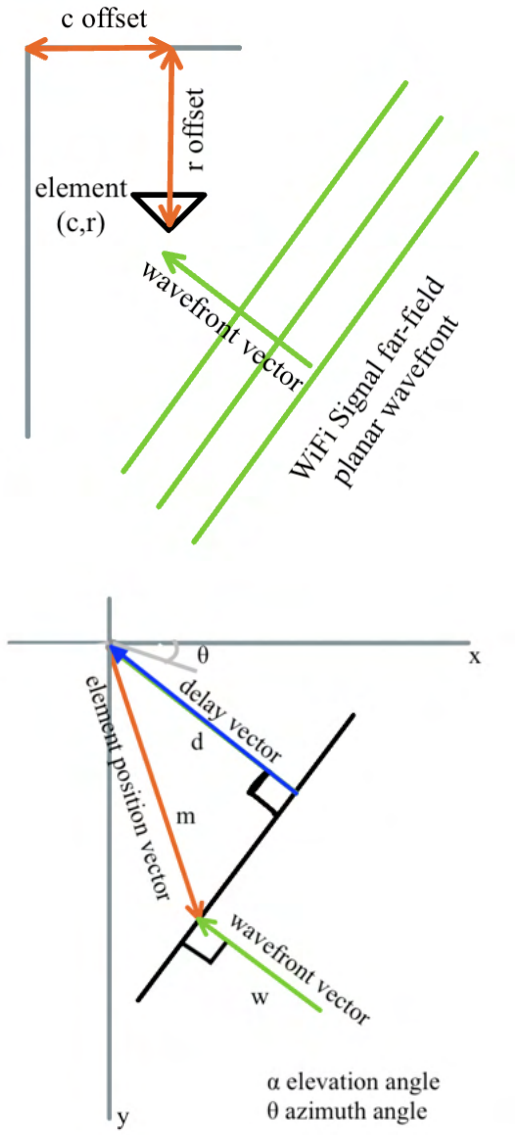

Figure 2: Top: A depiction in two dimensions of how the WiFi plane waves interact with a single antenna under the far-field assumption. Notice that the antenna is located at an x-axis offset c and a y-axis offset r. Bottom: A conversion of the top figure into vector form in order to calculate the delay between the WiFi plane waves at the (c, r) antenna and the origin. The text below elucidates the computation of the delay vector from the wavefront and element position vectors.

Taking w to be a unit vector, we find that the projection onto the x-axis is given by $w_x = \cos(\alpha)\cos(\theta)$, $m_x = c$. Thus, the delay incurred between an array element at location (c, r) is

$$d_{c,r} = m_x w_x = c\cos(\alpha)\cos(\theta). \tag{1}$$

The corresponding phase difference between a signal arriving at the element at (c, r) and the origin is given by $e^{jkd_{c,r}}$ where $k = \frac{2\pi}{\lambda}$ is the wavenumber[1] of the WiFi signal.

Using Equation 1 from above, we can rewrite the phase difference as

$$\Phi_{\alpha,\theta}(c, r) = e^{jk(c\cos(\alpha)\cos(\theta)))} \tag{2}$$

4.2.2 Sum of Antenna Responses

Suppose that a WiFi source transmits a signal x that is sampled by each array element at $x[c, r]$. Then the sum of the responses, including the corresponding phase shift, is a cross correlation between x and the phase difference given by

$$\begin{aligned} X(\alpha, \theta) &= \sum_{c,r} x[c, r]\Phi_{\alpha,\theta}(c, r) \\ &= \sum_{c,r} x[c, r]e^{jk(c\cos(\alpha)\cos(\theta))} \end{aligned}$$

Which, if we let $u = \cos(\alpha)\cos(\theta)$ and drop the extraneous r terms since the array is one-dimensional, is simply the definition of the one-dimensional Fourier Transform up to a constant factor:

$$X(u) = \sum_{c} x[c]e^{jk(cu)} \tag{3}$$

This is a useful construction because, instead of correlating in the time domain, which will scale quadratically in the number of samples, we can leverage optimized implementations of one-dimensional FFTs which will scale $O(n \log n)$ – a significant decrease in computation.

