Indoor Localization Systems using Channel State Information

ECE496Y Design Project Course Final Report

Mohaimen Khan - mohaimen.khan@mail.utoronto.ca
Shrey Parikh - shrey.parikh@mail.utoronto.ca
Matthew Perry - m.perry@mail.utoronto.ca

Team number: 2019100
Supervisor: Shahrokh Valaee
Administrator: Hamid Timorabadi

Submission date: March 24, 2020

Report Attribution Table

Section	Student Initials		
	1. MK	2. SP	3. MP
Executive Summary		RS, RD, ET	
Group Highlights	ET	RD, MR, ET	ET
Acknowledgements		RD	
Background and Motivation	ET	RS, RD, ET	
Project Goal	RS, RD	MR, ET	
Project Requirements	RS, RD	MR, ET	
System-Level Overview	RS, RD, MR	ET	
Module-Level Descriptions	RS, RD, MR, ET	RS, RD, MR, ET	RS, ET
Assessment of Proposed Design	RS, RD, MR, ET		
Validation Matrix	RS,	RS, RD, ET	RS, ET
Module-Level Test Results	RS, RD, MR	RS, RD, MR, ET	
Testing and validation	RS, RD, MR	RS, RD, MR, ET	RS, MR, ET
Summary and Conclusions		RD, MR, ET	MR, ET
Appendix A – Gantt Chart History	RD	MR, ET	MR, RD
Appendix B – Financial Plan		MR, ET	RD
Appendix C – Validation and Acceptance Tests		ET	RD
Appendix D – Raspberry pi	RS	RD, RS, ET	RS
Appendix E – Set up of Receivers and Transmitters in Lab	RS, MR	RD, ET	
Appendix F – Data from Access points		RD, ET	
Appendix G – GNU Radio CSI Filtering		RD, ET	
Appendix E – Setup of transmitters and receivers in the lab		RD, ET	
Appendix H – Position Estimation			RD, MR, ET, RS
Appendix I – Express.js Server	RS, RD, MR, ET		
Appendix J – React.js Server on Frontend	RS, RD, MR, ET		
Appendix K – Assessment of proposed design	RS, RD, MR, ET		
Appendix L – Transmitting from Laptop	MR, RS, RD	ET	
All	FP, CM	FP, CM	FP, CM

Abbreviation Codes:
Fill in abbreviations for roles for each of the required content elements. You do not have to fill in every cell. The "**All**" row refers to the complete document and should indicate who was responsible for the final compilation and final read through of the completed document.

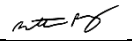
RS – responsible for research of information
RD – wrote the first draft
MR – responsible for major revision
ET – edited for grammar, spelling, and expression
OR – other
"All" row abbreviations:
FP – final read through of complete document for flow and consistency

CM – responsible for compiling the elements into the complete document
OR - other

If you put OR (other) in a cell please put it in as OR1, OR2, etc. Explain briefly below the role referred to:
OR1: enter brief description here
OR2: enter brief description here

Signatures

By signing below, you verify that you have read the attribution table and agree that it accurately reflects your contribution to this document.

Name	**Shrey Parikh**	Signature		Date:	**March 20, 2020**
Name	**Mohaimen Khan**	Signature		Date:	**March 20, 2020**
Name	**Matthew Perry**	Signature		Date:	**March 20, 2020**

Voluntary Document Release Consent Form

To all ECE496 students:

To better help future students, we would like to provide examples that are drawn from excerpts of past student reports. The examples will be used to illustrate general communication principles as well as how the document guidelines can be applied to a variety of categories of design projects (e.g. electronics, computer, software, networking, research).

Any material chosen for the examples will be altered so that all names are removed. In addition, where possible, much of the technical details will also be removed so that the structure or presentation style are highlighted rather than the original technical content. These examples will be made available to students on the course website, and in general may be accessible by the public. The original reports will not be released but will be accessible only to the course instructors and administrative staff.

Participation is completely voluntary, and students may refuse to participate or may withdraw their permission at any time. Reports will only be used with the signed consent of all team members. Participating will have no influence on the grading of your work and there is no penalty for not taking part.

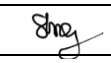
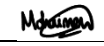
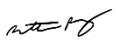
If your group agrees to take part, please have all members sign the bottom of this form. The original completed and signed form should be included in the hardcopies of the final report.

Sincerely,
Khoman Phang
Phil Anderson
ECE496Y Course Coordinators

Consent Statement

We verify that we have read the above letter and are giving permission for the ECE496 course coordinator to use our reports as outlined above.

Team #: **2019100** Project Title: **Deep Learning in CSI Based Localization**
Supervisor: **Shahrokh Valaee** Administrator: **Hamid Timorabadi**

Name	**Shrey Parikh**	Signature		Date:	**March 20, 2020**
Name	**Mohaimen Khan**	Signature		Date:	**March 20, 2020**
Name	**Matthew Perry**	Signature		Date:	**March 20, 2020**

Executive Summary

As the world becomes more connected through the internet and the ubiquity of smartphones reaches even the most remote communities, there is a need for a robust indoor location system that can use these devices, along with the internet, to solve indoor positioning. The need for indoor positioning arises from indoor navigation and location. Indoor positioning can be extensively used in hospitals, manufacturing & robotics, and has use cases in augmented reality. The goal of the project is to develop an indoor positioning system to locate an object based on the information collected from wireless access points (APs).

This final report will start by stating the goal of our capstone project. From our goal, we list the various requirements for a valid, functional end-to-end design to achieve our goal. For example, our solution must be able to automatically collect data and send it over to a server for analysis and to estimate the position of the object. Further we will highlight the constraints for our design. Constraints can be environmental, such as the dimensions of the room or intentional such as the minimum number of APs. These requirements and constraints guided our design choices. We will discuss the objectives for our final solution; solutions incorporating machine learning techniques for signal noise reduction and researching methods to increase the number of antennas per AP are desired but not required.

We have divided our system into three components – data collection, data analysis and data visualization. Each component has been separated into modules, these modules describe the inputs and outputs between each component. Our modules include the data collection from transmitter, central processing server, data publishing method to the central processing server, data analysis module within central processing server and data visualization module.

To summarize the flow of our design, the data collection component firstly collects Channel State Information (CSI) data from the object via the antennas of the receivers. Each receiver is to a computer. Once the data is collected, the computers can calculate the Angle of Arrival (AoA), collate CSI data, and transmit the data over to a Central Processing Server (CPS) using a publisher-subscriber architecture. The data analysis component will use the data in the CPS and further clean this data and append useful metadata. The data visualization component will use the current and past estimate of the object's location (records) to visualize the objects location through API calls to the processing server. A MongoDB-Express-React-Node JS (MERN) stack will be used for the full stack server and APIs to develop a monolithic application.

In the testing and verification chapter, we will discuss the various tests we will use to validate our design and confirm our goal is met. These validation tests are guided by our project requirements; each validation test is directly derived from a requirement and a test is developed to validate the requirement. Each test has referenced documentation to provide proof of meeting each requirement. We have passed all tests except one; we did not complete the test of the number of packets received by each AP (R7) due to our laboratory closing.

Lastly, we will conclude this report by recoding the achievements of our design. For example, we were able to develop a scalable end-to-end system which could digest CSI data from a transmitter and visualize the location of a transmitter. We will also discuss drawbacks of our design and how our design could work in the real world. One such drawback is that our design relies on modern software – which may not interface with older embedded systems. We discuss the future work needed to improve our design and build on top of our design to realize a scalable, end-to-end indoor localization system - for example real time location remains to be incorporated in our design.

Group Highlights and Individual Contributions

Group Contributions

- CSI Data collection from multiple access points (APs)
    - Setup LimeSDR boards with laptop computers.
    - Added to existing work to store relevant information - Angle of Arrival (AoA) CSI Data, etc.
    - Collected test data between a transmitter and receiver (Access Point), with the AP in 4 different locations.
    - Processed CSI data and added unique identifiers, sent data to a central database in MongoDB with an enforced schema by interacting with the REST API.
- Backend REST API
    - Created a REST API to simultaneously collect data from all the computers connected to the receivers.
    - Interact with the database by organizing data in accordance to their timestamps for synchronization.
    - Provide real-time position calculation simulation based on the received data from the receivers on the database by interacting with a position calculation module.
    - Interact with the client facing application to determine the location in real time.
- Position Calculation
    - Used API to get data from MongoDB via a request from the Central Processing Server (CPS).
    - Used AoA and AP position to visualize lines of signal arrival from transmitter.
    - Used AoA, transmitter position (fixed), and AP position to estimate the transmitter position by determining probability of transmitter at each location within a grid.
    - Sent CPS array of probabilities of where the transmitter is located within a fixed grid
    - CPS stores position estimation with a unique timestamp for future use.
- Data Visualization
    - Using calculated position information to produce a heatmap of probability densities
    - Able to click on each AP and get relevant information on position estimation
    - Able to visualize lines of AoA for each AP
    - Able to visualize where transmitter is located using heatmap
- Web Application Platform
    - Scalable web platform developed, can connect to database which has CSI data
    - Motivation tab gives insight into overall project and goals
    - Setup tab gives a step-by-step look into how this capstone project was developed
    - Able to filter and sort through CSI data in database

Mohaimen Khan

My major role for the project involved with setting up the end-to-end software infrastructure and building a client facing application, that could determine the position of the transmitter in real time. Coming into the project I brought in my experience of building scalable software systems. My first step was to help the team with designing the end to end system, from the collection of data from the hardware components to building individual microsystems, which eventually connected to the cloud database built on MongoDB [1].

Once an end to end software system was designed, I started developing microservices through software by dividing up the components. To allocate and help avoiding bottlenecks, I created an API endpoint for my teammates to collect and analyze data without relying on any external dependencies. Once setup was completed, I built a system to mock data transmission, collection and analysis modules. Furthermore, once the hardware could transmit data from one transmitter to one receiver, I built an entire web application on a MERN stack, which could expand to n transmitters and receivers.

I have also helped with developing end points for data collection and data analysis methods. I made up scalable pipelines for creating 1 to many connections between the data collectors and the API endpoints, allowing to seamlessly collect data in the future as well as within its present scope. Furthermore, data is synchronized on the API endpoint and stored in the database through UNIX timestamps. On the data analysis end, I have created UNIX processes to trigger microservices that analyze and return intersections of the arriving angles. This helps with the estimation of the position of the transmitter with respect to the receivers on the ends.

I have also tried helping with the expansion of our project's transmission mechanism to multiple transmitting objects. I have researched into transmission of custom Wi-Fi packets from my laptop (Appendix-L) and raspberry pi (Appendix-D), which could have helped us with data collection synchronously from multiple receivers at the same time.

Finally, I have created a consumer application, which uses the research to estimate the position of the object. It is production ready, and it has been packaged to be set up on any environment, which can be used in the future. This application takes the existing research and provides a visualization to the position of the object in real time, that can be triggered from the user end. This utilizes my built endpoint API, which is also connected to the data collection and data analysis microservices.

Shrey Parikh

My major role involved data processing and developing a data pipeline. This involved setting up the receiver (access point) to be able to receive data from the transmitter and store it. After cleaning the data, I was responsible for sending it to the central processing server (CPS). This required developing scripts to send automated requests to the CPS at every angle of arrival calculated (using MUSIC algorithm) at each access point (AP). This was a challenging task as it required to dive deep into the existing code base to learn how to route the CSI data into an into intermediately data representation for later use. I was responsible for developing a schema, with another teammate, for data sent between the APs and CPS. This schema was effective as it is used in our web application to enable filtering by time, access point, etc.; this enables future users to easily acquire custom datasets for use cases such as training machine learning models.

I was also responsible for developing the testing methods to determine the accuracy of our position estimate models. This task involved conferring with the lab supervisor and past project leads to understand the workflow and lab setup.

Through the course of this project, I was one of the members responsible for researching whether a Raspberry Pi (Version 3 Model B) could be used as an access point. This involved downloading, installing, and setting up multiple software requirements to be able to use GNU Radio. I was responsible for discovering that the Raspberry Pi needed a GNU Radio version of 3.8 after researching extensively through the Git Logs from GNU Radio (see Appendix D for more detailed accounts). To use GNU Radio 3.8, I compiled the program from source and set up multiple libraries. Since this did not work (Appendix D), I tried different avenues of installing the program such as PyBombs, the Raspbian Package Manager, and other methods of installation found through research. This work has been documented for future work.

I was also responsible in integrating the web application with our CSI data (via MongoDB). On the web application, I developed the content for the Motivation tab. Further, I published said content on the web application using the established framework developed by another team member. With the other team members, I was responsible for developing the methods to estimate the position of the transmitting device. Throughout the course of the project, I worked with various team members to develop scalable software architecture which could be seamlessly integrated - for example, the Data Collection and Data Analysis modules. Finally, I was responsible for integrating different platforms of our project into an end-to-end design with other team members.

Matthew Perry

My main contribution to the project was to estimate the transmitters position. This consisted of the design, development and testing of this module. I had started by creating a generic program to convert angles (i.e. angle of arrivals) to a geometric line on a 2D cartesian plane. This then allowed me to proceed calculating intersection points between each receiver's calculated angle of arrival and gather a range of possible locations in which the transmitter might be, on a centimeter level scale. Once I had arrived at this milestone, I had then started to look into possible ways in which I could use this new data. I had looked into the current methods used in other projects/products. This started with using a normal distribution to estimate the position and later looking into how the global navigation satellite system (GNSS) estimates positions of devices outdoors. I then worked on scaling the predicted position of the receiver from the centimeter level scale to meter accuracy (in which it is displayed on the front-end). Finally, this task had involved the integration of the position calculation module with the central processing server. This involved the module being called to start a new calculation, the module fetching the relevant data to make the calculation and the posting the data back to the central processing server, to be used for visualization.

I had also worked on getting the Raspberry Pi configured to be used as a transmitter and using the SDR boards as just receivers. A majority of this work involved looking into errors that arose when trying to install the required software, primarily GNU radio 3.8 (Appendix D). This involved looking up alternative methods of installation for the software as well trying these methods on multiple Raspberry Pis to rule out any system configuration issues/missed steps. In addition to the Raspberry Pi alternative, I had looked into the possibility of transmitting from a phone or laptop. In talking to the students from the prior summer (Radomir and Qinziyue) for guidance and suggestions they were also unable to achieve using a non-SDR transmitter.

Acknowledgements

We would like to thank everyone who supported us through the course of this project. We appreciate your advice, guidance, and mentorship.

We are grateful to have such inspirational colleagues and peers at the University of Toronto. This final year design project gives all of us the opportunity to showcase what we have learned over the course of our undergraduate studies. Every team in this course inspires us to push harder without own project and have influenced us greatly.

We would like to thank our course administrator, Prof. Timorabadi, for his mentorship and careful guidance. He has influenced us to make our project marketable, producible, and most crucially, feasible. We thank him for his advice, which we will take with us in our future careers.

We acknowledge Radomir Djogo and Qinziyue Zu, without whom this project would have been extremely difficult to set up. Their previous work on this project was a crucial foundation for our work and we appreciate their help in our endeavors.

We would like to especially acknowledge our course supervisor Prof. Valaee for his guidance and support. We feel very lucky to have been mentored by a leading researcher and thank Prof. Valaee for his key insights in improving our designs and his ability to explain how different components could come together.

Finally, our team would like to acknowledge one another for our constant support. This course introduced many challenges, but we were consistent in our ability to think positively and collaborate on solutions as a team. As engineers in our final year, we are humbled to be able to take so many life-long engineering lessons with us into the industry.

Table of Contents

1.0 Introduction

This section will introduce the background and motivation behind our capstone project. Further, in this section we will state the goal of our project. We will highlight the project constraints and project objectives that guided our design process and development.

1.1 Background & Motivation

With the introduction of GPS to the general public, outdoor positioning had been solved and GPS had transformed industries and spawned new technologies. For example, navigation, robotics, and agriculture were radically transformed [1]. However, indoor positioning is yet to be solved. The need for indoor positioning arises from indoor navigation and location. Indoor positioning can be extensively used in hospitals, manufacturing & robotics, and augmented reality [2]. For example, indoor localization can transform augmented reality since expensive external sensors, such as cameras, or inputs are no longer needed. Existing Wi-Fi access points can be used as beacons to localize headset devices; this research area is rapidly developing as the current localization accuracy is only a few metres [3]. Another example includes Amazon Go which uses indoor navigation, currently with sensor fusion and computer vision, techniques to transform the retail space [4]. Note that these techniques are expensive and require extensive power usage due to data transmission and procession [5]. As the world becomes more connected through the internet and the ubiquity of smartphones reaching even the most remote communities, there is a need for a robust indoor location system that can use these devices, along with the internet, to solve indoor positioning.

