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 an 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 5 feet in a 20x20 foot room, we need an approximate localization resolution of $\arctan(5/20) \approx 15$ degrees squared. 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 1 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 5 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 5 feet in a 20x20 foot room.
- Antenna array main lobe width of 15 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 will have 16 antennas in a 4x4 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 will downconvert (downmix) the channel 6 WiFi signals to baseband using an off-the-shelf downconverter module attached to each of our 16 antennas. The downconverted signals will then be 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 FPGA through the GPIO pins.

3.2 FPGA

The array of 16 signals coming from the RTL-SDRs will be sent into the GPIO pins of the Zynq Ultra96 v2 FPGA. These signals will be sent to the **frequency_domain_correlation** module which will handle the correlation of signals in the frequency domain. These signals will be sent to the **2D_FFT** module which will apply a 2D FFT to the signals to produce a square matrix output of signal intensities. This matrix will be sent to the **intensity_evaluation** module which handles the calculation of the critical values used to produce the heatmap chunk values. The first value is the maximum intensity value which is the maximum signal intensity found in matrix produced by the 2D FFT. The second value is the azimuth value which is the horizontal degree from source. The third value is the elevation value which is the vertical degree from source. These 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 three values will 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 will be sent to the ARM Core of the FPGA which we will run an image of Linux in order to use Python. We will pass the webcam data input via USB into the ARM core as a background layer for 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 6.

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 Professor's 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.

We have attached a copy of our schematic before we switched to SDRs in Figure 7 at the end of the document.

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.

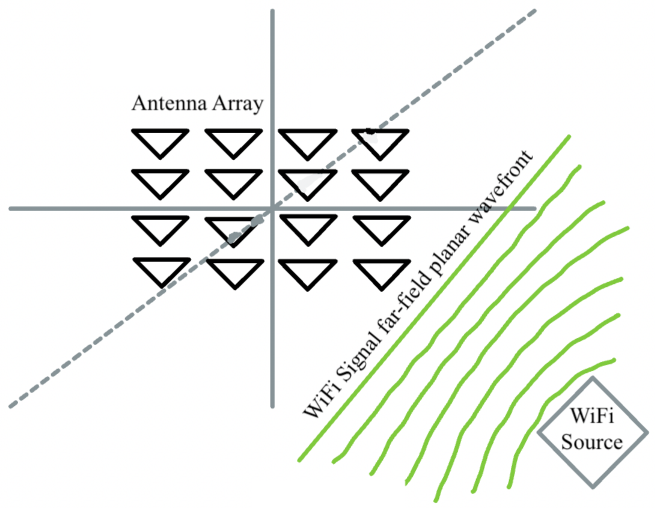


Figure 1: Simple depiction of our 4x4 antenna array with a WiFi-transmitting device shown in the lower-right corner of the image. The WiFi signals emanating from the device appear curved when nearby, but from far away the WiFi signals appear to be plane waves. This is called the far-field assumption, and we will leverage it in the following derivations.

4.2 Signal Processing

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 and elevation angle θ 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 2.

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 ??. In particular, the delay $d_{c,r}$ is given by $d_{c,r} = m \cdot w = m_x w_x + m_y w_y$.