4.2.3 Spatial Localization from the Frequency Domain

We identify whether there is a WiFi-transmitting device located at a given azimuth with respect to our array by performing the following procedure:

1. Compute the one-dimensional Fourier Transform using inputs from our antenna array.
2. Identify the regions in the frequency domain which have the highest intensity values. The centerpoint of a given high-intensity region will have a value, say, u'.
3. Convert u' to angles θ by the relation $\theta = \arccos(u')$ The result is the signal intensity at azimuth θ.

4.2.4 Spatial Resolution

From traditional delay-and-sum beamforming, we know that the sensitivity curve for a 4-element linear array is given by the equation

$$\text{Sensitivity}(\gamma) = \frac{\sin(\frac{4\pi d\sin(\gamma)}{\lambda})}{\sin(\frac{\pi d\sin(\gamma)}{\lambda})} \tag{4}$$

[1]The wavenumber is a measure of the spatial periodicity of a wave; it is the number of oscillations of the signal per unit length.

Where γ is the angle of arrival, d is the spacing between elements on a single axis, and λ is the wavelength of WiFi. For our application, $d = 6.25$ and $\lambda = 12.5$.

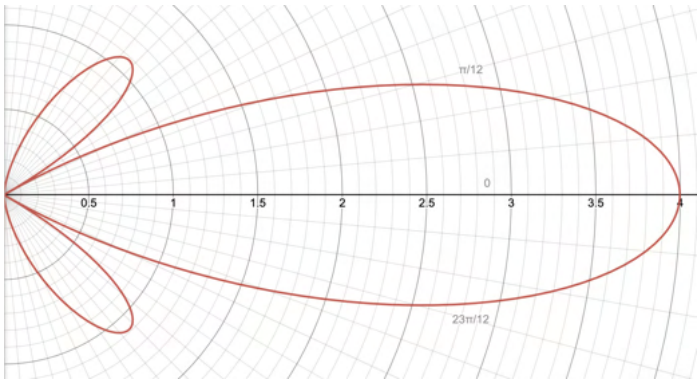

Figure 3: The beamwidth of our 1x4 element antenna array. Notice that the main lobe has a width of approximately $\frac{\pi}{6}$ radians, or approximately ± 15 degrees.

From Figure 3, we can see that our array has a high sensitivity to WiFi transmitting devices within $\frac{\pi}{12}$ radians. This is approximately ± 15 degrees of our field of view, and will satisfy the design constraint of object localization to within 10 feet in a 20x20 foot room.

5 SYSTEM DESCRIPTION

In this section we outline all of the components that make up our system. A system block diagram can be seen in Figure 7.

5.1 Downmixing

For the downmixing portion of our project we were inspired to use the design we found from IanWraith on GitHub. It consists of an ADL5350 Evaluation Kit which is the down mixer itself, this can be found for around $20-$25. In order to downmix the signal it requires a reference signal which is provided by the 1 GHz oscillator on the ADF4350 Evaluation Kit. This can be found for roughly $12. Finally in order to program this oscillator board we need to use SPI so we used an Arduino Uno which we already had to program these register.

5.2 Software Define Radio

For the software defined radio we are using the RTL-SDR. This is an affordable SDR based on an open source design that supports frequency ranges up to 1.7 GHz. This limitation is why we were required to use a down mixer to bring the 2.4 GHz signal into a frequency range that the SDR can receive. It was convenient for us to use this SDR as not only are they affordable, but we were also able to borrow some from Professor Kumar meaning we could spend our budget elsewhere.

5.3 Overlay

The overlay code was written in Python. We used the OpenCV library to grab images from the webcam and then modify them before displaying the image on the computer. To generate a box from the angle of arrival we used some calculations based on the lobe width of the antenna array, and the field of view of the webcam. This allowed us to have a bounding box to move around based on computed angle and highlight the relevant portions of the screen. Unfortunately due to difficulties with the SDRs performance, this had to be scaled back. In the end we ended up with only a box centered around the main lobe which changed color as a result of the computation made on the received signal data.

5.4 Computer

We opted to process the signal data on a laptop (Samsung Notebook 500 running Linux Ubuntu 20.04). The computer has 3 USB ports, one of which received input from the USB hub to which the SDRs were connected and another received webcam input. This computer is one that Enock has extensive experience with so the time to learn and setup the computer was very low. The computer allows for both hardware and software design to occur on the same board at the same time and interface. This is good since it allowed all group members to work on the programming of the signal processing code and heatmap generation code at the same time since they were two different workflows. Furthermore, this computer has 4 cores that were taken advantage of to more efficiently produce the heatmap.