Traditional mapping and location methods such as the Global Positioning System (GPS) cannot be used indoors as those microwave signals from GPS satellites are distorted, scattered, and lose power – such signals do not penetrate the indoors at all if the surrounding building is constructed of material such as reinforced concrete. Currently, there is research being done on a variety of techniques for indoor location. For example, Bluetooth signals, RFID, FM signals, etc. are all being researched for their applications on indoor locating systems [2]. However, Wi-Fi is increasingly being adapted for indoor localization research due to its ability to scale, relatively low cost, and open access to the public [6]. A state if the art system, CiFi, can achieve an accuracy of 150 centimetres [7]. Other state of the art systems can achieve accuracy as well as 2 cm; however, these systems are not standard and have yet to be widely adapted and scaled.

The Channel State Information (CSI) is used to describe the phase and amplitude of a signal. Note that CSI is signal agnostic. To accomplish indoor localization, wireless access points (APs) are set at fixed, known locations. These APs detect and process signals from a transceiver. Further, APs extract metadata such as the phase delay and signal strength of the received signal. There are a variety of algorithms to use this CSI data for indoor localization. In particular, the MUSIC algorithm is such an algorithm used for localization. Using the collected signals and CSI data, the MUSIC algorithm can estimate the angle of arrival (AoA) and time of flight (ToF). These features can be used to estimate the location of the transmitting device on a 2D grid.

To set up APs Software Defined Radio (SDR) can be used. SDR is software to implement radio communication using an embedded system or a personal computer [8]. To emulate a transceiver, a Raspberry Pi can be used. Raspberry Pi boards are a minimal, single-board computer. These boards include a network interface card (NIC) which can be used to transmit signals.

To achieve an end-to-end design, where data received from receivers (access points) can be analyzed, processed, and visualized, a MEAN / MERN stack is desirable. A MERN is stack is a JavaScript software stack which is used to deploying web applications; this software stack consists of MongoDB, Express JS, React JS, and Node JS [9]. Deploying these web applications is a simple process which can easily be scaled [9]. This is desirable in the realm of indoor location as scale is essential when working with multiple nodes which process large amounts of data; the ability to visualize location estimates is crucial for use cases such as robotics.

To the best of our knowledge, a universal, scalable system which provides indoor localization data to be digested from an edge (such as an access point) and visualized is missing in current research.

1.2 Project Goal

The goal of the project is to develop an indoor positioning system to locate an object based on the information collected from wireless access point and to visualize this location.

1.3 Project Requirements and Constraints

1. Software Defined Radio (SDR) – Must have at least 3 APs set up using SDR
2. Must be able to locate an object within a 10 m x 10 m grid
3. Must be able to locate an object with at least the accuracy of RSSI (correct position within 2 m of accuracy) [10]
4. The Access Points must be able to collect data automatically and send it over to a server to analyze
5. The system should be accurately able to measure the change of angles, along with the location, with a change of position
6. The system must be able to collect Channel-State-Information data without modification of any existing hardware
7. Each access point within the system should collect at least 30 data packets out of the 56 packets broadcasted from the object for measuring an accuracy over state-of-the-art research.

1.4 Project Objectives

1. Each SDR should have at least 2 antennas. The number of antennas per SDR should increase for additional signals.
2. Machine learning techniques described in [11] and [12] should be used to reduce signal noise and increase position accuracy.
3. The front-end application should be able to track the position in real-time.

2.0 Final Design

2.1 System-level overview

Once the data is collected, the computers can collate and transmit the data over to a central Data Processing Unit (DPU) using a publisher-subscriber architecture. The data analysis component will use the data in the DPU to calculate the phase changes between each antenna. The calculated phase changes and the phase amplitudes will be used to estimate the object's location – which is sent to a central server. The data visualization component will use the current and past estimate of the object's location (records) to visualize the objects location through API calls to the processing server. A MongoDB-Express-React-Node JS (MERN) stack will be used for the full stack server and APIs to develop a monolithic application.

As shown in Figure-1(a) and Figure-1(b) with the section of system block diagrams, the system has been divided in to three components – data collection, data analysis and data visualization. The arrows within the system highlights the flow of data between the components. Firstly, data collection component firstly collects CSI data from the transmitting object via the antennas of the receivers. Each receiver is to a computer. Once the data is collected, the computers can transmit the data over to a central Data Processing Unit. Using a publisher-subscriber architecture, the computers can publish (POST) the data to the central processing server. The data analysis component will use the data in the DPU to calculate the phase changes between each antenna. The calculated phase changes and the phase amplitudes will be used to estimate the object's location – which is sent to a central server [13].

Finally, once the analysis finishes, the data visualization component will use current and past records are stored within the database to visualize the change in the object's location through API calls to the processing server. For the frontend server and the backend APIs with the data a MERN stack will be used for a monolithic application [14].

2.2 System-block diagrams

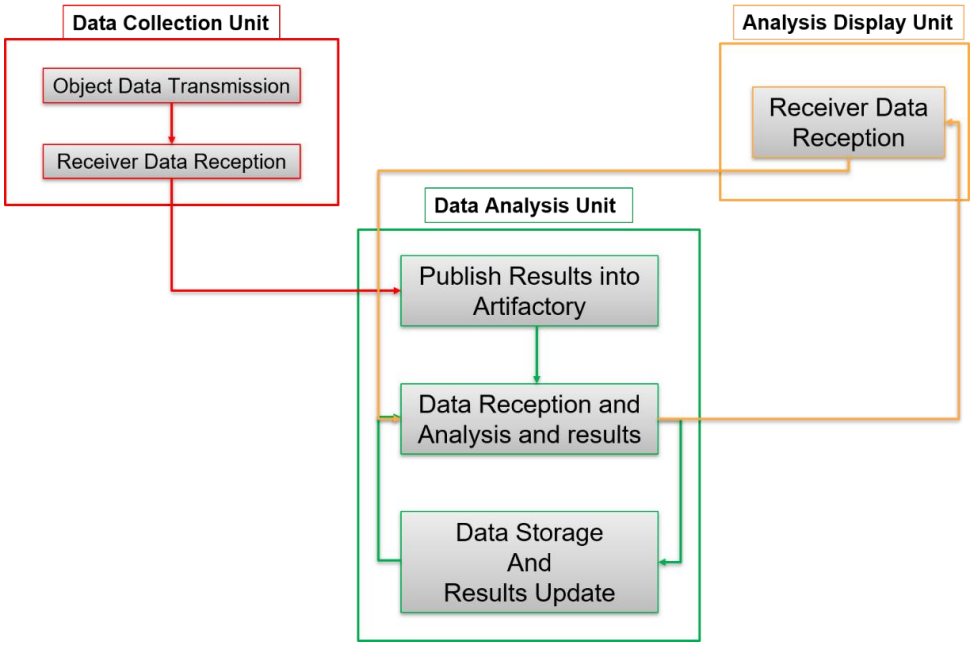


Figure 1(a). System level diagram and flow between component

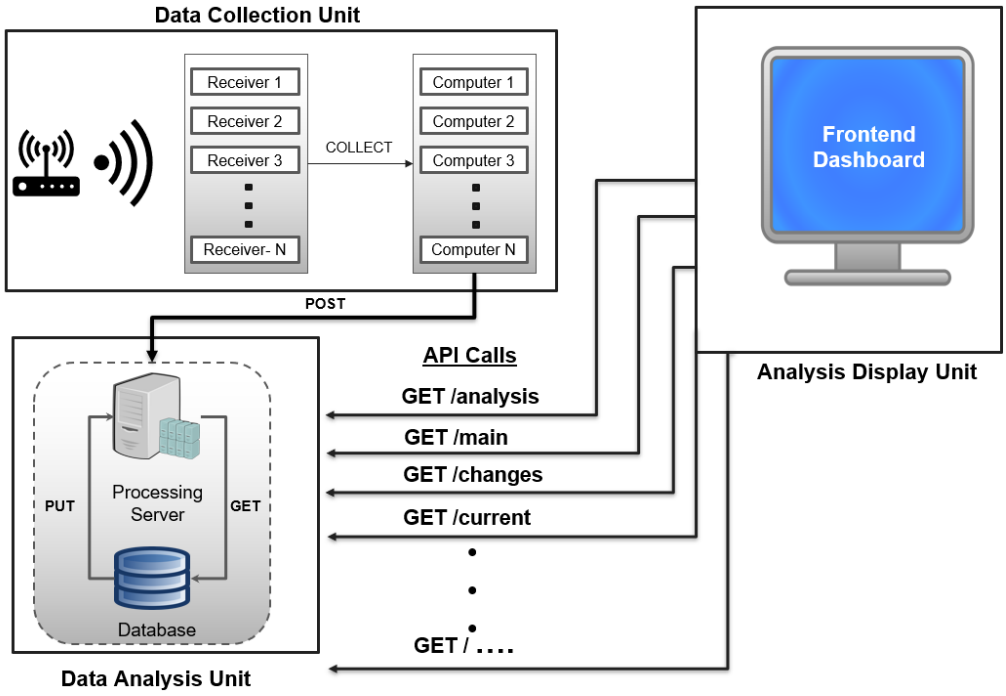


Figure 1(b). System level diagram and flow between components

2.3.0 Module Level Description

2.3.1 Data Collection from Transmitter (Author: S. Parikh)

Inputs	▪ CSI packets generated from the transmitting object ▪ Wireless Access Points with antennas
Outputs	▪ Angle of Arrival data from the wireless access points (figure 4 and 5)
Specifications	▪ Based on the requirements minimum of 3 wireless access points is needed. ▪ Each wireless access point is at a minimum distance of 3 meters apart. ▪ The transmitter generates packets using GNU radio, which allows the team to synchronize the packet data collected.
Functions	▪ This module collects data from an input transmitter. Using GNU radio, an open source software-defined radio system, transmitting packets are generated [15]. Using the two antennas between the receiver, any phase change is calculated.

Description: This module consists of the transmitter and access points to collect CSI data. The transmitter is an SDR board (LimeSDR), with attached antennas, connected to a laptop computer. This collective set of devices is an access point (AP). Work had been done to test various devices which better emulate a transmitting device. In Appendix D, we describe the significant progress done to use a Raspberry Pi (Version 3 Model B) as the transmitting device. In Appendix L, we list the work done to use a laptop computer, without an SDR board, as the transmitting device in this module. Each access point digests CSI data and calculates an Angle of Arrival (AoA) estimate using the MUSIC Algorithm. This calculation is done using GNU Radio modules.

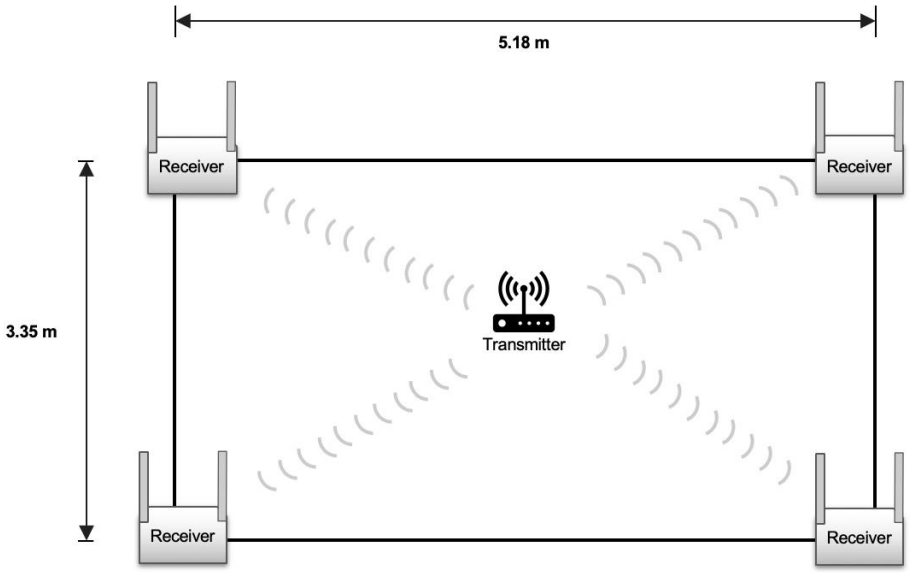


Figure 2. System Layout Diagram for data collection

The final test setup of Figure 2 inside the wireless lab in BA-4105 has been provided in Appendix E.

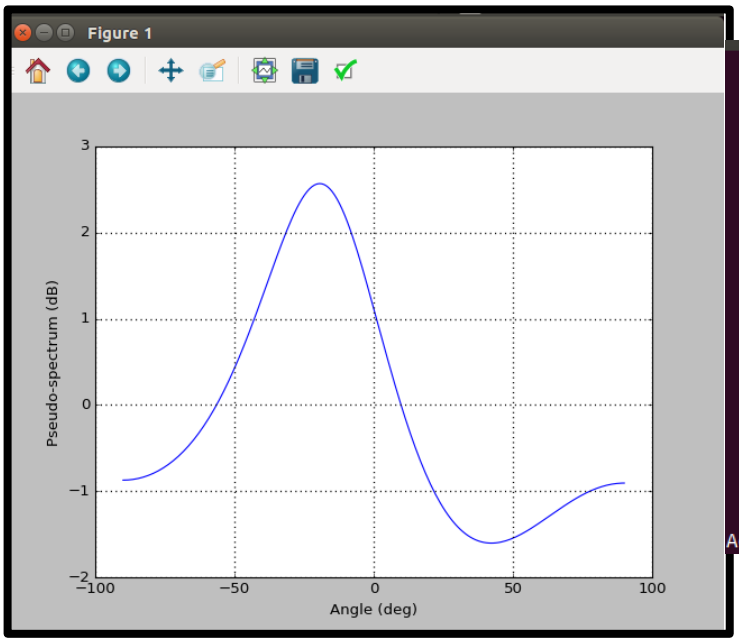


Figure 4. Phase Diagram at an AP

```
-0.3394183  -0.3236314  -0.30784451 -0.29205761 -0.27627071 -
-0.24469692 -0.22891002 -0.21312312 -0.19733622 -0.18154932 -
-0.14997553 -0.13418863 -0.11840173 -0.10261484 -0.08682794 -
-0.05525414 -0.03946724 -0.02368035 -0.00789345  0.00789345
 0.03946724  0.05525414  0.07104104  0.08682794  0.10261484
 0.13418863  0.14997553  0.16576243  0.18154932  0.19733622
 0.22891002  0.24469692  0.26048381  0.27627071  0.29205761
 0.3236314   0.3394183   0.3552052   0.3709921   0.386779
 0.41835279  0.43413969  0.44992659  0.46571348  0.48150038
 0.51307418  0.52886107  0.54464797  0.56043487  0.57622177
 0.60779556  0.62358246  0.63936936  0.65515626  0.67094315
 0.70251695  0.71830385  0.73409075  0.74987764  0.76566454
 0.79723834  0.81302523  0.82881213  0.84459903  0.86038593
 0.89195972  0.90774662  0.92353352  0.93932042  0.95510731
 0.98668111  1.00246801  1.01825491  1.0340418   1.0498287
 1.0814025   1.09718939  1.11297629  1.12876319  1.14455009
 1.17612388  1.19191078  1.20769768  1.22348458  1.23927147
 1.27084527  1.28663217  1.30241906  1.31820596  1.33399286
 1.36556666  1.38135355  1.39714045  1.41292735  1.42871425
 1.46028804  1.47607494  1.49186184  1.50764874  1.52343563
 1.55500943  1.57079633]
AoA is at: -32.11055276381909 degrees
```

Figure 5. CSI data and AoA Estimation using MUSIC algorithm

2.3.2 Central Processing Server (Author: M. Khan)

Input	▪ POST requests from computers connected with receivers (Data Collection Unit). ▪ GET/POST requests to fetch and store data (Data Analysis Unit). ▪ GET requests from the frontend server (Data Visualization Unit).
Output	▪ Store receiver CSI data to the cloud database connected to the central server. ▪ Synchronize data from receivers to calculate the position of the input object. ▪ Send data to frontend to estimate change of angle within a space.
Specifications	▪ Monolithic backend Node.js server built using Express.js framework, allowing concurrent requests at the same time [16]. ▪ Non-relational MongoDB database to scale for large amount of data [17].
Functions	▪ Central processing server handles all the API calls. After the computers connected to the receivers publish the data into the processing server through POST call, the central processing server synchronizes the data from all the receivers. ▪ Once all the data is processed, the central server stores the data in the database as a record for past data processing purposes. ▪ Central processing server estimates the phase changes through the data and provides measurements to the frontend for visually estimating the changes.