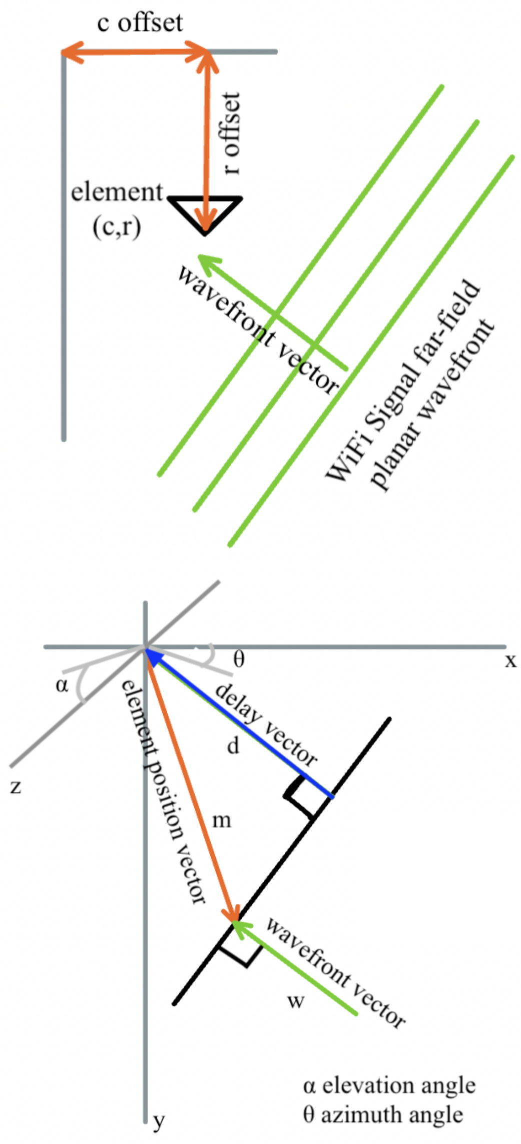


Figure 2: Top: A depiction in two dimensions of how the WiFi plane waves interact with a single antenna under the far-field assumption. Notice that the antenna is located at an x-axis offset c and a y-axis offset r. Bottom: A conversion of the top figure into vector form in order to calculate the delay 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 $w_x = \cos(\alpha)\cos(\theta)$, $m_x = c$ and the projection onto the y-axis is given by $w_x = \sin(\alpha)$, $m_y = r$. Thus, the delay incurred between an array element at location (c,r) is

$$d_{c,r} = m_x w_x + m_y w_y = c\cos(\alpha)\cos(\theta) + r\sin(\alpha). \quad (1)$$

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

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

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

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

Which, if we let $u = \cos(\alpha)\cos(\theta)$ and $v = \sin(\alpha)$, is the definition of the two-dimensional Fourier Transform up to a constant factor:

$$X(u,v) = \sum_{c,r} x[c,r]e^{jk(cu+rv)} \quad (3)$$

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 two-dimensional FFTs which will scale $O(n\log n)$ – a significant decrease in computation.

4.2.3 Spatial Localization from the Frequency Domain

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

1. Compute the two-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', v').

3. Convert u' and v' to angles α and θ by the relations $\alpha = \arcsin(v')$ and $\theta = \arccos(\frac{u'}{\cos(\arcsin(v'))})$. The result is the signal intensity at elevation α and azimuth θ.

[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.

4.2.4 Spatial Resolution

We want to determine the elevation and azimuthal sensitivity of our antenna design. We will examine the response for the case when the input $x[c, r] = 1$. That is:

$$\begin{aligned} X(\alpha, \theta) &= \sum_{c,r} \Phi_{\alpha,\theta}(c, r) \\ &= \sum_{c,r} e^{jk(c\cos(\alpha)\cos(\theta)+r\sin(\alpha))} \\ &= \sum_{c,r} e^{jkc\cos(\alpha)\cos(\theta)} e^{jkr\sin(\alpha)} \\ &= \sum_{c} e^{jkc\cos(\alpha)\cos(\theta)} \sum_{r} e^{jkr\sin(\alpha)} \end{aligned}$$

So the beamformer sensitivity is independent across azimuth and elevation; and since we are using a 4x4 array, the elevation sensitivity will be equal to the azimuth sensitivity. From traditional delay-and-sum beamforming, we know that the sensitivity curve for a 4-element linear array is given by the equation

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

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 $\lambda = 12.5$.