6 TEST & VALIDATION

6.1 Results for Design Specification

In order to test the accuracy of our system we decided to take a series of tests where we measured the received signal power. For these tests we sampled the signal received at 3 different distances from the antennas (5, 10, 15 ft.) as well as at different angles ($0°$, $\pm 15°$, $\pm 30°$). We tested at different distances in order to prove that our design worked for the designated 20x20ft room. We further tested at different angles to see how accurately our system could detect signals within, at the edge, and outside of the lobe-width. In order to calibrate the camera FOV and antennas FOV we implemented crosshairs in the center and at the right and left side of the screen. With known locations marked in our testing room and straight lines (rulers) placed at those locations we were able to align the camera center crosshair and overlay box with objects in the room. In addition, since the camera was in the center of the antenna array there was not much calibration needed since our implementation of the algorithm would place a theorertical (one) big antenna in the exact center of our 4 antennas. In our testings we achieved an 80% accuracy of locating the device which is not the 100% accuracy we had hoped

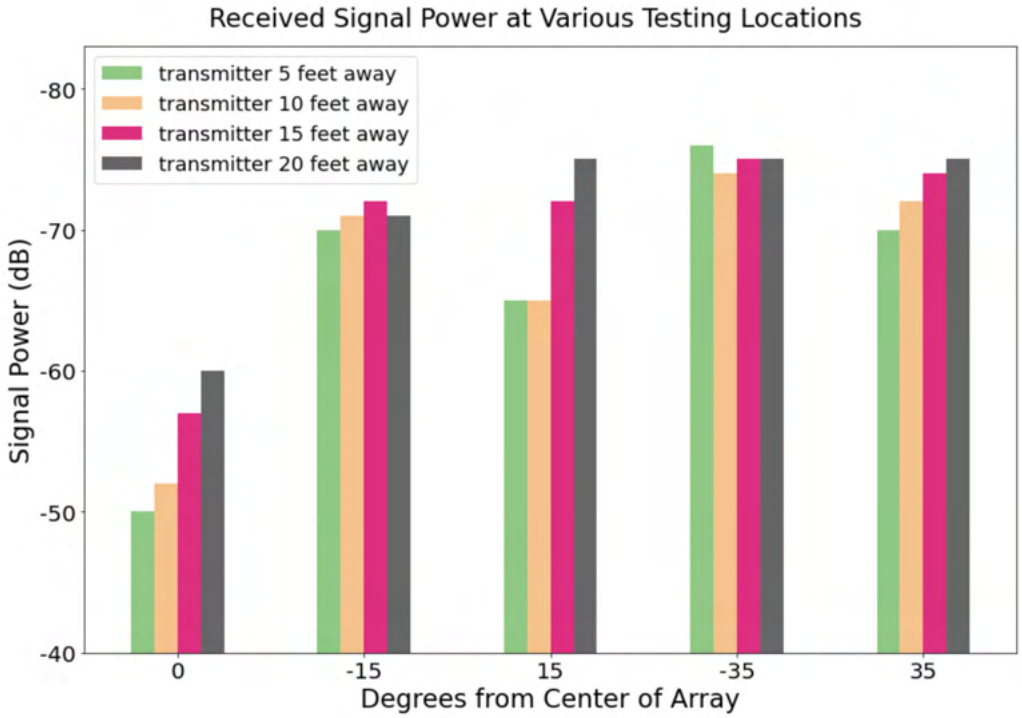

Figure 4: Signal power in decibels averaged over 10 trials at each location Each trial 15 seconds duration (sampled at $2 \cdot 10^6$ samples/sec) System is roughly 80% accurate. 0° is in the middle of the main lobe, so power is high. 15° is right at the edge of the main lobe, so power fluctuates. 35° is outside the main lobe, so power is low.

achieve. This is due to the fact that our SDRs were not as efficient in data capture as we had theoretically assumed. The tests that failed so happened to be at the 15 degree marks which isn't surprising since those are the edges of the lobe-width and thus would be tough to accurately pick up signals at the border. We ran 50 tests for all locations as opposed to our initial 10 tests per location when it came time for final presentation which changed our observed accuracy from 75% to 80%. In terms of the severity of the use cases 80% accuracy is too low for high-stakes sweeps. This could be improved by having better SDRs as well as more SDRs to collect more data. However, the project is mobile and so panning the camera around should solve this accuracy issue since devices at the lobe-width edges will be accounted for when moved closer, in frame, to the center of the antennas/camera. Furthermore, due to the pivot in project from beam steering to just beam forming we could not achieve or test resolution since our device and antenna array only allowed for a vertical slice of the room at the center of the FOV to be scanned. Finally, we achieved a 1Hz refresh rate which matched our initial metric which proved to update fast enough when the device moves.