Description: Backend server is solely responsible for handling any incoming and outgoing requests. It is running on an Express.js API [18], connected on a MERN web stack [14]. Once the server is started on a NPM module, the server is able to connect the data collecting components to the data storage and frontend components [19]. The API has been limited to have a payload of max 50MB, based on the best practices guidelines provided by Express.js [20]. The server has been designed based on the MVC framework used to develop standard monolithic APIs [21], allowing separation of database models and respective controllers to handle the corresponding logics to the routes.

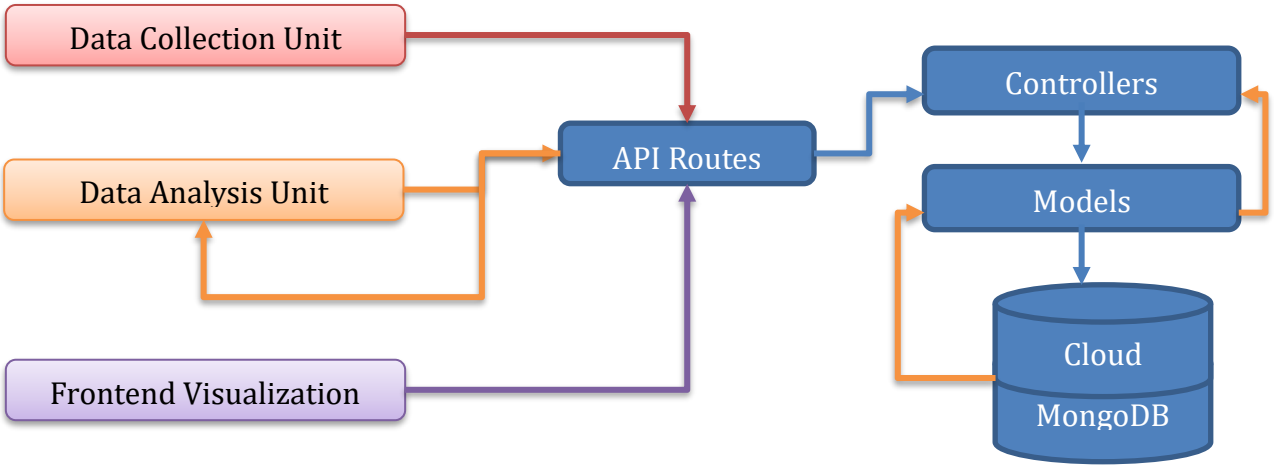


Figure 6. A visual representation of the monolithic server with MVC framework

The instructions to set up the server and installation of corresponding packages have been provided under Appendix I. For setup convenience purposes, the server has been packaged with all the provided libraries under package.json [Appendix I]. To extend the backend over a local network local tunneling has been used, allowing computers with receivers connected on the same local network to make POST request calls to the central processing server and publish data [22].

The following modules are all connected to the central processing server and they all communicate to the end user through REST API calls [23]. A list of API calls used by the different units have been provided in the following table with functionalities [Appendix I]. A full description of all the functions has been provided in Appendix I with their supporting code:

Main route	API route	Function	HTTP Request	Description	Used By
Port-8888/	/receiver	/addData	POST	Allows the individual computers to make a post request and store data from all receivers.	Data Collection Unit
	/analysis	/allData	GET	Allows to retrieve all the stored data from all the receivers.	Data Analysis Unit
		/byTime	GET	Allows to retrieve receiver data stored within a specific time frame.	Data Analysis Unit
		/storeAnalysisData	POST	Allows data analysis unit to automatically store all positioning data from all 4 receivers post analysis, along with receivers val.	Data Analysis Unit
		/sendAnalysisData	POST	Allows frontend to retrieve the analysis values within a specific time frame.	Frontend Visualizer Unit
		/getAllDates	GET	Retrieves all the dates and times for past runs.	Frontend Visualizer Unit
		/getCurrentRun	GET	Runs the end to end module automatically.	Frontend Visualizer

Table 1. API calls and their corresponding routes [Appendix I]

2.3.3 Data Publishing Method to the Central Processing Server (Author: S. Parikh)

Inputs	▪ Phase change data from the wireless access points (Data Collection Unit)
Outputs	▪ Publish data to the processing server for analysis (Data Analysis Unit)
Specifications	▪ All connected computers from the receivers need to be able to make POST request to the central processing servers.
Functions	▪ Each receiver is connected to a computer which needs to be connected within the same local network as the server. ▪ Using POST request, each computer can publish data on every receiver to the central processing server

Description: Each Access Point is connected to the central processing server. Once local tunneling exposes the backend API to the local network, each computer with the receiver can make POST request through a python script, packetToDatabase.py. The python script scrapes the data output from GNU radio and generates a packet consisting of CSI data, Angle of Arrival, Receiver number and the Unix timestamp.

```
shrey@Shreys-Laptop MINGW64 ~/OneDrive/Documents/Engineering/Capstone/Indoor-Localization-Capstone-Project/scripts (master)
$ python packetToDatabase.py
Sending post request ...
Data: {'access_point': 1, 'timestamp': '2020-02-20T12:11:42.531123', 'angle_of_arrival': '42.0603015075377'}
Reqest response: <Response [200]>
Sending post request ...
Data: {'access_point': 1, 'timestamp': '2020-02-20T12:11:42.648975', 'angle_of_arrival': '-65.57788944723619'}
Reqest response: <Response [200]>
```

Figure 7. POST Request Output of packetToDatabase.py

Figure 8. Packet generated by packetToDatabase.py

Once the packet has been created, the data is transferred to the Central Processing Server through a POST request. Express.js API receives the data and formats it to store into the MongoDB database in the form of the schema defined so that it can later be used by the data analysis unit:

```
let receiveSchema = new Schema({
    access_point: { type: Number, default: -1},
    timestamp: { type: Date, default: Date.now },
    angle_of_arrival: { type: Number},
    csi_frame: {type: Array, default: null},
    grid: {type: Array, default: null}
});
```

Figure 9. Receiver Data Schema

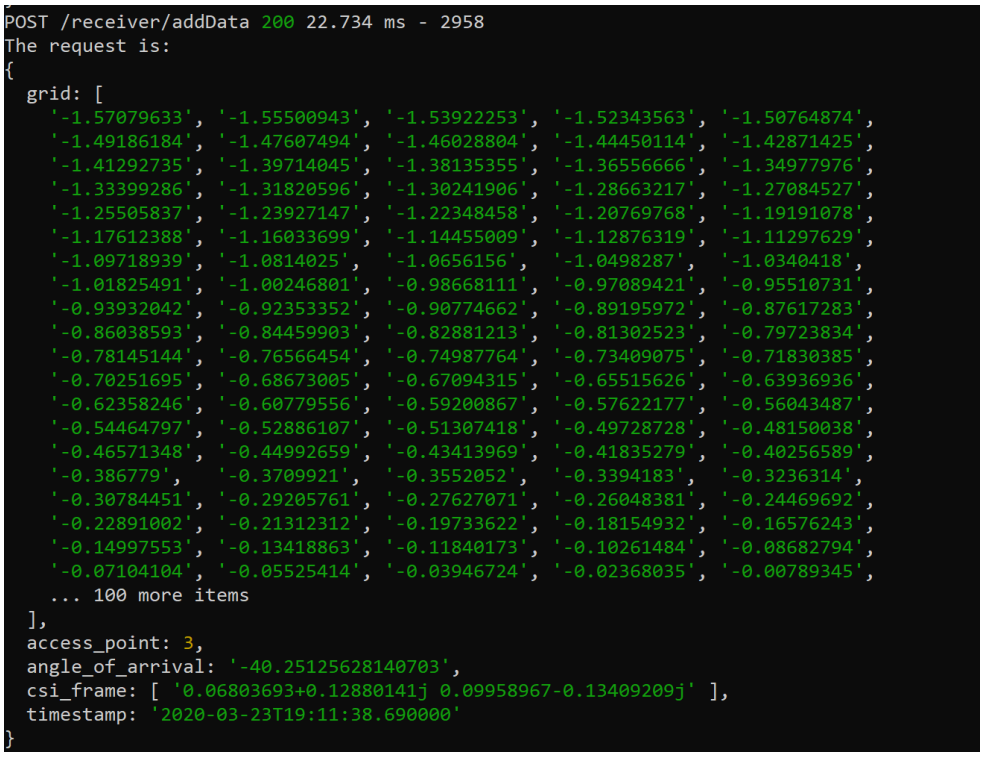


Figure 10. Example of Processing Server receiving data from AP 3

2.3.4 Data Analysis Module within the Central Processing Server (Author: M. Perry)

Input	▪ All receiver data, consisting of Angle of Arrivals and CSI data, through an automated call from central processing server to analyze the latest dataset.
Output	▪ Synchronize data from all the receivers to provide an estimate the position of the transmitters with respect to all receivers on a grid. ▪ Provide an outlined dimension of the area covered to estimate the position.
Specifications	▪ Output data grid must include all the receiver positions on the specified grid. ▪ The analysis unit should be an independent microservice which can be triggered through automated call from central processing server for automation purposes.
Functions	▪ Retrieve data from the Central Processing Server through GET calls and estimate the probability of the position of an object from the intersection of angles of arrivals from all the receivers.

Description: Data analysis unit has been separated out as an individual microservice from the module. This has been completed to ensure independency over automated call from the frontend, allowing the users of the end-product to run it in real time. With an API call from the frontend, the central processing server spawns a python process to run the service in real time. The module fetches all the data from the receivers through a GET call, allowing synchronization of the data.

The module returns a position grid with an estimate of the object position corresponding the intersection of the angles of arrivals and normalization of the data with an estimation factor of 1/3. A detailed calculation and the functionality of the module has been provided in Appendix H.

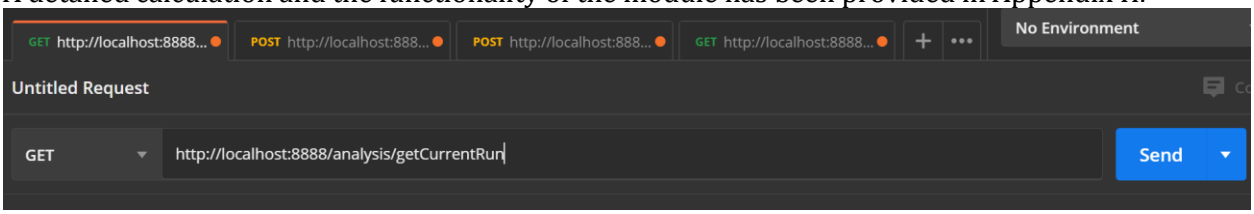


Figure 11. Running the entire module through a GET request on Postman [24]

Once the module is triggered, the module it fetches all the receiver data from the API using a GET request as following, as it returns the equation of the lines of the angle of arrivals from all receivers.

```
Adding new receiverData -> receiver: 3  timeStamp: 2020-03-23T23:15:46.470Z  AoA: -40.2512562814
Adding new receiverData -> receiver: 4  timeStamp: 2020-03-23T23:15:46.050Z  AoA: -7.68844221106
Adding new receiverData -> receiver: 2  timeStamp: 2020-03-23T23:15:45.876Z  AoA: 10.4020100503
Adding new receiverData -> receiver: 1  timeStamp: 2020-03-23T23:15:45.698Z  AoA: -2.26130653266
y = -0.846600130229*x + 773.538867459
y = -0.134999901212*x + 69.9299488276
y = 0.183570608548*x + 0.0
y = -0.039487749395*x + 335.0
```

Figure 12. Python module outputting lines of intersection

To confirm the heatmap the team was able to generate a heatmap using matplotlib provided by python:

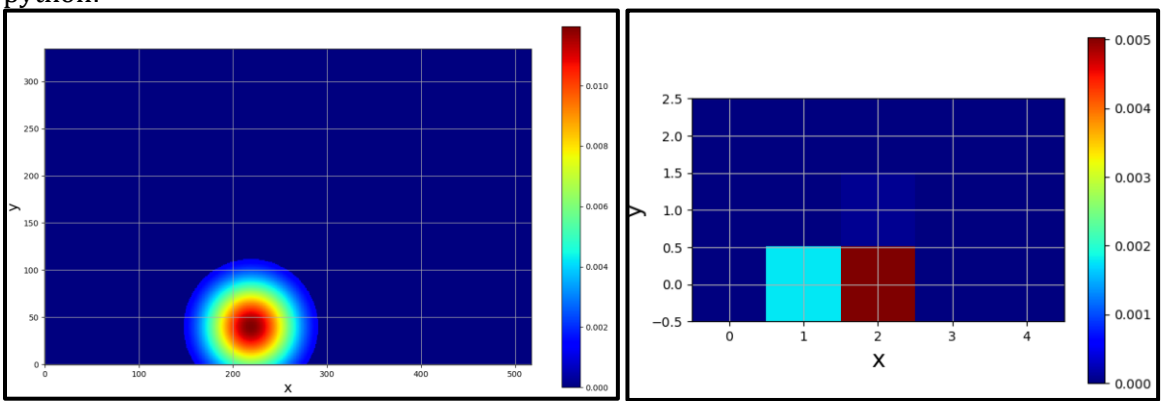


Figure 13. Heatmap generated using python Matplotlib (a) in centimeters (b) in meters

Once the data is calculated, it gets sent over the database through a POST call. The central processing server receives the data in the following schema, and stores data inside the database:

Figure 14. Schema and Data storage from the API

2.3.5 Data Visualization Module (Author: M. Khan)

Inputs	▪ Geodata calculated to visualize the phase changes. ▪ Layout of the space the access points and the transmitter are set.
Outputs	▪ Phase change estimation through visualization. ▪ Heatmaps of past runs over time.
Specifications	▪ A react.js server can be used for frontend visualization. This will be a part of the MERN stack, allowing easy integration [14]
Functions	▪ Display the data of phase changes over time. ▪ Provide a visualization on how the product works. ▪ Run the end to end-product in real time.

Description: The frontend module allows the users to estimate their locations. It is currently a dashboard created over localhost on react.js [25]. A detailed description of the hierarchy of the components with their associated directories have been provided in Appendix J

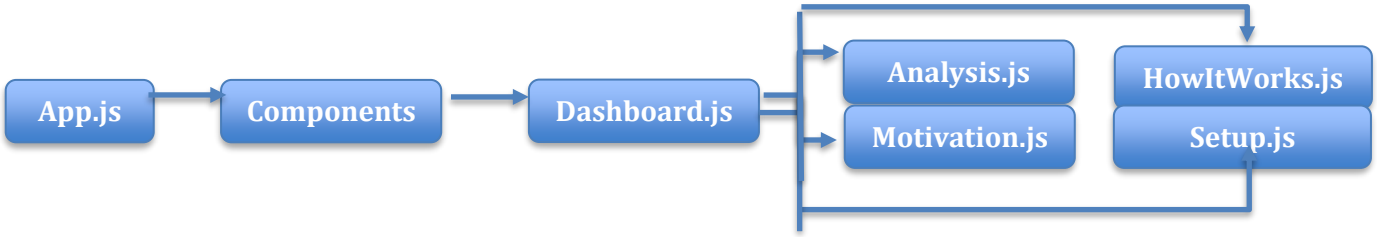


Figure 15. Hierarchy of Frontend

The dashboard fetches all the data analyzed from the backend, by making API calls over network.