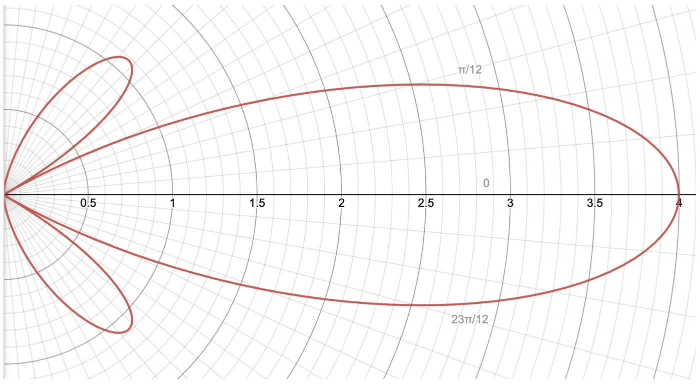


Figure 3: The beamwidth of our 4x4 element antenna array. Notice that the main lobe has a width of approximately $\frac{\pi}{12}$ radians, or approximately 15 degrees.

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 5 feet in a 20x20 foot room.

5 SYSTEM DESCRIPTION

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.

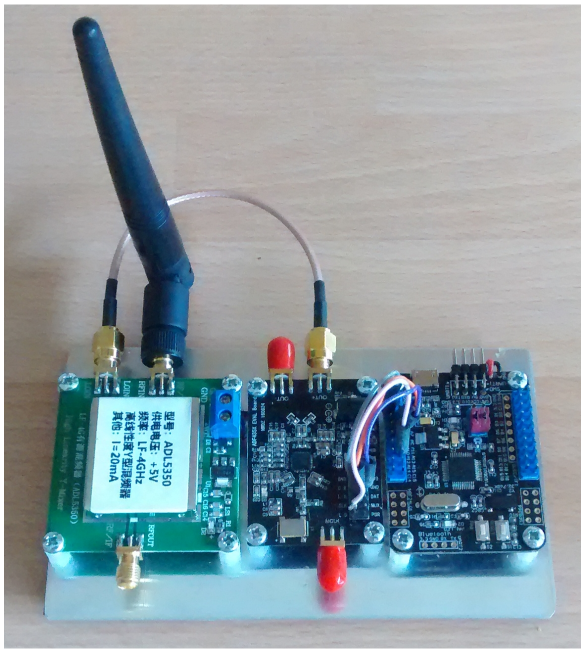


Figure 4: The downmixing components. Source: https://github.com/IanWraith/24DownConvert/raw /master/images/full_unit.jpg

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 FPGA

We chose the the Zynq Ultra96 v2 board because it is an MPSoC with enough GPIO pins to input all of the antenna signals. This board is one that Enock has extensive experience with so the time to learn and setup the board will be very low. The MPSoC allows for both hardware and software design to occur on the same board at the same time and interface. This is good since it will allow for Enock to work on the FPGA fabric and Txanton to work on the programming of the ARM Core/heatmap generation at the same time since they are two different workflows. Furthermore, since this is an MPSoC it has two cores that can be taken advantage of to more efficiently produce the heatmap.

5.4 Backup Plan

As of now everything seems to be on track, however in the case of a major hurdle where we are not able to progress we have an additional backup plan. We won't be able to get as accurate of a reading and it will not resistant to obstacles since it requires well formed data to be received but our backup plan is to use the ESP8266 line of microcontrollers which have WiFi radios embedded in them. This will allow us to receive data and we can then attempt to use RSSI as a means of beamforming since we would be unable to get raw signal data. This is not ideal, but should be workable, hence why it is our backup strategy.

This section can use 1.5-3.5 pages. Most groups will use between 2 and 3 pages.

6 PROJECT MANAGEMENT

6.1 Schedule

Our schedule is shown at the end of the document in Figure 6 Overall our schedule has not changed much. We have partially used up some of our slack time due to pivoting from custom PCB design to using SDRs meaning that everything is slightly later however we luckily had space to account for this in our original plan.

6.2 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. Since the hardware component is critical to this project he had not worked as much on the beamforming implantation however now that most of the component selection is completed he will be assisting there as well.