7 PROJECT MANAGEMENT

7.1 Schedule

Our schedule is shown at the end of the document in Figure 8 Overall our schedule had to be restructured a bit once we switched to SDRs. Our schedule was delayed by the late switch to SDRs from the FPGA but roughly remained similar although the tasks were changed to match the new approach. We also extended our schedule as originally we did not know when the final due dates were so estimated that we had to be finished sooner. This gave us an extra week for administrative tasks and planning.

7.2 Txanton Bejos

Initially Txanton was working on the RF Circuitry. He spent a lot of time of component selection and had most of the schematic before deciding to pivot to using SDRs. After the decision to switch to SDRs he got into contact with Professor Kumar in order to borrow some SDRs so that we could start experimenting with them in order to see what kind of data we would be receiving. He was also responsible for the component selection and ordering. After the switch to SDRs Txanton started working on the Python code to generate the image overlay. He worked with OpenCV and researched the cameras in order to generate a method to draw boxes on different portions of the screen based on angle of arrival. Unfortunately due to issues with the SDRs this had to be scaled back and instead changed to be a static box that changed color based on computed values.

7.3 Vrishab Commuri

Vrishab will be designing and implementing the signal processing algorithm and pipeline. He spent a lot of time

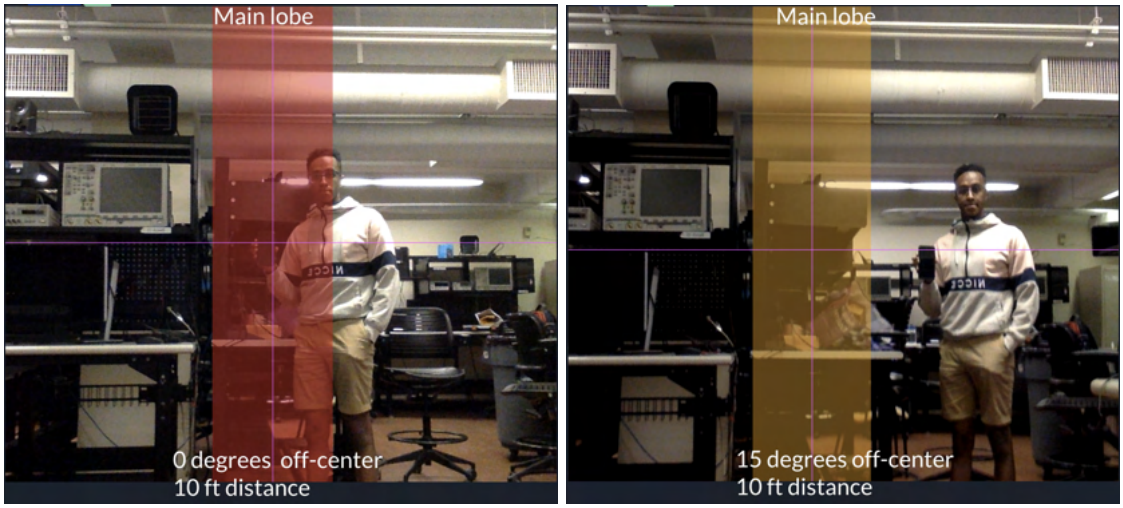

Figure 5: Top: A depiction of the main lobe (rectangle) with high signal power (red color) due to transmitting device in the center of the array. Bottom: A depiction of the main lobe (rectangle) with medium signal power (yellow color) due to transmitting device at 15° off-axis with respect to the array.

designing the frequency-domain correlation algorithm as well as identifying the correct array configuration to achieve the main lobe width desired for the project. He also worked with Txanton to design the signal capture pipeline, which downmixes the signal and converts it from the analog to the digital domain.