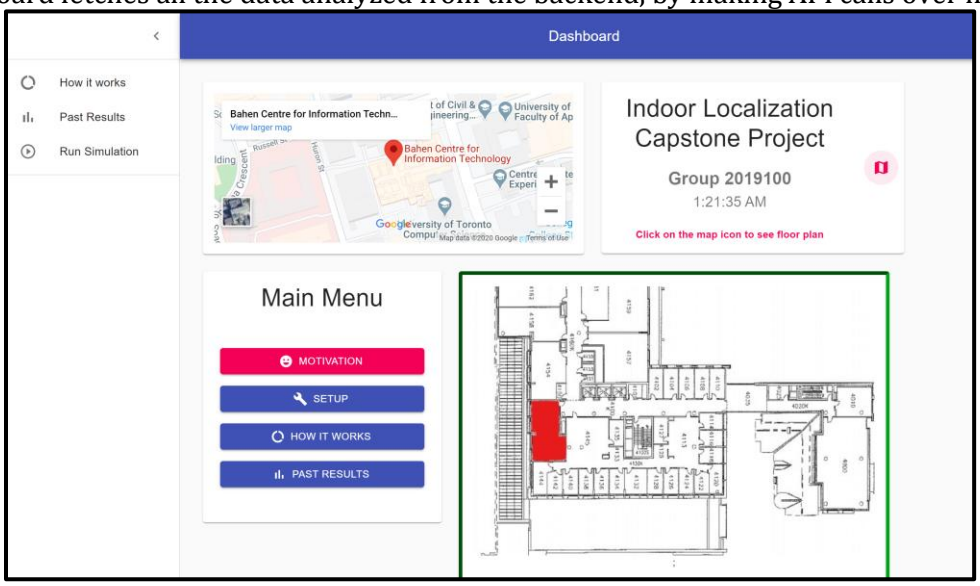


Figure 16. Landing Page of the dashboard

The dashboard allows users to trigger a position estimation run in real time. With run simulation, users can fetch results by running the Data Analysis unit in backend. Furthermore, users can fetch past reports by fetching all the past results stored in the databased through API call:

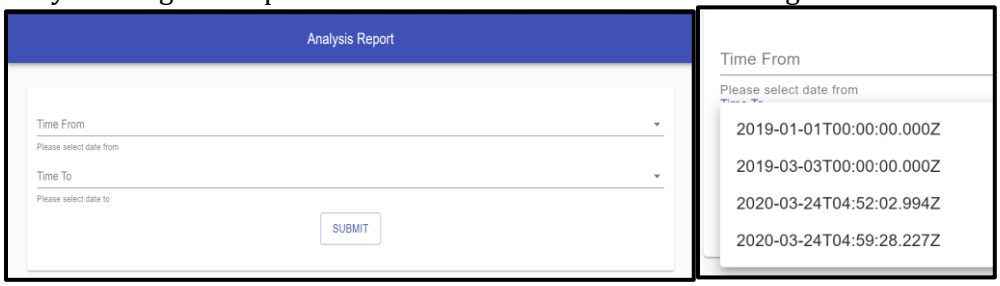


Figure 17. Analysis Report Form with dates to select from

Once the reports are selected, the frontend software generates the heat map to show an estimate of the position of the object, through a heat map as following:

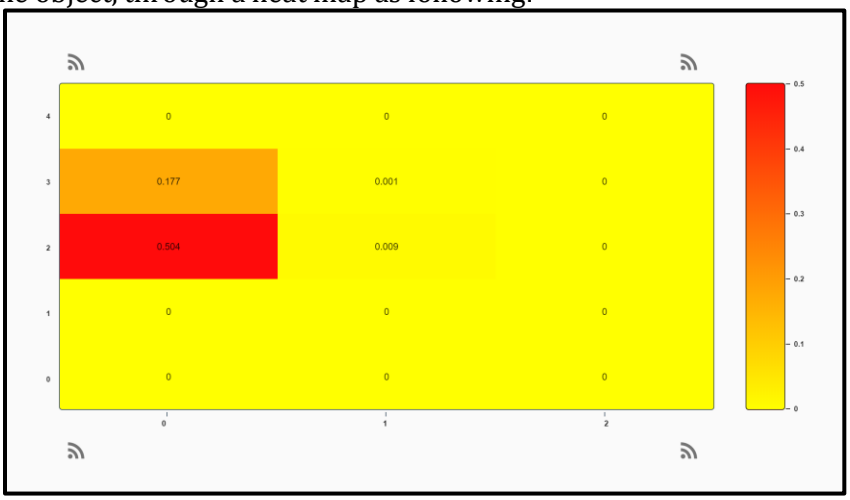


Figure 18. Heatmap generated from the shown data with a position probability out of 1

With the selection of each receiver, the line of intersection can be shown on the visualized heatmap. It identifies the components and shows how the module is currently with a simulation module:

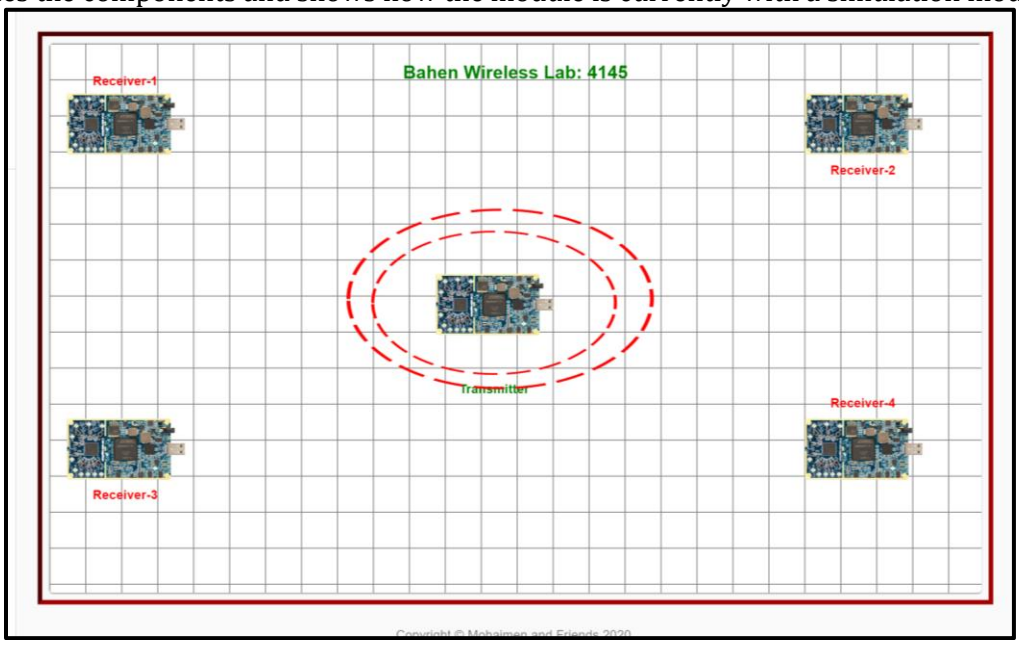


Figure 19. Simulation of how the transmitters and receivers are set up

2.4 Assessment of Final Design

The team has decided to assess all the provided modules, mentioned above, based on the categories of strengths, weaknesses and any tradeoffs. Any images corresponding to the section have been provided in Appendix K

Module	Strengths	Weaknesses	Tradeoffs and Considerations
Data Collection from Transmitter	The module can incorporate CSI data over 5 meters over the minimal requirements of 3 meters. It creates custom Wi-Fi packets using IEEE-802 to emulate the environment with no existing positioning methods [26].	The module is not able to collect from 4 separate access points at the same points. It only reads from one access point at a time. As a result, results can not be fetched in absolute real time and a manual intervention is necessary.	A setup with 4 separate access points have been manually set up to overcome the situation [Appendix E]. This helped the team to collect data and analyze it to estimate position.
Central Processing Server	The module can handle a payload of 50 mb JSON encoded data [20]. The server can handle a minimal of 2844 requests/second, which allows scalability upto 2844 receivers [Appendix K].	JavaScript based servers are not computationally competent in comparison to other web servers built on Java, PHP and Go programming languages [27].	JavaScript MERN based web framework allowed quick setup of the application. Data analysis calculations have been separated in different python module [Appendix H].
Data Publishing Method to Central Processing Server	Data publisher scrapes through the text files generated from the receivers. This CSI grid data and necessary timestamps, that are used in the backend to synchronize data.	The data scraper only accepts the file format generated by GNU radio now and it requires manual intervention.	
Data Analysis Module within Central Processing Server	The unique module calculates the position of the object in real time through the intersection of data points from angles of arrivals [Appendix K]		
Data Visualization Method on Frontend	React.js allows users to run it as a web on any platform. The scalable architecture allows users to run it from their cell phones, and computers in real time		Frontend server has been chosen to be on React.js to expand to an application used by consumers as an end-product.

3.0 Testing and Verification

ID	Project Requirement	Verification Method	Verification Result and Proof
1 a.	Software-Defined-Radio	TEST: Start the measurement of the phase changes through a physical NIC. Then use SDR to do the same. SDR should return a more breakdown structure of the internal phase changes within the access points.	Pass. See section Data Collection from Transmitter
1 b.	Access Points Number: 3	TEST: Direct measurement.	Pass. See section Data Collection from Transmitter.
2.	Margin of Error	TEST: Comparing if direct measurement of position and calculated position is less than 2 m.	Pass. See figure 13, showing the probability distribution of where the transmitter is located
4 a.	Automatic Data Collection	TEST: Start the locating system and check the collected data after collection processes ends and confirm data from all APs has been received during the time of the test	Pass. See Data Analysis Module for details. Results of runs provided in Appendix K.
4 b.	Independent Deployable Architecture	TEST: To do any direct measurements, none of the existing hardware needs to be changed.	Pass. All the servers have been packaged with npm libraries [19]. See Appendix I for setup details.
5.	Measure changing angles and position	TEST: Direct measurement.	Pass. See section Data Collection from Transmitter and Appendix F.
6.	Using CSI	TEST: Confirm signals sent and received by all APs and the server are strictly CSI signals. This can be done a location not affected by Wi-Fi signals, indoors and outdoors.	Pass. See section Data Collection from Transmitter and Appendix G.
7.	Number of packets received by each AP	TEST: Develop a subroutine to count the number of packets received by each AP and compare to the number sent by the transmitter.	Not passed. Could not measure directly due to laboratory closure.

To perform the testing we used one receiver, which was moved between receiver locations, to collect data. The receiver locations were picked to be the corners of the laboratory measured, as shown in figure 2. For the transmitter, a fixed position was picked, to make the data time independent, and measured the same way as the receivers. We had performed this process multiple times to ensure valid data was being received, identify any issues with the configuration and to a range of error for the angle of arrival being returned. We were able to

4.0 Summary and Conclusions

Indoor localization is an existing problem which cannot be accurately solved with existing technology of using GPS data due to the limitations of available data. A robust indoor location system requires a design which can use geo and object data, which are not limited by the number of access points and barriers within a location.

Our end-to-end design digests data at an edge device and can visualize where the transmitter is located. In the development of our design, our test edge device was an SDR board connected to a computer i.e. an access point (AP). We were able to connect to our access point central processing server (CPS). At the access point, we cleaned and processed data to send to the CPS – this data would be used to calculate the position of a transmitting device. The position is estimated using the data sent to the CPS, through analyzing the angle of arrivals from each access point. The intersection points between the angle of arrivals, from each receiver, were found. The position and distance between these intersection points was used to generate the final estimated position, which was then sent back to the CPS. Since this data processing relied on data schemas, it is highly reliable, scalable, and maintainable. Although there are drawbacks with our design, chiefly the modern software required, we believe our end-to-end indoor localization system is scalable. While our design uses CSI data to estimate transmitter location, CSI data can be extracted from a variety of signal sources.

We were able to meet most of our requirements. We were not able to meet requirement 7 due to the closure of our laboratory. This impacted our ability to further test our design as well – although significant data had been collected prior to the closure. Further, while we were not able to use a Raspberry Pi or a laptop computer as a solely emitting device, significant work had been documented on how to set up such devices for use in indoor location research.

Based on how our design has met requirements and objectives, we feel our design can have an immediate impact in society. Since wireless mesh networks are becoming increasingly common, our design can be easily integrated with such networks and digest data. Since our data processing pipelines rely on schemas, which are modular and can be modified, custom data schemas can be developed for each use case. Further, our MERN stack allows us to scale our web application and database, this allows our design to work on a relatively large network in the same method as our testing network. Further work must be done to ensure multiple access points can calculate the position of a transmitting device as it is moving from one location to another. Additionally, further work must explore how multiple transmitting devices and access points interact with one another. It is possible that with signal noise and distortion, machine learning techniques must be integrated into this design to further process data to reduce noise.

References

[1] "GPS.gov: Applications," Gps.gov, [Online]. Available: https://www.gps.gov/applications/. [Accessed 08 10 2019].

[2] J. Karnik and K. Streit, "Summary of available indoor location techniques," *IFAC-PapersOnLine,* vol. 49, no. 25, pp. 311-317, 2016.

[3] P. Herbers and M. König, "Indoor Localization for Augmented Reality Devices," *Applied Sciences - MDPI,* vol. 9, no. 20, p. 4260, 2019.

[4] N. Stein, "Indoor Positioning System & Amazon – What is their next move?," indoors, 29 11 2016. [Online]. Available: https://indoo.rs/amazon/. [Accessed 08 10 2019].

[5] B. Lashkari, J. Rezazadeh, R. Farahbakhsh and K. Sandrasegaran, "Crowdsourcing and Sensing for Indoor Localization in IoT: A Review," *IEEE SENSORS JOURNAL,* vol. 19, no. 7, pp. 2408-2434, 2019.

[6] K. Wu, J. Xiao, Y. Yi., D. Chen, X. Luo and L. Ni, "CSI-Based Indoor Localization," *IEEE Transactions on Parallel and Distributed Systems,* vol. 24, no. 7, pp. 1300-1309, 2012.

[7] X. Wang, X. Wang and S. Mao, "CiFi: Deep convolutional neural networks for indoor localization with 5 GHz Wi-Fi," *2017 IEEE International Conference on Communications (ICC),* pp. 1-6, 2017.

[8] "What is Software Defined Radio," Wirelessinnovation.org, 2019. [Online]. Available: https://www.wirelessinnovation.org/assets/documents/SoftwareDefinedRadio.pdf. [Accessed 19 09 2019].

[9] O. C. Jude, "How To Get Started with the MERN Stack," Digital Ocean, 12 December 2019. [Online]. Available: digitalocean.com/community/tutorials/getting-started-with-the-mern-stack. [Accessed 23 March 2020].

[10] U. Bekclibasi and M. Tenruh, "Increasing RSSI Localization Accuracy with Distance Reference Anchor in Wireless Sensor Networks," Mugla S. K. University, 2014.

[11] X. Wang, L. Gao, S. Mao and S. Pandey, "CSI-based Fingerprinting for Indoor Localization: A Deep Learning Approach," *EEE Transactions on Vehicular Technology,* p. 1, 2016.

[12] X. Wang, X. Wang and S. Mao, "CiFi: Deep convolutional neural networks for indoor localization with 5 GHz Wi-Fi," *2017 IEEE International Conference on Communications (ICC),* pp. 1-6, 2017.

[13] Group Name: Awesome Possum, "Publish Subscribe Architecture," *Publish Subscribe Architecture,* p. 4, 2017.

[14] MERN.JS, "MERN.JS," MERN.JS, 2019. [Online]. Available: https://www.mernjs.org/.

[15] The GNU Radio Foundation, Inc, [Online]. Available: https://www.gnuradio.org/. [Accessed December 2019].

[16] Node.js, "Express," Node.js, [Online]. Available: https://expressjs.com/. [Accessed 1 November 2019].

[17] MongoDB, Inc., "Big Data Explained," MongoDB, 2019. [Online]. Available: https://www.mongodb.com/big-data-explained. [Accessed December 2019].

[18] Node.js Foundation, "express() API," Node.js Foundation, 16 November 2010. [Online].

[19] NPM Inc., "https://www.npmjs.com/," npm, 12 January 2010. [Online]. Available: https://www.npmjs.com/.

[20] Node.js Foundation, "Production best practices: performance and reliability," Node.js Foundation, 16 November 2010. [Online]. Available: https://expressjs.com/en/advanced/best-practice-performance.html.

[21] Future. Co-created., "API services & MVC," Futurice, 2000. [Online]. Available: https://www.futurice.com/blog/api-services-mvc.

[22] Node.js Foundation, "localtunnel," Node.js Foundation, [Online]. Available: https://www.npmjs.com/package/localtunnel.

[23] Z. Liew, "Understanding And Using REST APIs," Smashing Magazine, [Online]. Available: https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/.

[24] Postman, Inc., Postman, Inc., 2014. [Online]. Available: https://www.postman.com/.

[25] J. Walke, "React," Facebook and community, 29 May 2013. [Online]. Available: https://reactjs.org/.

[26] S. Zirari, P. Canalda and F. Spies, "WiFi GPS based Combined positioning Algorithm," *IEEE Int. Conf. on Wireless Communications, Networking and Information Security,* no. hal-00563629, pp. 3-4, 2010.

[27] B. Peabody, "Server-side I/O Performance: Node vs. PHP vs. Java vs. Go," Toptal, [Online]. Available: https://www.toptal.com/back-end/server-side-io-performance-node-php-java-go.