6.3 Vrishab Commuri

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

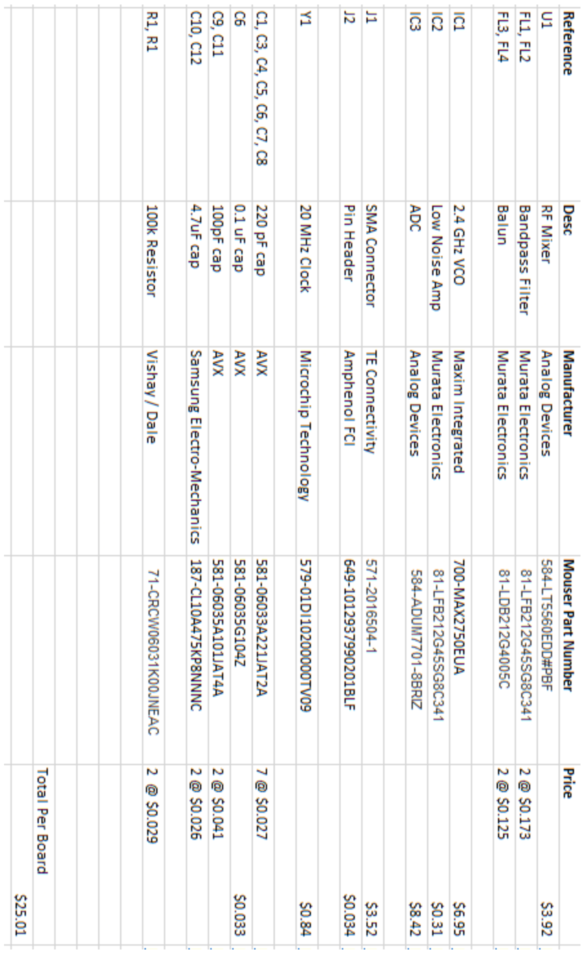
6.4 Enock Maburi

Enock will be working on the FPGA integration of the project which is 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 and Txanton who helped with the ARM Core setup and Python programming. Specifically he will work on creating the SystemVerilog modules to represent the block diagram modules. This will be done in Vivado which will help in writing/editing the code, creating a block diagram of the IP's and Zynq board components, and creating a bitstream to send to the board. Furthermore, he may use the HLS software which will allow for algorithms to be written in C, optimized, and converted to SystemVerilog to be ran on the board. This will allow for suite tests to be written in C for debugging purposes which will show up on a terminal connected to the board. Furthermore, he will work on the programming of the heatmap generation on the ARM core as well as setting up the Linux image on the board to run the python program.

6.5 Budget

Component	Quantity	Price
Ultra96 Zynq MPSoC	1	Borrowed
Logitech c270	1	$27.47
RTL-SDR	16	Borrowed
ADL5350 Evaluation Kit	16	$320
ADF4350 Evaluation Kit	16	$192
Arduino	16	Borrowed
Antenna	16	$64

Now that we are using SDRs here is our budget. For our initial design with a custom PCB we created a bill of materials for what we need for each antenna as shown here. We did not end up purchasing these however they were required for our initial design.



Reference	Desc	Manufacturer	Mouser Part Number	Price	
U1	RF Mixer	Analog Devices	584-LT5560EDD#PBF		$3.92
FL1, FL2	Bandpass Filter	Murata Electronics	81-LFB212G45SG8C341	2 @ $0.173	
FL3, FL4	Balun	Murata Electronics	81-LDB212G4005C	2 @ $0.125	
IC1	2.4 GHz VCO	Maxim Integrated	700-MAX2750EUA		$6.95
IC2	Low Noise Amp	Murata Electronics	81-LFB212G45SG8C341		$0.31
IC3	ADC	Analog Devices	584-ADUM7701-8BRIZ		$8.42
J1	SMA Connector	TE Connectivity	571-2016504-1		$3.52
J2	Pin Header	Amphenol FCI	649-1012937990201BLF		$0.034
Y1	20 MHz Clock	Microchip Technology	579-01DI10200000TV09		$0.84
C1, C3, C4, C5, C6, C7, C8	220 pF cap	AVX	581-06033A221JAT2A	7 @ $0.027	
C6	0.1 uF cap	AVX	581-06035G104Z		$0.033
C9, C11	100pF cap	AVX	581-06035A101JAT4A	2 @ $0.041	
C10, C12	4.7uF cap	Samsung Electro-Mechanics	187-CL10A475KP8NNNC	2 @ $0.026	
R1, R1	100k Resistor	Vishay / Dale	71-CRCW06031K00JNEAC	2 @ $0.029	
				Total Per Board	
					$25.01