Vrishab has met with Dr. Richard Stern twice to discuss the beamformer implementation.

7.4 Enock Maburi

Enock initially worked on the FPGA integration of the project which would have been the final part of the system. He worked on creating the block diagrams of both the FPGA fabric and the ARM core in tandem with Vrishab who helped plan the signal processing dataflow on the FPGA. Specifically he will work on creating the SystemVerilog modules to represent the block diagram modules. He worked in Vivado to create a block diagram of the IP's and Zynq board components, and creating a bitstream to send to the board. Furthermore, he used HLS to create rudimentary tests in C to implement the beamforming algorithm that Vrishab wrote. However, upon pivoting the project he moved to working on putting the hardware (evaluation boards) system together as well as working on the software in Python for the overlay.

7.5 Budget

In Table 1 we have listed all of the components we ordered including shipping costs. We primarily made use of Digikey and Amazon as distributors but did have to order one board off of eBay. After some design revisions and testing we ended up not using the Logic Level Converter or the USB Extension Cables. We also only ended up needing 2 of the Voltage Controlled Oscillators as the down-conversion boards we were using were dual channel and supported two inputs per reference frequency.

7.6 Risk Management

For risk management, we identified two aspects of the project that required risk management:

1. **Modularity.** We ensured that the various system components were modular, so that they could be tested and debugged individually. For the VCOs, we measured the programming signals with an oscilloscope. For the downconverters, we tested each output channel to determine that the output was as expected. For the SDRs, we measured the received signals on standard FM signals to set the RF gain for each.

2. **Signal Processing.** We broke the signal processing pipeline into several software stages that could each be debugged individually. The first of stage was the cross-correlation function, which was tested on synthetic data. The second stage was the beamforming algorithm which, likewise, was tested on synthetic data. The final stage was the signal power measurement and heatmap generation which was tested separately from the rest of the system.

8 ETHICAL ISSUES

One major ethical issue would be the use of a device like this to restrict speech. If the citizens of a country are unhappy and protesting against the government then the government may restrict internet access in the country as can be seen by the 2011 incident in Egypt. While this prevents people from connected to the rest of the Internet, people are still able to create their own peer to peer networks over

WiFi in order to stay in communication with loved ones, and to help get news to reporters who may be able to take it out of the country. With a device like ours where you can find WiFi transmitting devices, the government could systematically sweep areas in order to find people who are using these networks and disable them preventing people from being able to communicate and organize. This definitely a very extreme example and unfortunately there is not a good way for us to prevent this from occurring, but luckily is an unlikely scenario.

9 RELATED WORK

We had added a bibliography section to our document in which we have referenced the relevant research papers we found in order to get an idea of how to apply beamforming to signals such as WiFi.

10 SUMMARY

Our design was not able to fully meet our original design specifications. Our design is only able detect devices in the main lobe which is in a fixed position instead of being steerable. This means that we can only detect devices directly in front of us so it requires the user to re-position the array more often. If we had additional time we could have done more testing. This would have allowed us to tweak the algorithms used in order to try and get better accuracy. We also may have had time to add two more antennas to the array allow us to possibly get better performance since we would be using 6 antennas instead of 4. Overall we were mostly cost constrained as if we were able to use an SDR such as the HackRF One we believe we would not have run into many of the issues that we ended up encountering.

Throughout the course of the semester we learned a lot of valuable lessons. We learned that working with high-frequency signals is a lot more difficult than we expected. As such, it is difficult to design printed circuit boards with high frequency signals unless you already have quite a bit of experience. We also learned that theory and actual results do not perfectly align. We were able to compute a lot of theoretical values for the calculations we were planning to do, however when we actually started working with the Software Define Radios it was a lot more difficult than expected. As such, we only ended up being able to do beam-forming without actually being able to implement the beam-steering we initially planned on doing. Regarding the engineering process as a whole we also learned that it is beneficial to do more risk management and planning early on. If we had not focused so long on our FPGA + PCB design then we could have spent more time working with the SDRs. Similarly we learned that paying for more expensive components can be beneficial as the cheaper components we used really struggled to perform how we wanted them to. If we had spent more time in the planning phase searching for different SDRs we may have been able to alleviate some of those issues meaning we might have been able to fully deliver on our original design.