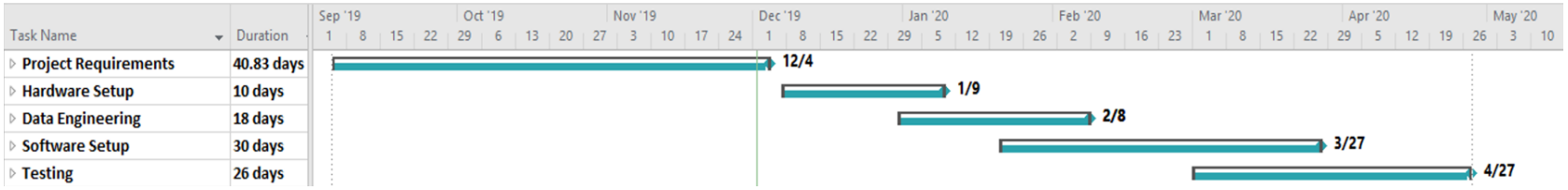
[28] Z. Yang, Z. Zhou and Y. Liu, "From RSSI to CSI," *ACM Computing Surveys,* vol. 46, no. 2, pp. 1-32, 2013.

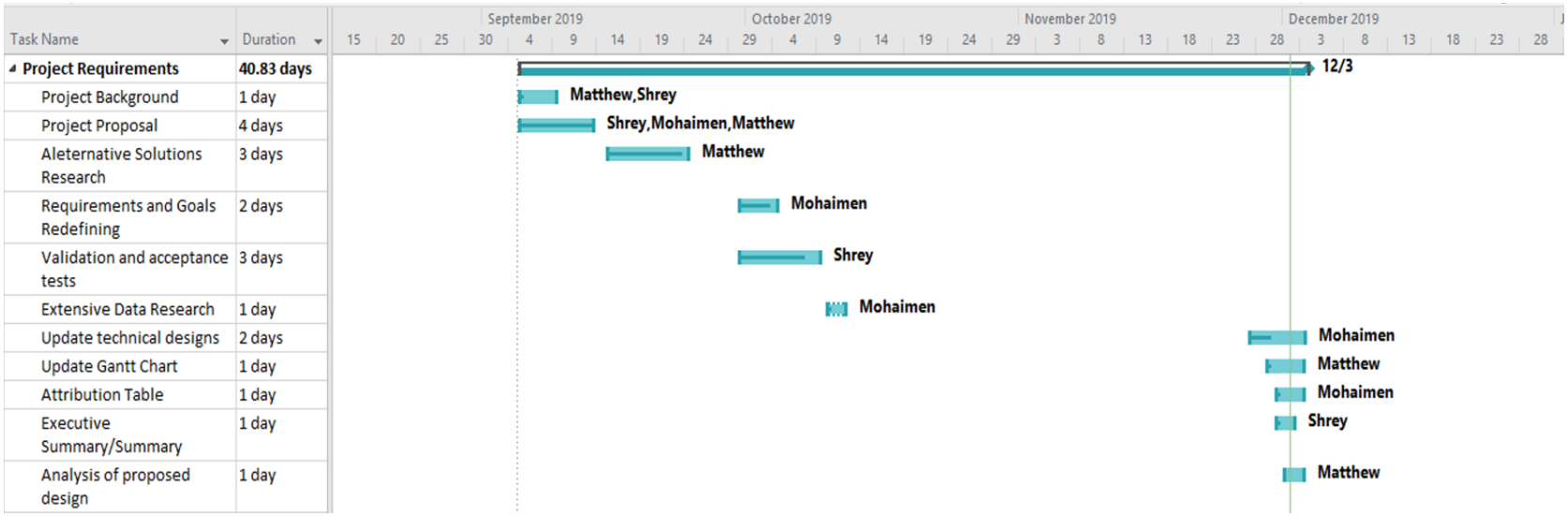
[29] C. Zhang, P. Patras and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," *IEEE Communications Surveys & Tutorials,* vol. 21, no. 3, pp. 2224-2287, 2019.

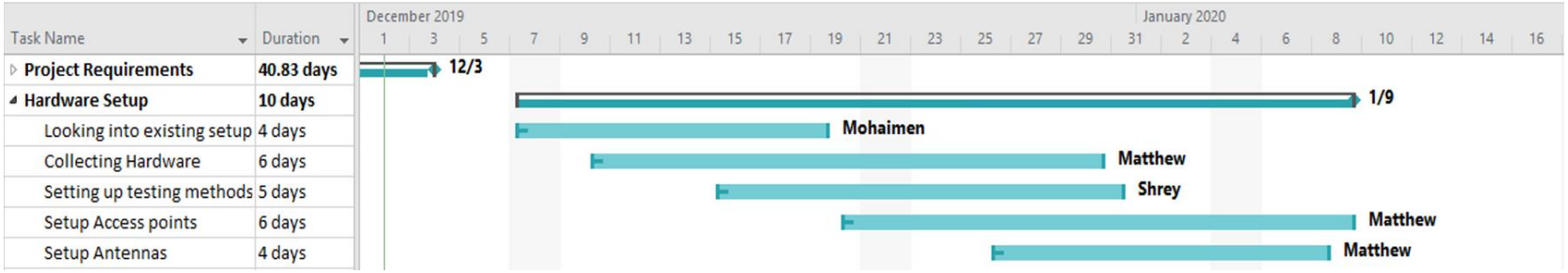
[30] "What is GNSS?," 29 August 2017. [Online]. Available: https://www.gsa.europa.eu/european-gnss/what-gnss.

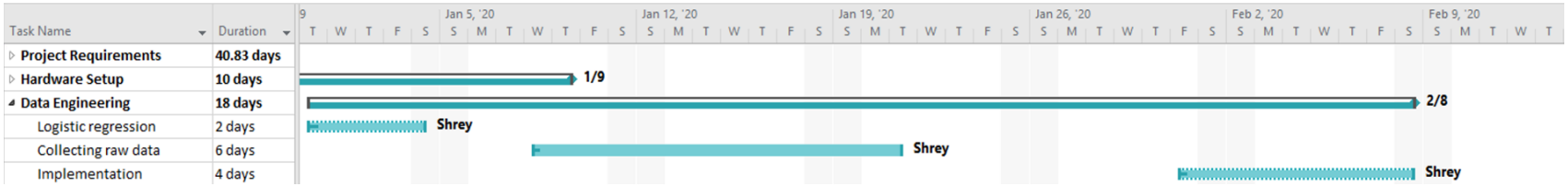
[31] Z. YANG, Z. ZHOU and Y. LIU, "From RSSI to CSI: Indoor Localization via Channel Response," *ACM Computing Surveys,* pp. 25-32, 2013.

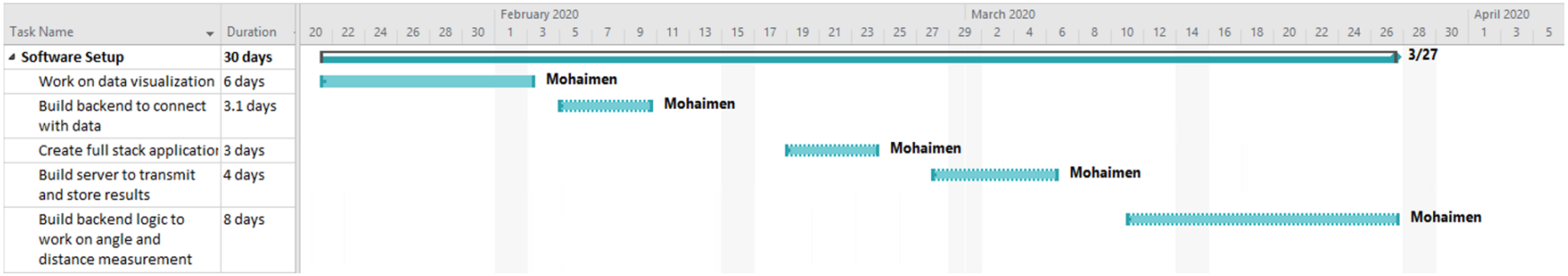
[32] V. Loscri and A. M. Vegni, "Software Defined Radio," 2015. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/software-defined-radio.

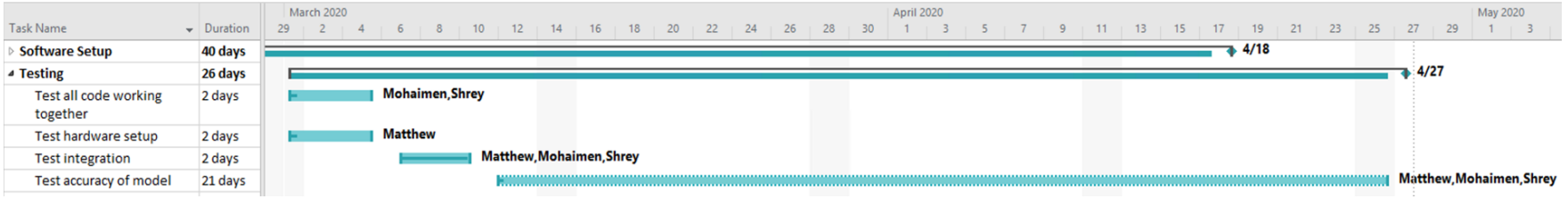

Task Name
Duration
Sep '19
Oct '19
Nov '19
Dec '19
Jan '20
Feb '20
Mar '20
Apr '20
May '20
Project Requirements
40.83 days
Hardware Setup
10 days
Data Engineering
18 days
Software Setup
30 days
Testing
26 days
12/4
1/9
2/8
3/27
4/27


Task Name
Duration
September 2019
October 2019
November 2019
December 2019
Project Requirements
40.83 days
Project Background
1 day
Project Proposal
4 days
Aleternative Solutions Research
3 days
Requirements and Goals Redefining
2 days
Validation and acceptance tests
3 days
Extensive Data Research
1 day
Update technical designs
2 days
Update Gantt Chart
1 day
Attribution Table
1 day
Executive Summary/Summary
1 day
Analysis of proposed design
1 day
12/3
Matthew,Shrey
Shrey,Mohaimen,Matthew
Matthew
Mohaimen
Shrey
Mohaimen
Mohaimen
Matthew
Mohaimen
Shrey
Matthew


Task Name
Duration
December 2019
January 2020
Project Requirements
40.83 days
Hardware Setup
10 days
Looking into existing setup
4 days
Collecting Hardware
6 days
Setting up testing methods
5 days
Setup Access points
6 days
Setup Antennas
4 days
12/3
1/9
Mohaimen
Matthew
Shrey
Matthew
Matthew


Task Name
Duration
Jan 5, '20
Jan 12, '20
Jan 19, '20
Jan 26, '20
Feb 2, '20
Feb 9, '20
Project Requirements 40.83 days
Hardware Setup 10 days
1/9
Data Engineering 18 days
2/8
Logistic regression 2 days
Shrey
Collecting raw data 6 days
Shrey
Implementation 4 days
Shrey


Task Name
Duration
February 2020
March 2020
April 2020
Software Setup 30 days
3/27
Work on data visualization 6 days
Mohaimen
Build backend to connect with data 3.1 days
Mohaimen
Create full stack applicatior 3 days
Mohaimen
Build server to transmit and store results 4 days
Mohaimen
Build backend logic to work on angle and distance measurement 8 days
Mohaimen


Task Name
Duration
March 2020
April 2020
May 2020
Software Setup 40 days
4/18
Testing 26 days
4/27
Test all code working together 2 days
Mohaimen,Shrey
Test hardware setup 2 days
Matthew
Test integration 2 days
Matthew,Mohaimen,Shrey
Test accuracy of model 21 days
Matthew,Mohaimen,Shrey

Appendix B - Financial Plan

Upon discussions with our supervisor, Professor Valaee, it has been established that additional equipment, hardware or software, should not need to be purchased. There are currently 3 LimeSDR USB receivers that can be used and will be enough for our uses in this project. For software in this project we'll use GNU Radio Framework, the same software as in a capstone project last year and a project over the summer. This is a free piece of software that's responsible for the collection and handling of data from the SDR.

Since the project proposal, we have purchased a Raspberry Pi (Version 3 Model B) for our development.

Item	Priority	Cost/Unit	Quantity	Total Cost	Requires Funding	Kept/Paid for by students
Raspberry Pi (Version 3 Model B)	1	$120	1	$10	N	Y

Since we are choosing to keep this equipment, we have chosen not to use funding.

If there is any additional hardware or software needed, for unexpected reasons or went unaccounted for, Professor Valaee has offered to fund the purchase.

Appendix C – Validation and Acceptance Tests

To verify our design, our location system must perform with the following requirements:

ID	Project Requirement	Verification Method
1 a.	Software-Defined-Radio	TEST: Start the measurement of the phase changes through a physical NIC. Then use SDR to do the same. SDR should return a more breakdown structure of the internal phase changes within the access points.
1 b.	Access Points Number: 3	TEST: Direct measurement.
2.	Margin of Error	TEST: Direct measurement by checking if error is less than 2 m.
4 a.	Automatic Data Collection	TEST: Start the locating system and check the collected data after collection processes ends and confirm data from all APs has been received during the time of the test
4 b.	Independent-Deployable-Architecture	TEST: To do any direct measurements, none of the existing hardware needs to be changed.
5.	Measure changing angles and position	TEST: Direct measurement.
6.	Using CSI	TEST: Confirm signals sent and received by all APs and the server are strictly CSI signals. This can be done a location not affected by Wi-Fi signals, indoors and outdoors.
7.	Number of packets received by each AP	TEST: Develop a subroutine to count the number of packets received by each AP and compare to the number sent by the transmitter.

Appendix D - Raspberry Pi

To simulate a transmitting device, a Raspberry Pi (Version 3 Model B) was chosen to emulate devices such as cell phones or laptops. A Raspberry Pi is a single board computer with compatibility with networks such as Wi-Fi and Bluetooth.

Throughout the course of the project, members on the team worked to make the Raspberry Pi transmit signals which could be received by the existing software defined radio (SDR) boards (LimeSDR).

This Appendix will list work done towards transmitting a wave, with channel state information (CSI), to the LimeSDR boards.

The following is quick summary (details follow):

1. Able to transmit wave using GNU Radio 3.7
2. Able to download, install, and integrate all GNU Radio components for transmitting wave with prior work except gr-ieee802-11 (IEEE 802.11 a/g/p transceiver for GNU Radio)
3. Investigative work done to learn gr-ieee802-11 only available for GNU Radio 3.8 on Raspberry Pi due to integration with ARM processors
4. Able to download and set up GNU Radio 3.8 on Raspberry Pi, required compiling and building from source
5. Downloaded and installed libraries to execute GNU Radio Companion (GUI) on Raspberry Pi, for example SWIG
6. Found resource to download GNU Radio Companion on Raspberry Pi by cross compiling, unable to use resource due to Raspberry Pi board crashing (https://wiki.gnuradio.org/index.php/Cross_compile_for_Raspberry_Pi)
7. Need to completely install new OS on Pi board was not able to due to time limitations

As shown in the figure below, the team was able to transmit a simple wave using the Raspberry Pi. This was done as a proof of concept that signals could be transmitted via the Pi board. This was done using GNU Radio 3.7.

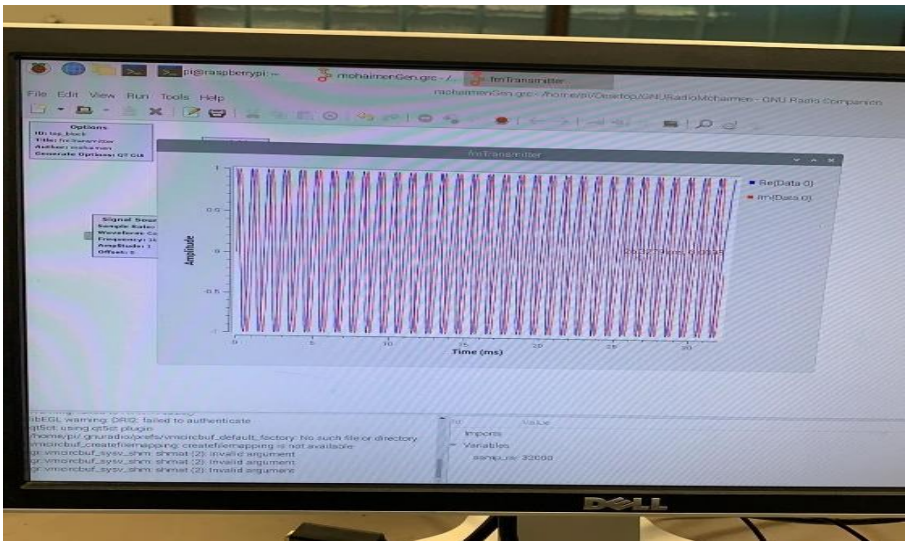


Figure 1. Screenshot of Raspberry Pi board transmitting Sinusoidal Waveform

As mentioned in the summary, while SWIG was installed on the Raspberry Pi to enable GNU Radio Companion (GUI), we needed to cross compile GNU Radio with aarch64-linux-gnu.