Figure 5: Initial Bill of Materials

7 Risk Management

7.1 Signal Capture

Our initial approach to signal capture – the portion of our design involving reception of the analog WiFi signal and its conversion to discrete-time – we planned to fabricate the downmixing and A/D conversion hardware on a custom PCB. However, we found that a custom system would be too complicated, so we opted to utilize off-the-shelf hardware to perform these tasks. We will now be using a commodity downmixer as well as RTL-SDRs to perform the A/D conversion. Since these parts are all modular, we can simply replace each in case of a failure. Each of the RTL-SDRs are modular, and are provisioned with their own software that can be used for analysis and debugging. We will use this software, should the need arise, to diagnose problems in our implementation prior to integration with the FPGA.

7.2 FPGA

In the case that we cannot implement our custom beamforming algorithm on the FPGA, we can use established IP blocks to perform the beamforming. In addition, some modules that may not perform efficiently on the FPGA can be offloaded to the ARM core for processing in a higher-level language. Though this will come at the cost of a reduction in efficiency. If the FPGA cannot meet the target refresh rate of 30Hz, we will be forced to lower the target refresh rate; since the FPGA is quite fast, we don't foresee a considerable reduction past 30Hz.

7.3 Signal Processing

If the signal processing algorithm does not work as intended, either due to implementation complexity or more fundamental issues, we can always implement a traditional delay-and-sum beamforming algorithm using four separate beamformers. The downside of this approach is that the processing time will scale quadratically, rather than $O(n \log n)$ in our implementation, so our processing time will be correspondingly slower.

This full column should describe how you handled your project risk from the standpoints of design, schedule, and resources (budget and personnel) and identify how you mitigated against the risk that cropped up through the semester (e.g., fallback designs, risk reduction measures).

Focus on the primary risk elements, and use about a full column so that there is no more than one page for the entire Project Management section.

8 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.

9 SUMMARY

Overall we believe the our system will be able to meet our design specifications. Unfortunately do to the fact that we are using affordable SDRs we are limited in the bandwidth we can receive meaning we will not be able to view the full bandwidth of WiFi signals that are transmitted. This is affect our performance since it means we may not always retrieve accurate readings and the received signals will have less power. If we had more time then we definitely would have liked to go with the custom PCB design as we could have addressed this by choosing specific components since SDRs are usually made to be very general. We will try our best to write a custom algorithms for the FPGA, however, with time it may be the case that we will just need to copy the algorithms from research papers. It is possible that the 30Hz refresh rate may not be achieved due to the time in which all the signals will be passed and processed in the FPGA fabric and then the ARM Core but our lower bound of refresh rate will be 1Hz since that was our previous rate.

9.1 Lessons Learned

One recommendation we would have to other groups addressing this is to start as early as possible on the PCB design. Due to some delays in our design process we did not have our schematic completed until roughly the middle of the semester. This meant that we did not have sufficient time to order to PCB's and then still have time to update and revise them in the future should any issues arise. Another recommendation when working with antennas is to make sure you can see the signals being sent to the antennas, that multiple ones can work in sync, and that you can process the data digitally.

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.