Glossary of Acronyms

- DSP - Digital Signal Processing
- FM - Frequency Modulation
- FOV - Field of View
- FPGA - Field Programmable Gate Array
- IF - Intermittent Frequency
- IP - Intellectual Property
- LLC - Logic Level Converter
- PCB - Printed Circuit Board
- RF - Radio Frequency
- SDR – Software Define Radio
- VCO – Voltage Controlled Oscillator

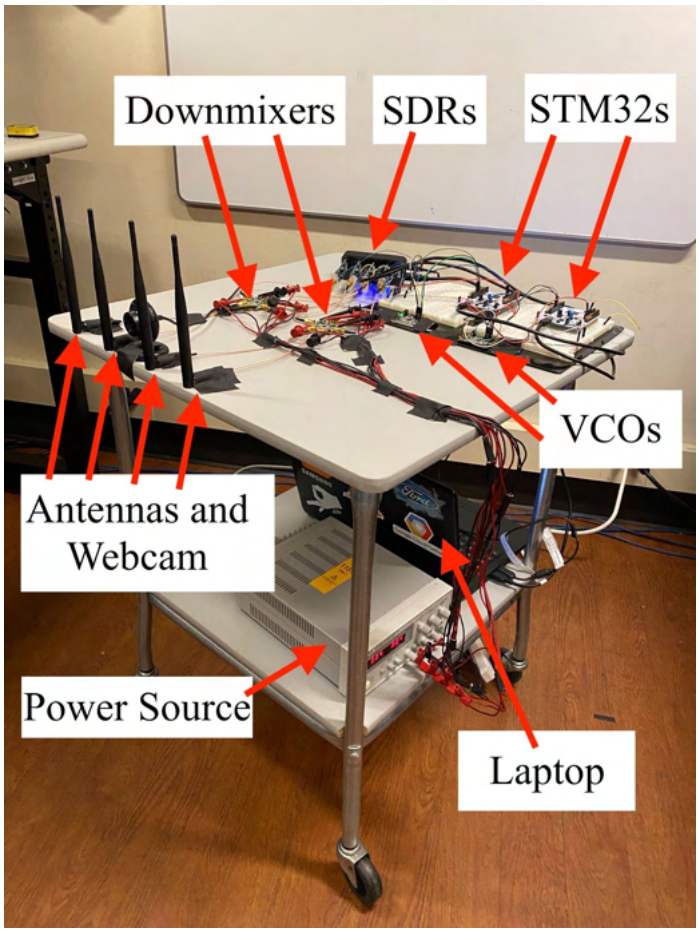

Figure 6: A labelled diagram of our completed project

Table 1: Bill of materials (Cost's include Shipping)

Description	Model #	Manufacturer	Quantity	Cost @	Total
RTL-SDR	RTL2832U	RTLSDR	4		Borrowed
Microcontroller	Nucleo STM32F401RE	ST Microelectronics	2		Borrowed
Webcam	USB 2.0 HD Pro	Stopmotion Explosion	1		Owned
Downconverter	DC1710A-D	Analog Devices	2	$128.49	$256.99
VCO	ADF4350	Analog Devices	4	$14.99	$59.96
SMA Cable	5 Pack SMA Cable	SDTC Tech	2	$12.99	$25.98
RP SMA to SMA	2-Pack RF Coax Adapter	DGZZI	2	$5.50	$11.00
Logic Level Converter	BOB-12009	SparkFun	4	$4.69	$18.79
USB Hub	4-Port USB 3.0 Hub	Sabrent	1	$17.98	$17.98
USB Extension	2 Pack USB 3.0 Extension Cable	NIMASO	2	$9.99	$19.98
					$424.66

Bibliography

[1] S. Patole and M. Torlak, "Two Dimensional Array Imaging With Beam Steered Data," IEEE Trans. on Image Process., vol. 22, no. 12, pp. 5181–5189, Dec. 2013, doi: 10.1109/tip.2013.2282115.

[2] D. Huang, R. Nandakumar, and S. Gollakota, "Feasibility and limits of wi-fi imaging," presented at the SenSys '14: The 12th ACM Conference on Embedded Network Sensor Systems, Nov. 2014, doi: 10.1145/2668332.2668344.

[3] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti, "SpotFi," presented at the SIGCOMM '15: ACM SIGCOMM 2015 Conference, Aug. 2015, doi: 10.1145/2785956.2787487.