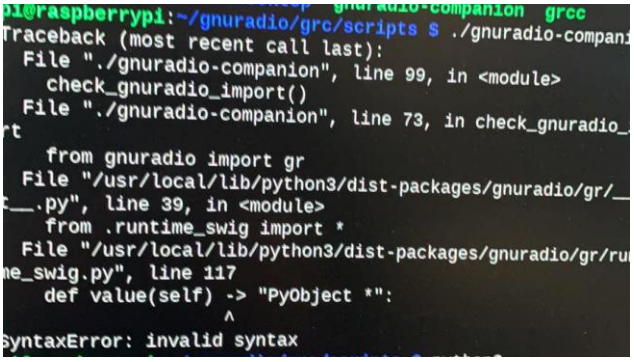


Figure 2. Screenshot of executing GNU Radio Companion, failed due to a lack of cross compilation

After researching issues with Raspberry Pi and installing gr-ieee802-11, we learned this issue stemmed from a C++ Math library issue with ARM Processors, which Raspberry Pi boards used (https://github.com/bastibl/gr-ieee802-11/issues/120, https://github.com/bastibl/gr-ieee802-11/issues/175). After tediously looking through Git commit logs, we learned that the GNU Radio Open Source community had installed a fix – however it was only available for GNU Radio 3.8.

We learned from the Raspbian Package manager that the OS could only currently install GNU Radio 3.7. This meant we had to compile GNU Radio 3.8 from source. While we were able to compile the correct version, which took a total of two days to install, we were not able to get to the step of cross compiling. The Raspberry Pi had crashed and since we were time limited, we could not restart the entire process as there were tasks with higher priorities.

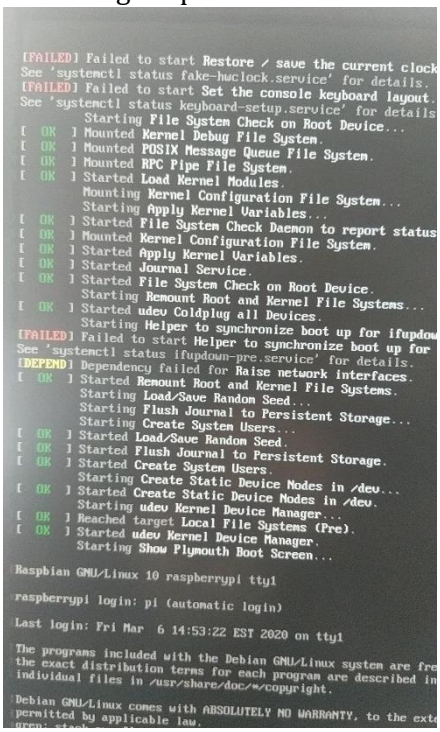


Figure 3. Screenshot of Raspberry Pi unable to boot

Appendix E – Set Up of Receivers and Transmitters in the Lab

The following pictures identify the 4 access points (receivers) and the transmitter within the lab

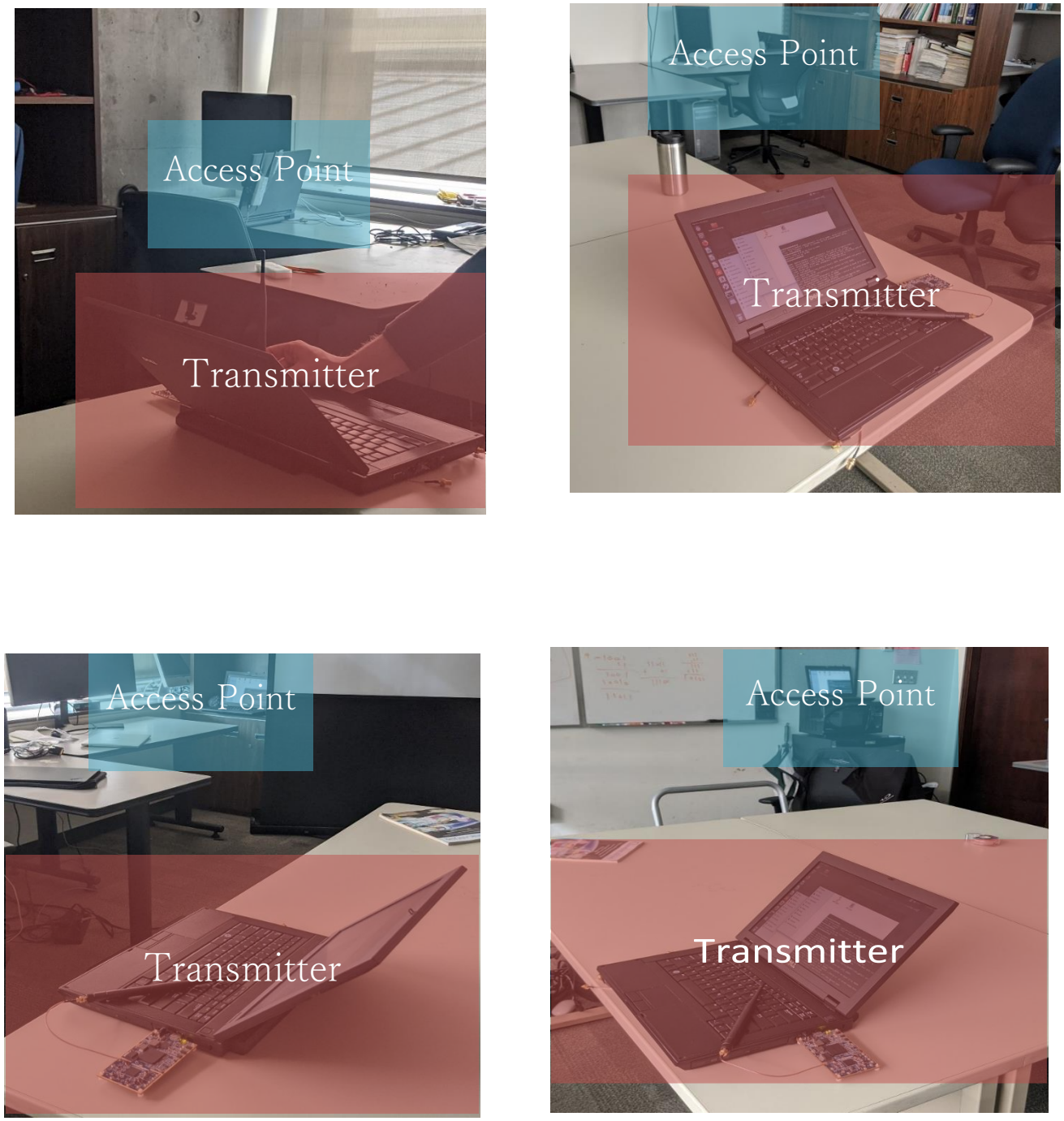


Figure 1. Images of the receivers and transmitters

Appendix F – Data from Access points

Access point 1

```
Using Volk machine: ssse3_64_orc
----------------------------------------------------------------
LimeSuite Source (RX) info

##################
Connecting to device
##################
Devices found: 1
##################
Device list:
Nr.:|0|device:|LimeSDR-USB, media=USB 2.0, module=FX3, addr=1d50:6108,
serial=00090726074E1515
##################
INFO: device_handler::open_device(): no serial number. Using first device in the list.
Use "LimeUtil --find" in terminal to find prefered device serial.
##################
Device number 0 from the list is used.
##################

INFO: device_handler::enable_channels(): MIMO mode set for device number 0.
INFO: device_handler::set_samp_rate(): set sampling rate: 10 MS/s.
INFO: device_handler::set_rf_freq(): RF frequency set [RX]: 2472 MHz.
INFO: device_handler::set_analog_filter(): INFO: device_handler::set_analog_filter():
INFO: device_handler::set_gain(): set gain [RX] CH0: 60 dB.
INFO: device_handler::set_gain(): set gain [RX] CH1: 60 dB.
INFO: device_handler::set_antenna(): CH0 antenna set [RX]: LNAH.
INFO: device_handler::set_antenna(): CH1 antenna set [RX]: LNAH.
INFO: device_handler::calibrate(): INFO: device_handler::calibrate(): INFO:
source_impl::init_stream(): source channel 0 (device nr. 0) stream setup done.
INFO: source_impl::init_stream(): source channel 1 (device nr. 0) stream setup done.
grep2
release2
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
1/1]
grep1
[1,1] packets
release1
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
2/2]
grep2
start common
[18]
TEMP_FILLED
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
3/3]
[[ 1.        +0.j          1.        +0.j          1.        +0.j
   1.        +0.j          1.        +0.j          1.        +0.j
  -0.03318378-0.06879272j -0.04055043-0.08796936j -0.06396719-0.11251349j
  -0.03962846-0.14112845j -0.07151581-0.14848293j -0.06481779-0.19551724j
  -0.07898325-0.2146766j  -0.12874189-0.27033722j -0.15228498-0.28056172j
  -0.19655275-0.33286217j -0.25633261-0.35125136j -0.2810303 -0.32857007j
  -0.32328248-0.35526952j -0.36825883-0.3230302j  -0.36428386-0.30717528j
  -0.43183783-0.25393167j -0.4390929 -0.24617124j -0.41799793-0.25594327j
  -0.45397192-0.27442101j -0.44262683-0.23514903j -0.42817944-0.23575553j
  -0.43351305-0.2374526j  -0.43482709-0.18805394j -0.44256735-0.21050586j
  -0.43723413-0.2081998j  -0.42268747-0.17351012j  1.        +0.j
```

```
  -0.44306999-0.15811843j -0.46065629-0.17953059j -0.44123593-0.18651848j
  -0.44520795-0.13394581j -0.4523693 -0.19045548j -0.4117671 -0.16263479j
  -0.43655935-0.16852628j -0.46406731-0.17093349j -0.45378417-0.13477927j
  -0.4485254 -0.11400923j -0.43663239-0.09003469j -0.44976559-0.07811324j
  -0.46010122-0.07922189j -0.43650964-0.00510101j -0.40961426+0.04674703j
  -0.39182162+0.08720061j -0.34526241+0.07644176j -0.30698538+0.10696229j
  -0.24165332+0.08727103j -0.20689003+0.08257225j -0.1846007 +0.09653105j
  -0.13879037+0.06488773j -0.12937136+0.07121811j -0.11535815+0.02025426j
  -0.09014918+0.05686087j -0.07962444+0.04010059j  1.        +0.j
   1.        +0.j          1.        +0.j          1.        +0.j
   1.        +0.j          1.        +0.j          1.        +0.j
   1.        +0.j          1.        +0.j          1.        +0.j
   1.        +0.j         -0.0410018 -0.0521758j  -0.04638257-0.08615027j
  -0.04396746-0.09791517j -0.04359647-0.11553998j -0.06717829-0.13679574j
  -0.05521963-0.16382536j -0.09924456-0.18075113j -0.08489913-0.21622977j
  -0.12303425-0.24581084j -0.18094167-0.26675239j -0.2200384 -0.283732j
  -0.26902348-0.30700493j -0.3004587 -0.27886951j -0.33289427-0.28152609j
  -0.33801287-0.24131408j -0.36386865-0.21938768j -0.36302257-0.21940222j
  -0.36923832-0.17540514j -0.39743298-0.18924116j -0.39585978-0.15519206j
  -0.38129261-0.17437315j -0.39116901-0.18214472j -0.39197084-0.12466607j
  -0.39455456-0.14487676j -0.41776651-0.16050634j -0.37396735-0.14550498j
   1.        +0.j         -0.38762799-0.12518202j -0.36878553-0.14021111j
  -0.37855148-0.14089166j -0.395605  -0.12334j    -0.36171931-0.11445225j
  -0.35333979-0.11579772j -0.34354514-0.08864617j -0.41371843-0.08223445j
  -0.37136975-0.0958467j  -0.35499227-0.08257207j -0.36703813-0.06533527j
  -0.35543963-0.02876419j -0.35142401-0.03282484j -0.35701007+0.01306679j
  -0.30840841+0.03449601j -0.27415451+0.07879004j -0.27239215+0.06268968j
  -0.22372766+0.09907051j -0.16550389+0.05266673j -0.15668938+0.07619157j
  -0.12401906+0.0676765j  -0.09435624+0.03406971j -0.09198502+0.02947473j
  -0.09160943+0.05849718j -0.07553883+0.04619421j -0.03894448+0.01236591j
   1.        +0.j          1.        +0.j          1.        +0.j
   1.        +0.j          1.        +0.j        ]]
CSI FRAME
[[-0.45397192-0.27442101j -0.39743298-0.18924116j]]
start calculation
OUR STUFF
GRID
[-1.57079633 -1.55500943 -1.53922253 -1.52343563 -1.50764874 -1.49186184
 -1.47607494 -1.46028804 -1.44450114 -1.42871425 -1.41292735 -1.39714045
 -1.38135355 -1.36556666 -1.34977976 -1.33399286 -1.31820596 -1.30241906
 -1.28663217 -1.27084527 -1.25505837 -1.23927147 -1.22348458 -1.20769768
 -1.19191078 -1.17612388 -1.16033699 -1.14455009 -1.12876319 -1.11297629
 -1.09718939 -1.0814025  -1.0656156  -1.0498287  -1.0340418  -1.01825491
 -1.00246801 -0.98668111 -0.97089421 -0.95510731 -0.93932042 -0.92353352
 -0.90774662 -0.89195972 -0.87617283 -0.86038593 -0.84459903 -0.82881213
 -0.81302523 -0.79723834 -0.78145144 -0.76566454 -0.74987764 -0.73409075
 -0.71830385 -0.70251695 -0.68673005 -0.67094315 -0.65515626 -0.63936936
 -0.62358246 -0.60779556 -0.59200867 -0.57622177 -0.56043487 -0.54464797
 -0.52886107 -0.51307418 -0.49728728 -0.48150038 -0.46571348 -0.44992659
 -0.43413969 -0.41835279 -0.40256589 -0.386779   -0.3709921  -0.3552052
 -0.3394183  -0.3236314  -0.30784451 -0.29205761 -0.27627071 -0.26048381
 -0.24469692 -0.22891002 -0.21312312 -0.19733622 -0.18154932 -0.16576243
 -0.14997553 -0.13418863 -0.11840173 -0.10261484 -0.08682794 -0.07104104
 -0.05525414 -0.03946724 -0.02368035 -0.00789345  0.00789345  0.02368035
  0.03946724  0.05525414  0.07104104  0.08682794  0.10261484  0.11840173
  0.13418863  0.14997553  0.16576243  0.18154932  0.19733622  0.21312312
  0.22891002  0.24469692  0.26048381  0.27627071  0.29205761  0.30784451
  0.3236314   0.3394183   0.3552052   0.3709921   0.386779    0.40256589
  0.41835279  0.43413969  0.44992659  0.46571348  0.48150038  0.49728728
  0.51307418  0.52886107  0.54464797  0.56043487  0.57622177  0.59200867
  0.60779556  0.62358246  0.63936936  0.65515626  0.67094315  0.68673005
  0.70251695  0.71830385  0.73409075  0.74987764  0.76566454  0.78145144
  0.79723834  0.81302523  0.82881213  0.84459903  0.86038593  0.87617283
```

```
  0.89195972  0.90774662  0.92353352  0.93932042  0.95510731  0.97089421
  0.98668111  1.00246801  1.01825491  1.0340418   1.0498287   1.0656156
  1.0814025   1.09718939  1.11297629  1.12876319  1.14455009  1.16033699
  1.17612388  1.19191078  1.20769768  1.22348458  1.23927147  1.25505837
  1.27084527  1.28663217  1.30241906  1.31820596  1.33399286  1.34977976
  1.36556666  1.38135355  1.39714045  1.41292735  1.42871425  1.44450114
  1.46028804  1.47607494  1.49186184  1.50764874  1.52343563  1.53922253
  1.55500943  1.57079633]
AoA is at: -2.261306532663317 degrees
try to plot
done
end calculation
end common
release2

##################
INFO: device_handler::close_device(): Disconnected from device number 0.
##################
```