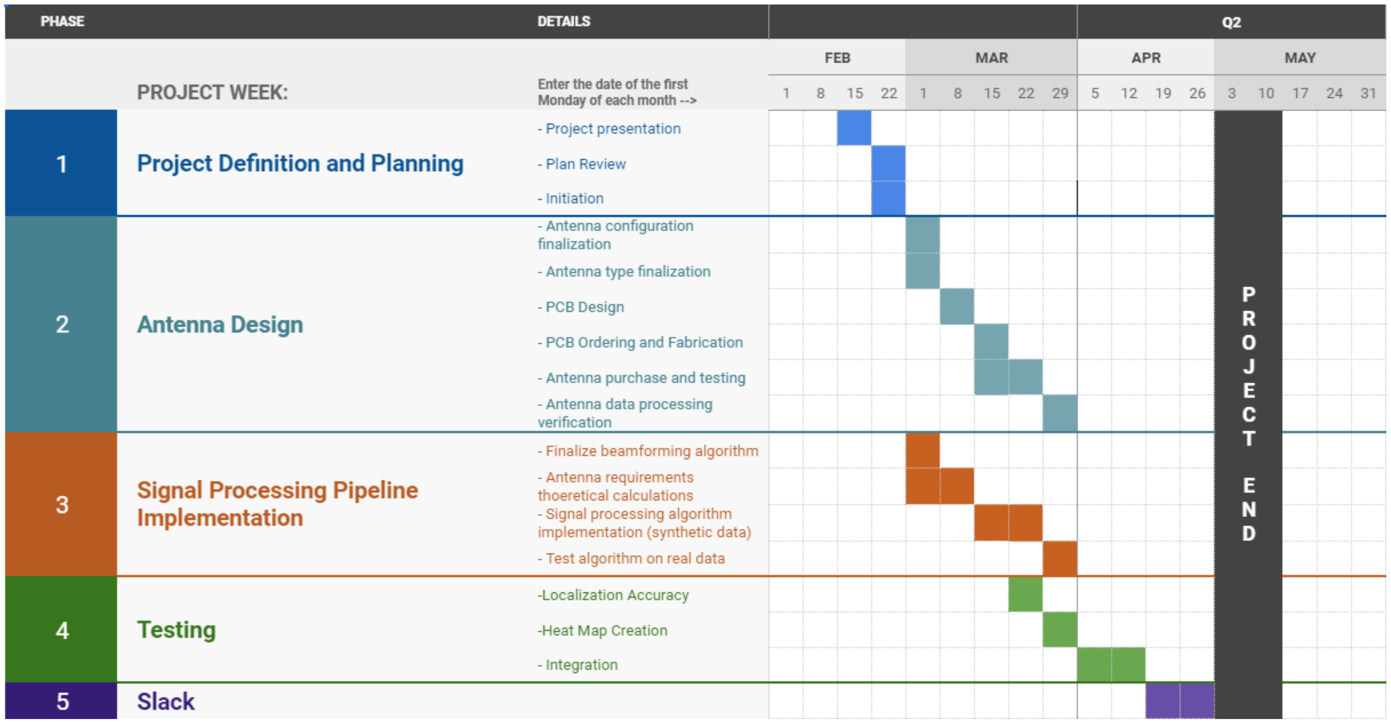
[2] D. Huang, R. Nandakumar, and S. Gollakota, “Feasibility and limits of wi-fi imaging,” presented at the SenSys ’14: The 12th ACM Conference on Embedded Network Sensor Systems, Nov. 2014, doi: 10.1145/2668332.2668344.

[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.



PHASE		DETAILS										Q2								
			FEB				MAR					APR				MAY				
	PROJECT WEEK:	Enter the date of the first Monday of each month -->	1	8	15	22	1	8	15	22	29	5	12	19	26	3	10	17	24	31
1	Project Definition and Planning	- Project presentation			■											PROJECT END	■			
		- Plan Review				■										■	■			
		- Initiation				■										■	■			
2	Antenna Design	- Antenna configuration finalization					■									■	■			
		- Antenna type finalization					■									■	■			
		- PCB Design						■								■	■			
		- PCB Ordering and Fabrication							■							■	■			
		- Antenna purchase and testing							■	■						■	■			
		- Antenna data processing verification									■					■	■			
3	Signal Processing Pipeline Implementation	- Finalize beamforming algorithm					■									■	■			
		- Antenna requirements theoretical calculations					■	■								■	■			
		- Signal processing algorithm implementation (synthetic data)							■	■						■	■			
		- Test algorithm on real data									■					■	■			
4	Testing	-Localization Accuracy								■						■	■			
		-Heat Map Creation									■					■	■			
		- Integration										■	■			■	■			
5	Slack													■	■	■	■			