[4] T. Kodera, "Adaptive antenna system by ESP32-PICO-D4 and its application to web radio system," HardwareX, vol. 3, pp. 91–99, Apr. 2018, doi: 10.1016/j.ohx.2018.03.001.

[5] "Effect of Positive Pressure Ventilation on a Room Fire". National Institute of Standards and Technology. 1 March 2005. Retrieved 21 February 2021. https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=861347

[6] J. Spjut et al., "Latency of 30 ms Benefits First Person Targeting Tasks More Than Refresh Rate Above 60 Hz," presented at the SIGGRAPH Asia 2019 Technical Briefs, 2019, doi: 10.1145/3355088.3365170.

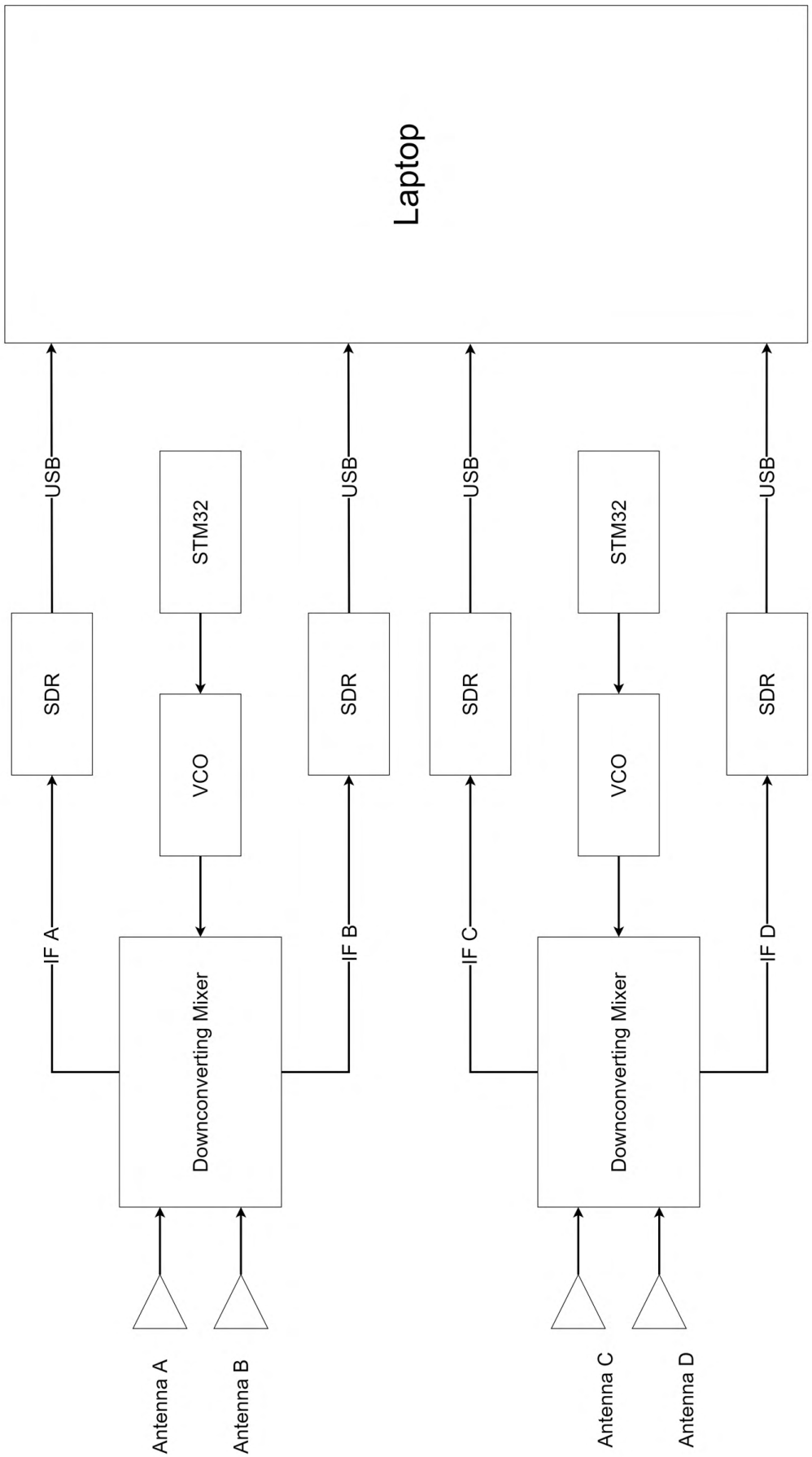

Figure 7: A full-page version of the same system block diagram as depicted earlier.