Access point 2

```
Using Volk machine: ssse3_64_orc
----------------------------------------------------------------
LimeSuite Source (RX) info

##################
Connecting to device
##################
Devices found: 1
##################
Device list:
Nr.:|0|device:|LimeSDR-USB, media=USB 2.0, module=FX3, addr=1d50:6108,
serial=00090726074E1515
##################
INFO: device_handler::open_device(): no serial number. Using first device in the list.
Use "LimeUtil --find" in terminal to find prefered device serial.
##################
Device number 0 from the list is used.
##################

INFO: device_handler::enable_channels(): MIMO mode set for device number 0.
INFO: device_handler::set_samp_rate(): set sampling rate: 10 MS/s.
INFO: device_handler::set_rf_freq(): RF frequency set [RX]: 2472 MHz.
INFO: device_handler::set_analog_filter(): INFO: device_handler::set_analog_filter():
INFO: device_handler::set_gain(): set gain [RX] CH0: 60 dB.
INFO: device_handler::set_gain(): set gain [RX] CH1: 60 dB.
INFO: device_handler::set_antenna(): CH0 antenna set [RX]: LNAH.
INFO: device_handler::set_antenna(): CH1 antenna set [RX]: LNAH.
INFO: device_handler::calibrate(): INFO: device_handler::calibrate(): INFO:
source_impl::init_stream(): source channel 0 (device nr. 0) stream setup done.
INFO: source_impl::init_stream(): source channel 1 (device nr. 0) stream setup done.
grep1
[1,0] packets
release1
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
1/1]
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
2/2]
grep1
[1,0] packets
release1
grep2
```

```
release2
Header: CB 1C 21
OFDM PHY Header: RATE 0x0B LENGTH 2278 RESERVED 0 TAIL 8 --> Parity check failed. RATE
not supported. Length invalid.  [Correct: 2/3]
grep2
release2
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
3/4]
grep2
release2
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
4/5]
grep1
[1,1] packets
release1
Header: 9C FD BB
OFDM PHY Header: RATE 0x0C LENGTH 4076 RESERVED 1 TAIL 46 --> Parity check failed.
RATE not supported. Length invalid.  [Correct: 4/6]
grep2
start common
Header: EF CC 09
OFDM PHY Header: RATE 0x0F LENGTH 3687 RESERVED 0 TAIL 2 --> n_ofdm_syms 820 [Correct:
5/7]
[18]
TEMP_FILLED
[[1.        +0.j         1.        +0.j         1.        +0.j
  1.        +0.j         1.        +0.j         1.        +0.j
  0.02502792-0.01754536j 0.05324674-0.02319369j 0.04904303-0.02623287j
  0.06471138-0.02200188j 0.07046349-0.02110075j 0.10517918-0.02776377j
  0.09712557-0.02685372j 0.13367787-0.04625028j 0.11379246-0.02920156j
  0.16114265-0.0161086j  0.20569056-0.0447535j  0.18024954-0.05281529j
  0.18792669-0.07708946j 0.18975183-0.08534939j 0.1820605 -0.08370092j
  0.20015559-0.0875971j  0.20190521-0.09274183j 0.20232715-0.0975506j
  0.20217139-0.08193624j 0.19025546-0.08011659j 0.20730302-0.07721317j
  0.21942225-0.10693344j 0.21721148-0.06879685j 0.19881557-0.05269167j
  0.21568632-0.0320236j  0.20948079-0.02247082j 1.        +0.j
  0.24417008-0.02073156j 0.24936655-0.00474834j 0.22772081-0.0090449j
  0.25155446-0.0170189j  0.24293014+0.00673996j 0.2489585 -0.00787383j
  0.24965009-0.00313993j 0.26965126+0.05035759j 0.26967448-0.00270695j
  0.27670723+0.01598806j 0.26667812+0.02140361j 0.27728787+0.01081964j
  0.26509041+0.000631j   0.27706945-0.0151807j  0.25991908-0.03800514j
  0.21369815-0.03893813j 0.1859743 -0.03852527j 0.1663989 -0.03238609j
  0.13394463-0.07331171j 0.13434528-0.03412919j 0.10957972-0.04556735j
  0.08528599-0.00333503j 0.07656209-0.03693504j 0.04687168-0.01473766j
  0.04412013-0.00027004j 0.04860766-0.00511554j 1.        +0.j
  1.        +0.j         1.        +0.j         1.        +0.j
  1.        +0.j         1.        +0.j         1.        +0.j
  1.        +0.j         1.        +0.j         1.        +0.j
  1.        +0.j         0.04093564-0.00382751j 0.0495878 +0.01125159j
  0.08095558-0.0242326j  0.08126359+0.02218871j 0.0864298 +0.05746982j
  0.10659153+0.05852195j 0.12275492+0.0354574j  0.16109222+0.07586928j
  0.17345452+0.05663243j 0.2037981 +0.08083975j 0.22290699+0.08820186j
  0.28334725+0.06936055j 0.2723583 +0.08339684j 0.3080202 +0.06527671j
  0.31633872+0.06078569j 0.31453115-0.00163572j 0.32153893+0.02420302j
  0.30851647+0.05970414j 0.32013732+0.05918991j 0.35330787+0.02162585j
  0.32183748+0.05250032j 0.32904831+0.07371707j 0.31917858+0.06820844j
  0.30695623+0.07914524j 0.32333469+0.0755944j  0.31428596+0.10900494j
  1.        +0.j         0.31432652+0.09799352j 0.31540823+0.11673859j
  0.30297738+0.10654442j 0.31992143+0.10495591j 0.30393353+0.12469724j
  0.31678575+0.14610165j 0.32200316+0.13187039j 0.31000295+0.15336607j
  0.30171782+0.16497j    0.30200195+0.16732003j 0.30495322+0.1398606j
```

```
  0.31050444+0.14172319j 0.29399198+0.13552135j 0.32687172+0.11652666j
  0.27145076+0.0597166j  0.26862618+0.0658114j  0.22855443+0.02229557j
  0.21827695+0.02655187j 0.1907087 +0.0136871j  0.17979805+0.01485206j
  0.12761864+0.02754357j 0.10659717+0.04106182j 0.10390486+0.02718505j
  0.09074229+0.02720314j 0.05455039+0.02125419j 0.03854245+0.00332428j
  1.        +0.j         1.        +0.j         1.        +0.j
  1.        +0.j         1.        +0.j        ]]
CSI FRAME
[[0.20217139-0.08193624j 0.32013732+0.05918991j]]
start calculation
OUR STUFF
GRID
[-1.57079633 -1.55500943 -1.53922253 -1.52343563 -1.50764874 -1.49186184
 -1.47607494 -1.46028804 -1.44450114 -1.42871425 -1.41292735 -1.39714045
 -1.38135355 -1.36556666 -1.34977976 -1.33399286 -1.31820596 -1.30241906
 -1.28663217 -1.27084527 -1.25505837 -1.23927147 -1.22348458 -1.20769768
 -1.19191078 -1.17612388 -1.16033699 -1.14455009 -1.12876319 -1.11297629
 -1.09718939 -1.0814025  -1.0656156  -1.0498287  -1.0340418  -1.01825491
 -1.00246801 -0.98668111 -0.97089421 -0.95510731 -0.93932042 -0.92353352
 -0.90774662 -0.89195972 -0.87617283 -0.86038593 -0.84459903 -0.82881213
 -0.81302523 -0.79723834 -0.78145144 -0.76566454 -0.74987764 -0.73409075
 -0.71830385 -0.70251695 -0.68673005 -0.67094315 -0.65515626 -0.63936936
 -0.62358246 -0.60779556 -0.59200867 -0.57622177 -0.56043487 -0.54464797
 -0.52886107 -0.51307418 -0.49728728 -0.48150038 -0.46571348 -0.44992659
 -0.43413969 -0.41835279 -0.40256589 -0.386779   -0.3709921  -0.3552052
 -0.3394183  -0.3236314  -0.30784451 -0.29205761 -0.27627071 -0.26048381
 -0.24469692 -0.22891002 -0.21312312 -0.19733622 -0.18154932 -0.16576243
 -0.14997553 -0.13418863 -0.11840173 -0.10261484 -0.08682794 -0.07104104
 -0.05525414 -0.03946724 -0.02368035 -0.00789345  0.00789345  0.02368035
  0.03946724  0.05525414  0.07104104  0.08682794  0.10261484  0.11840173
  0.13418863  0.14997553  0.16576243  0.18154932  0.19733622  0.21312312
  0.22891002  0.24469692  0.26048381  0.27627071  0.29205761  0.30784451
  0.3236314   0.3394183   0.3552052   0.3709921   0.386779    0.40256589
  0.41835279  0.43413969  0.44992659  0.46571348  0.48150038  0.49728728
  0.51307418  0.52886107  0.54464797  0.56043487  0.57622177  0.59200867
  0.60779556  0.62358246  0.63936936  0.65515626  0.67094315  0.68673005
  0.70251695  0.71830385  0.73409075  0.74987764  0.76566454  0.78145144
  0.79723834  0.81302523  0.82881213  0.84459903  0.86038593  0.87617283
  0.89195972  0.90774662  0.92353352  0.93932042  0.95510731  0.97089421
  0.98668111  1.00246801  1.01825491  1.0340418   1.0498287   1.0656156
  1.0814025   1.09718939  1.11297629  1.12876319  1.14455009  1.16033699
  1.17612388  1.19191078  1.20769768  1.22348458  1.23927147  1.25505837
  1.27084527  1.28663217  1.30241906  1.31820596  1.33399286  1.34977976
  1.36556666  1.38135355  1.39714045  1.41292735  1.42871425  1.44450114
  1.46028804  1.47607494  1.49186184  1.50764874  1.52343563  1.53922253
  1.55500943  1.57079633]
AoA is at: 10.402010050251262 degrees
try to plot
done
end calculation
end common
release2
grep2
release2
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
6/8]
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
7/9]
grep1
[2,2] packets
release1
```

```
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
8/10]
grep2
start common
[18]
TEMP_FILLED
[[ 1.        +0.00000000e+00j  1.        +0.00000000e+00j
   1.        +0.00000000e+00j  1.        +0.00000000e+00j
   1.        +0.00000000e+00j  1.        +0.00000000e+00j
   0.02502792-1.75453555e-02j  0.05324674-2.31936909e-02j
   0.04904303-2.62328684e-02j  0.06471138-2.20018774e-02j
   0.07046349-2.11007521e-02j  0.10517918-2.77637728e-02j
   0.09712557-2.68537179e-02j  0.13367787-4.62502837e-02j
   0.11379246-2.92015634e-02j  0.16114265-1.61085967e-02j
   0.20569056-4.47534993e-02j  0.18024954-5.28152883e-02j
   0.18792669-7.70894587e-02j  0.18975183-8.53493884e-02j
   0.1820605 -8.37009177e-02j  0.20015559-8.75971019e-02j
   0.20190521-9.27418321e-02j  0.20232715-9.75506008e-02j
   0.20217139-8.19362402e-02j  0.19025546-8.01165923e-02j
   0.20730302-7.72131681e-02j  0.21942225-1.06933445e-01j
   0.21721148-6.87968507e-02j  0.19881557-5.26916683e-02j
   0.21568632-3.20235975e-02j  0.20948079-2.24708207e-02j
   1.        +0.00000000e+00j  0.24417008-2.07315609e-02j
   0.24936655-4.74834489e-03j  0.22772081-9.04490054e-03j
   0.25155446-1.70188993e-02j  0.24293014+6.73996191e-03j
   0.2489585 -7.87382759e-03j  0.24965009-3.13992519e-03j
   0.26965126+5.03575876e-02j  0.26967448-2.70694960e-03j
   0.27670723+1.59880631e-02j  0.26667812+2.14036144e-02j
   0.27728787+1.08196437e-02j  0.26509041+6.30996190e-04j
   0.27706945-1.51807014e-02j  0.25991908-3.80051360e-02j
   0.21369815-3.89381275e-02j  0.1859743 -3.85252684e-02j
   0.1663989 -3.23860943e-02j  0.13394463-7.33117089e-02j
   0.13434528-3.41291949e-02j  0.10957972-4.55673486e-02j
   0.08528599-3.33502889e-03j  0.07656209-3.69350426e-02j
   0.04687168-1.47376582e-02j  0.04412013-2.70043034e-04j
   0.04860766-5.11554442e-03j  1.        +0.00000000e+00j
   1.        +0.00000000e+00j  1.        +0.00000000e+00j
   1.        +0.00000000e+00j  1.        +0.00000000e+00j
   1.        +0.00000000e+00j  1.        +0.00000000e+00j
   1.        +0.00000000e+00j  1.        +0.00000000e+00j
   1.        +0.00000000e+00j  1.        +0.00000000e+00j
   0.04093564-3.82751226e-03j  0.0495878 +1.12515911e-02j
   0.08095558-2.42325962e-02j  0.08126359+2.21887100e-02j
   0.0864298 +5.74698150e-02j  0.10659153+5.85219450e-02j
   0.12275492+3.54573950e-02j  0.16109222+7.58692771e-02j
   0.17345452+5.66324256e-02j  0.2037981 +8.08397532e-02j
   0.22290699+8.82018581e-02j  0.28334725+6.93605542e-02j
   0.2723583 +8.33968371e-02j  0.3080202 +6.52767122e-02j
   0.31633872+6.07856885e-02j  0.31453115-1.63571723e-03j
   0.32153893+2.42030174e-02j  0.30851647+5.97041361e-02j
   0.32013732+5.91899082e-02j  0.35330787+2.16258503e-02j
   0.32183748+5.25003150e-02j  0.32904831+7.37170726e-02j
   0.31917858+6.82084411e-02j  0.30695623+7.91452378e-02j
   0.32333469+7.55943954e-02j  0.31428596+1.09004937e-01j
   1.        +0.00000000e+00j  0.31432652+9.79935229e-02j
   0.31540823+1.16738588e-01j  0.30297738+1.06544420e-01j
   0.31992143+1.04955912e-01j  0.30393353+1.24697238e-01j
   0.31678575+1.46101654e-01j  0.32200316+1.31870389e-01j
   0.31000295+1.53366074e-01j  0.30171782+1.64969996e-01j
   0.30200195+1.67320028e-01j  0.30495322+1.39860600e-01j
   0.31050444+1.41723186e-01j  0.29399198+1.35521352e-01j
   0.32687172+1.16526656e-01j  0.27145076+5.97166009e-02j
```

```
  0.26862618+6.58114031e-02j  0.22855443+2.22955719e-02j
  0.21827695+2.65518669e-02j  0.1907087 +1.36871021e-02j
  0.17979805+1.48520647e-02j  0.12761864+2.75435690e-02j
  0.10659717+4.10618186e-02j  0.10390486+2.71850452e-02j
  0.09074229+2.72031352e-02j  0.05455039+2.12541856e-02j
  0.03854245+3.32427956e-03j  1.        +0.00000000e+00j
  1.        +0.00000000e+00j  1.        +0.00000000e+00j
  1.        +0.00000000e+00j  1.        +0.00000000e+00j]
[ 1.        +0.00000000e+00j  1.        +0.00000000e+00j
  1.        +0.00000000e+00j  1.        +0.00000000e+00j
  1.        +0.00000000e+00j  1.        +0.00000000e+00j
 -0.01852653+1.08782165e-02j -0.01752404+3.45410630e-02j
 -0.02517906+3.46125588e-02j -0.04233583+4.49371859e-02j
 -0.03279149+6.54523745e-02j -0.04040354+5.31232059e-02j
 -0.04343313+8.17134827e-02j -0.05889842+9.39890295e-02j
 -0.03137412+1.11707479e-01j -0.03079888+1.22326456e-01j
 -0.01647624+1.65690094e-01j -0.01711604+1.70488060e-01j
  0.02305486+1.74940124e-01j  0.01555228+1.92280427e-01j
  0.05189312+2.26910710e-01j  0.06040259+2.20772982e-01j
  0.05098821+1.91857561e-01j  0.10365467+2.00085193e-01j
  0.09719279+1.90452904e-01j  0.09784332+1.80213869e-01j
  0.11745837+1.77194625e-01j  0.1072951 +1.83346197e-01j
  0.12493315+1.76768422e-01j  0.15553103+1.84215024e-01j
  0.13998947+1.78673908e-01j  0.15221743+1.81847125e-01j
  1.        +0.00000000e+00j  0.15599295+1.63106486e-01j
  0.16476744+1.79658562e-01j  0.18440641+1.86968625e-01j
  0.16208443+1.55312523e-01j  0.18133892+1.46704867e-01j
  0.20391604+1.44436195e-01j  0.23592591+1.38289496e-01j
  0.17592597+1.25268742e-01j  0.23408437+1.31007403e-01j
  0.25215873+1.17857270e-01j  0.25079155+9.59571078e-02j
  0.21945575+7.58208260e-02j  0.24203749+6.73998147e-02j
  0.24248154+2.67187078e-02j  0.23830287-3.10492683e-02j
  0.22206119-3.39902863e-02j  0.16433778-4.82730083e-02j
  0.17632331-4.82306443e-02j  0.18422657-5.62389120e-02j
  0.09752978-6.87451512e-02j  0.10483387-6.90317899e-02j
  0.08102996-2.91149467e-02j  0.05515411-2.38280036e-02j
  0.04527949-8.66014510e-03j  0.05804988-2.71479301e-02j
  0.03796351-1.99366435e-02j  1.        +0.00000000e+00j
  1.        +0.00000000e+00j  1.        +0.00000000e+00j
  1.        +0.00000000e+00j  1.        +0.00000000e+00j
  1.        +0.00000000e+00j  1.        +0.00000000e+00j
  1.        +0.00000000e+00j  1.        +0.00000000e+00j
  0.01877958+7.11524859e-02j  0.0468221 +9.57961306e-02j
  0.06037822+1.32926837e-01j  0.04925836+1.17124364e-01j
  0.06496241+1.65769428e-01j  0.0905882 +1.97883114e-01j
  0.10226015+2.46886611e-01j  0.14641313+2.79027700e-01j
  0.19792524+3.00463557e-01j  0.23916966+3.46188307e-01j
  0.30021268+3.21109235e-01j  0.37498099+3.31109136e-01j
  0.4276644 +2.88239330e-01j  0.49885023+2.48261750e-01j
  0.52773285+2.35256761e-01j  0.54350507+1.97225451e-01j
  0.53121597+1.56286299e-01j  0.56706595+1.34104893e-01j
  0.56542313+1.12659939e-01j  0.58763367+9.04397517e-02j
  0.58974385+4.92904484e-02j  0.566185  +4.56551835e-02j
  0.59398502+1.15862936e-02j  0.59115738-1.59291774e-02j
  0.59094208+4.93966043e-04j  0.59821165-5.32498285e-02j
  1.        +0.00000000e+00j  0.56578207-9.72551331e-02j
  0.62704241-1.21453106e-01j  0.59731174-1.14962950e-01j
  0.57876402-1.65264755e-01j  0.55429578-2.06571162e-01j
  0.59769922-2.08878219e-01j  0.58067238-2.06698120e-01j
  0.52017677-1.98898062e-01j  0.51330894-2.38468021e-01j
  0.53339791-3.11659604e-01j  0.49015802-3.41467679e-01j
  0.46315241-4.04142916e-01j  0.37735868-3.84941638e-01j
```

```
   0.38113576-4.47842658e-01j  0.24578071-4.36498463e-01j
   0.22382936-4.35428172e-01j  0.13919096-3.85808140e-01j
   0.10862809-3.54847550e-01j  0.07400056-3.05613667e-01j
   0.04657095-2.65083492e-01j  0.05223424-2.51946807e-01j
   0.0054487 -2.25340873e-01j  0.02167384-1.64923981e-01j
   0.00667816-1.32368490e-01j  0.0011328 -1.33959800e-01j
   0.03770965-9.61219743e-02j  1.        +0.00000000e+00j
   1.        +0.00000000e+00j  1.        +0.00000000e+00j
   1.        +0.00000000e+00j  1.        +0.00000000e+00j]]
CSI FRAME
[[0.20217139-0.08193624j 0.32013732+0.05918991j]
 [0.09719279+0.1904529j  0.56542313+0.11265994j]]
start calculation
OUR STUFF
GRID
[-1.57079633 -1.55500943 -1.53922253 -1.52343563 -1.50764874 -1.49186184
 -1.47607494 -1.46028804 -1.44450114 -1.42871425 -1.41292735 -1.39714045
 -1.38135355 -1.36556666 -1.34977976 -1.33399286 -1.31820596 -1.30241906
 -1.28663217 -1.27084527 -1.25505837 -1.23927147 -1.22348458 -1.20769768
 -1.19191078 -1.17612388 -1.16033699 -1.14455009 -1.12876319 -1.11297629
 -1.09718939 -1.0814025  -1.0656156  -1.0498287  -1.0340418  -1.01825491
 -1.00246801 -0.98668111 -0.97089421 -0.95510731 -0.93932042 -0.92353352
 -0.90774662 -0.89195972 -0.87617283 -0.86038593 -0.84459903 -0.82881213
 -0.81302523 -0.79723834 -0.78145144 -0.76566454 -0.74987764 -0.73409075
 -0.71830385 -0.70251695 -0.68673005 -0.67094315 -0.65515626 -0.63936936
 -0.62358246 -0.60779556 -0.59200867 -0.57622177 -0.56043487 -0.54464797
 -0.52886107 -0.51307418 -0.49728728 -0.48150038 -0.46571348 -0.44992659
 -0.43413969 -0.41835279 -0.40256589 -0.386779   -0.3709921  -0.3552052
 -0.3394183  -0.3236314  -0.30784451 -0.29205761 -0.27627071 -0.26048381
 -0.24469692 -0.22891002 -0.21312312 -0.19733622 -0.18154932 -0.16576243
 -0.14997553 -0.13418863 -0.11840173 -0.10261484 -0.08682794 -0.07104104
 -0.05525414 -0.03946724 -0.02368035 -0.00789345  0.00789345  0.02368035
  0.03946724  0.05525414  0.07104104  0.08682794  0.10261484  0.11840173
  0.13418863  0.14997553  0.16576243  0.18154932  0.19733622  0.21312312
  0.22891002  0.24469692  0.26048381  0.27627071  0.29205761  0.30784451
  0.3236314   0.3394183   0.3552052   0.3709921   0.386779    0.40256589
  0.41835279  0.43413969  0.44992659  0.46571348  0.48150038  0.49728728
  0.51307418  0.52886107  0.54464797  0.56043487  0.57622177  0.59200867
  0.60779556  0.62358246  0.63936936  0.65515626  0.67094315  0.68673005
  0.70251695  0.71830385  0.73409075  0.74987764  0.76566454  0.78145144
  0.79723834  0.81302523  0.82881213  0.84459903  0.86038593  0.87617283
  0.89195972  0.90774662  0.92353352  0.93932042  0.95510731  0.97089421
  0.98668111  1.00246801  1.01825491  1.0340418   1.0498287   1.0656156
  1.0814025   1.09718939  1.11297629  1.12876319  1.14455009  1.16033699
  1.17612388  1.19191078  1.20769768  1.22348458  1.23927147  1.25505837
  1.27084527  1.28663217  1.30241906  1.31820596  1.33399286  1.34977976
  1.36556666  1.38135355  1.39714045  1.41292735  1.42871425  1.44450114
  1.46028804  1.47607494  1.49186184  1.50764874  1.52343563  1.53922253
  1.55500943  1.57079633]
AoA is at: -7.688442211055276 degrees
try to plot
done
end calculation
end common
release2

##################
INFO: device_handler::close_device(): Disconnected from device number 0.
##################
```