Figure 6: Gantt Chart

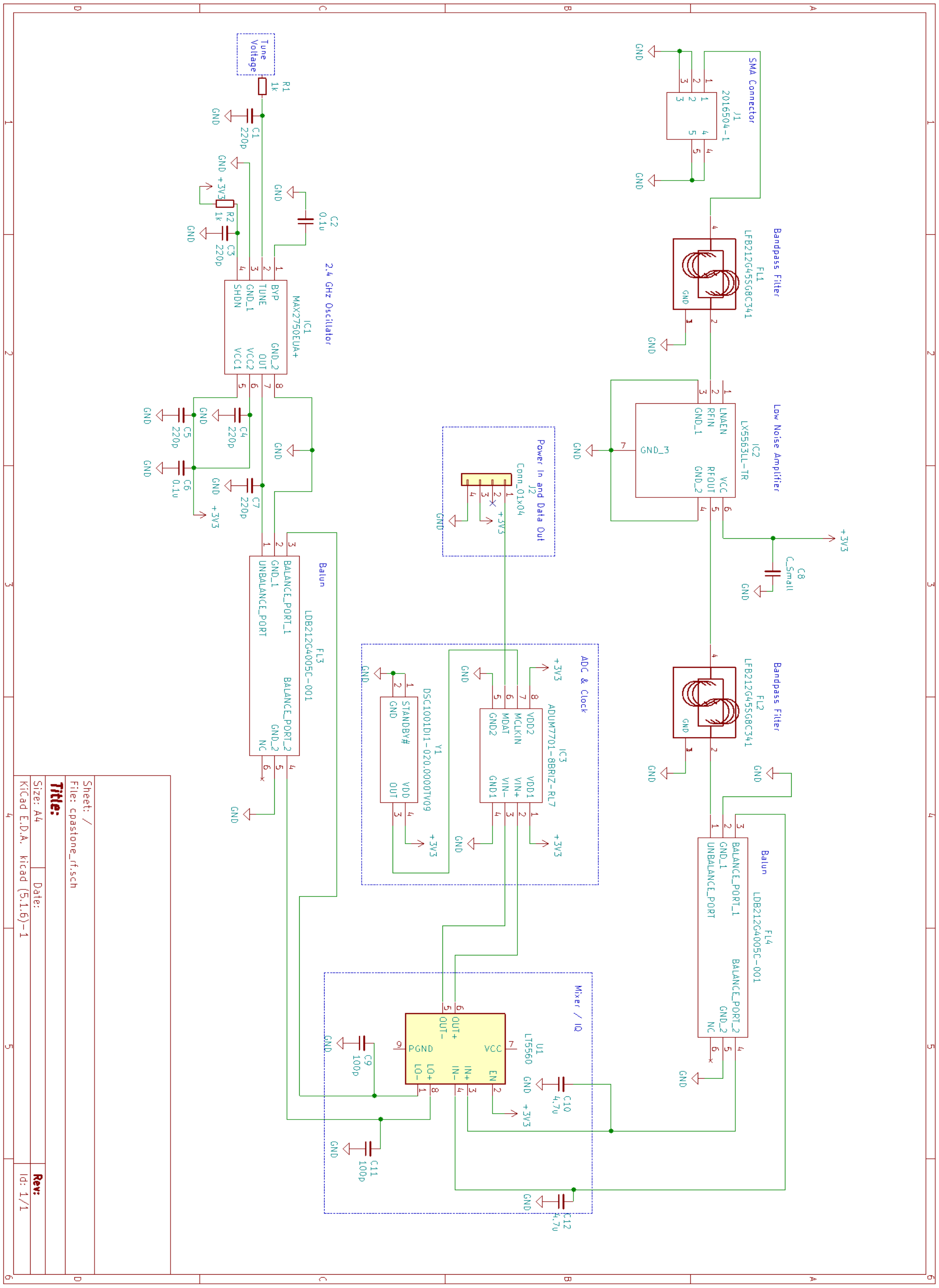


Figure 7: The schematic we ended up creating. The areas in blue boxes were not fully finished by the time we changed to using SDRs