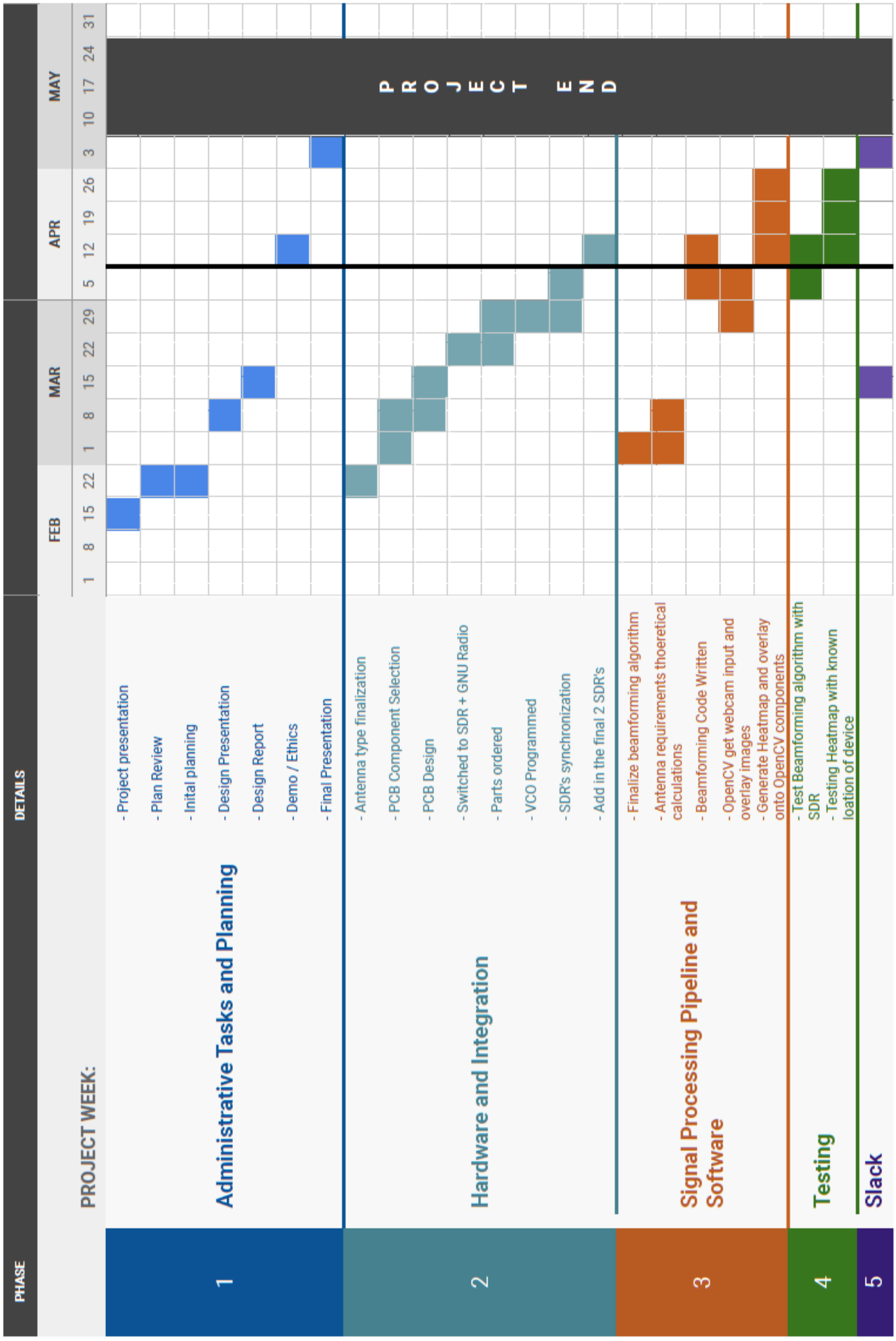

Figure 8: Gantt Chart