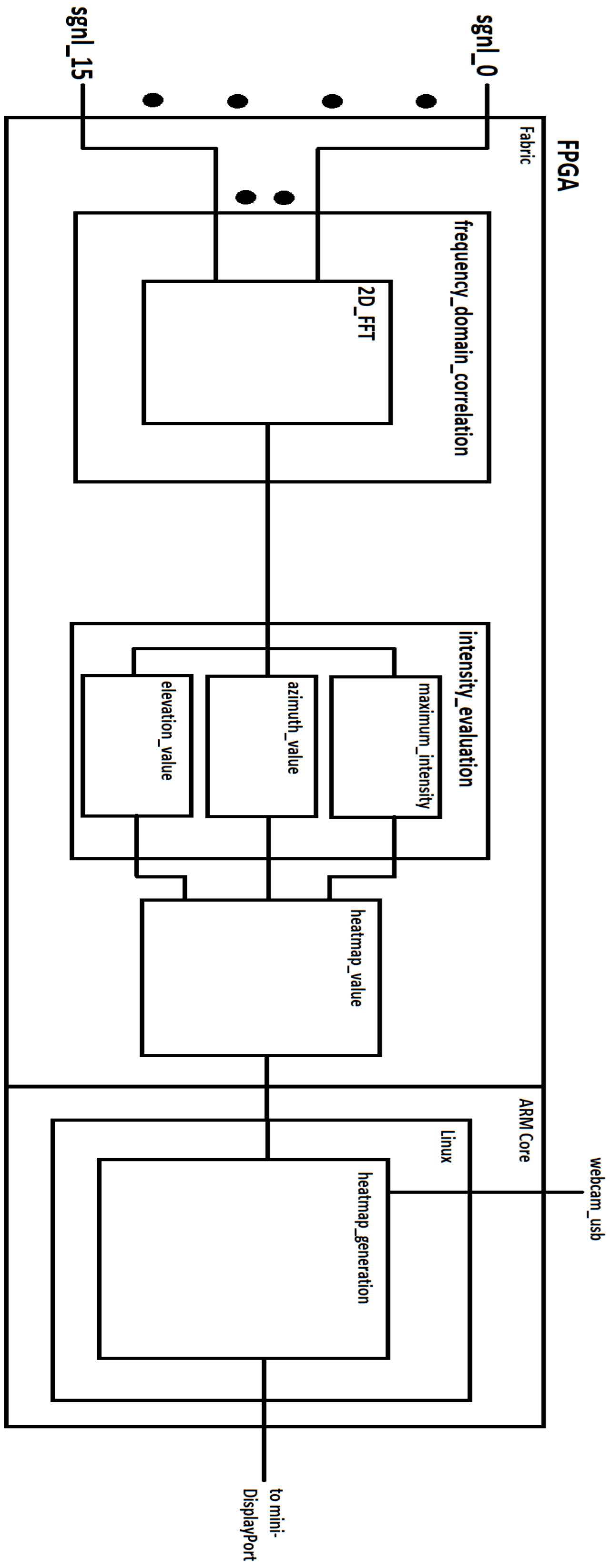


Figure 8: FPGA block diagrams which showcase the movement of data between the SDRs, FPGA Fabric, and ARM Core