Access point 3

```
Using Volk machine: ssse3_64_orc
------------------------------------------------------------
```

```
LimeSuite Source (RX) info

##################
Connecting to device
##################
Devices found: 1
##################
Device list:
Nr.:|0|device:|LimeSDR-USB, media=USB 2.0, module=FX3, addr=1d50:6108,
serial=00090726074E1515
##################
INFO: device_handler::open_device(): no serial number. Using first device in the list.
Use "LimeUtil --find" in terminal to find prefered device serial.
##################
Device number 0 from the list is used.
##################

INFO: device_handler::enable_channels(): MIMO mode set for device number 0.
INFO: device_handler::set_samp_rate(): set sampling rate: 10 MS/s.
INFO: device_handler::set_rf_freq(): RF frequency set [RX]: 2472 MHz.
INFO: device_handler::set_analog_filter(): INFO: device_handler::set_analog_filter():
INFO: device_handler::set_gain(): set gain [RX] CH0: 60 dB.
INFO: device_handler::set_gain(): set gain [RX] CH1: 60 dB.
INFO: device_handler::set_antenna(): CH0 antenna set [RX]: LNAH.
INFO: device_handler::set_antenna(): CH1 antenna set [RX]: LNAH.
INFO: device_handler::calibrate(): INFO: device_handler::calibrate(): INFO:
source_impl::init_stream(): source channel 0 (device nr. 0) stream setup done.
INFO: source_impl::init_stream(): source channel 1 (device nr. 0) stream setup done.
grep2
release2
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
1/1]
grep1
[1,1] packets
release1
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
2/2]
grep1
[1,1] packets
release1
Header: BD DC 34
OFDM PHY Header: RATE 0x0D LENGTH 1765 RESERVED 1 TAIL 13 --> Parity check failed.
RATE not supported. Length invalid.  [Correct: 2/3]
grep1
[1,1] packets
release1
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
3/4]
grep2
start common
[18]
TEMP_FILLED
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
4/5]
[[ 1.00000000e+00+0.j          1.00000000e+00+0.j
   1.00000000e+00+0.j          1.00000000e+00+0.j
   1.00000000e+00+0.j          1.00000000e+00+0.j
   1.13100119e-01+0.0830808j   1.20081879e-01+0.10076796j
   1.18162878e-01+0.0974371j   1.31830186e-01+0.10739402j
```

```
   1.03962846e-01+0.11066671j  1.06222540e-01+0.10836194j
   1.11371703e-01+0.10500407j  9.93776917e-02+0.14501326j
   1.04736522e-01+0.11317438j  9.66737047e-02+0.10058969j
   9.29509550e-02+0.11946456j  9.14477035e-02+0.13233547j
   8.51119682e-02+0.13742633j  9.55592841e-02+0.11182432j
   1.01674214e-01+0.14726484j  4.83208708e-02+0.15262201j
   8.58644024e-02+0.13353038j  6.94536269e-02+0.13344239j
   6.80369288e-02+0.12880141j  4.82346602e-02+0.15280014j
   5.57407811e-02+0.13627827j  5.91831766e-02+0.16184391j
   4.82643768e-02+0.15769516j  3.49294841e-02+0.13401599j
   2.35904232e-02+0.14065284j  5.48273660e-02+0.15903535j
   1.00000000e+00+0.j          3.73775437e-02+0.15449344j
   3.18435207e-02+0.16264023j  6.45992253e-03+0.17437364j
   2.22650878e-02+0.17391086j  3.70756425e-02+0.1865575j
  -8.07479396e-03+0.18540481j -4.36070189e-03+0.19345671j
   1.35777714e-02+0.14869902j  5.65548148e-03+0.19224216j
   3.13448044e-03+0.17545915j -3.20355361e-03+0.17350191j
  -2.61297310e-03+0.17245741j -1.36909611e-03+0.17400113j
   4.40618163e-03+0.18067254j -1.71865779e-03+0.15522403j
  -2.46218722e-02+0.19896144j -4.43789884e-02+0.13572706j
  -3.81782884e-04+0.18496028j -1.32796327e-02+0.1609866j
  -3.46162021e-02+0.15805638j -2.14860570e-02+0.18702278j
  -3.36788446e-02+0.17762306j -2.47416589e-02+0.17063124j
  -3.25925909e-02+0.15406927j -2.91311145e-02+0.13696766j
  -3.34484801e-02+0.13282952j  1.00000000e+00+0.j
   1.00000000e+00+0.j          1.00000000e+00+0.j
   1.00000000e+00+0.j          1.00000000e+00+0.j
   1.00000000e+00+0.j          1.00000000e+00+0.j
   1.00000000e+00+0.j          1.00000000e+00+0.j
   4.66231480e-02-0.11549701j  5.31337187e-02-0.10944202j
   5.43580353e-02-0.11446355j  6.07416034e-02-0.13260426j
   7.85999000e-02-0.14185511j  8.55216756e-02-0.11068997j
   7.47249797e-02-0.11141409j  6.94812089e-02-0.13823833j
   7.37054050e-02-0.12624782j  7.38445222e-02-0.14266431j
   8.65149200e-02-0.13731661j  9.67278853e-02-0.12672442j
   8.45587924e-02-0.14126502j  6.30623922e-02-0.13489126j
   8.69584680e-02-0.11629687j  1.00484483e-01-0.12259994j
   1.02843478e-01-0.13634107j  1.24305077e-01-0.12716264j
   9.95896682e-02-0.13409209j  8.96469206e-02-0.11880751j
   9.80385244e-02-0.11465932j  9.48628932e-02-0.1166728j
   1.07294835e-01-0.11682999j  8.70007724e-02-0.13057899j
   1.17794625e-01-0.12615468j  9.35891271e-02-0.09958196j
   1.00000000e+00+0.j          1.31588280e-01-0.11928421j
   9.97034460e-02-0.07082816j  1.28484905e-01-0.0971174j
   1.09988026e-01-0.08721123j  1.20360054e-01-0.08418572j
   1.08658254e-01-0.09013634j  1.15960896e-01-0.09780672j
   1.31232515e-01-0.07794113j  1.24548309e-01-0.06882032j
   9.66057256e-02-0.0547228j   1.35806441e-01-0.0708521j
   1.38315752e-01-0.06687567j  1.30057022e-01-0.04850531j
   1.15742713e-01-0.05866267j  1.53896928e-01-0.05531424j
   1.23762339e-01-0.06602737j  1.02564298e-01-0.05565438j
   1.30456731e-01-0.04410369j  1.36331409e-01-0.02904276j
   1.47053882e-01-0.03191298j  1.08759582e-01-0.03167722j
   1.09407715e-01-0.01664606j  1.15009896e-01-0.01368954j
   1.06207028e-01-0.02020986j  1.33869439e-01+0.00670278j
   1.18916921e-01+0.01207057j  1.00000000e+00+0.j
   1.00000000e+00+0.j          1.00000000e+00+0.j
   1.00000000e+00+0.j          1.00000000e+00+0.j        ]]
CSI FRAME
[[0.06803693+0.12880141j 0.09958967-0.13409209j]]
start calculation
OUR STUFF
```

```
GRID
[-1.57079633 -1.55500943 -1.53922253 -1.52343563 -1.50764874 -1.49186184
 -1.47607494 -1.46028804 -1.44450114 -1.42871425 -1.41292735 -1.39714045
 -1.38135355 -1.36556666 -1.34977976 -1.33399286 -1.31820596 -1.30241906
 -1.28663217 -1.27084527 -1.25505837 -1.23927147 -1.22348458 -1.20769768
 -1.19191078 -1.17612388 -1.16033699 -1.14455009 -1.12876319 -1.11297629
 -1.09718939 -1.0814025  -1.0656156  -1.0498287  -1.0340418  -1.01825491
 -1.00246801 -0.98668111 -0.97089421 -0.95510731 -0.93932042 -0.92353352
 -0.90774662 -0.89195972 -0.87617283 -0.86038593 -0.84459903 -0.82881213
 -0.81302523 -0.79723834 -0.78145144 -0.76566454 -0.74987764 -0.73409075
 -0.71830385 -0.70251695 -0.68673005 -0.67094315 -0.65515626 -0.63936936
 -0.62358246 -0.60779556 -0.59200867 -0.57622177 -0.56043487 -0.54464797
 -0.52886107 -0.51307418 -0.49728728 -0.48150038 -0.46571348 -0.44992659
 -0.43413969 -0.41835279 -0.40256589 -0.386779   -0.3709921  -0.3552052
 -0.3394183  -0.3236314  -0.30784451 -0.29205761 -0.27627071 -0.26048381
 -0.24469692 -0.22891002 -0.21312312 -0.19733622 -0.18154932 -0.16576243
 -0.14997553 -0.13418863 -0.11840173 -0.10261484 -0.08682794 -0.07104104
 -0.05525414 -0.03946724 -0.02368035 -0.00789345  0.00789345  0.02368035
  0.03946724  0.05525414  0.07104104  0.08682794  0.10261484  0.11840173
  0.13418863  0.14997553  0.16576243  0.18154932  0.19733622  0.21312312
  0.22891002  0.24469692  0.26048381  0.27627071  0.29205761  0.30784451
  0.3236314   0.3394183   0.3552052   0.3709921   0.386779    0.40256589
  0.41835279  0.43413969  0.44992659  0.46571348  0.48150038  0.49728728
  0.51307418  0.52886107  0.54464797  0.56043487  0.57622177  0.59200867
  0.60779556  0.62358246  0.63936936  0.65515626  0.67094315  0.68673005
  0.70251695  0.71830385  0.73409075  0.74987764  0.76566454  0.78145144
  0.79723834  0.81302523  0.82881213  0.84459903  0.86038593  0.87617283
  0.89195972  0.90774662  0.92353352  0.93932042  0.95510731  0.97089421
  0.98668111  1.00246801  1.01825491  1.0340418   1.0498287   1.0656156
  1.0814025   1.09718939  1.11297629  1.12876319  1.14455009  1.16033699
  1.17612388  1.19191078  1.20769768  1.22348458  1.23927147  1.25505837
  1.27084527  1.28663217  1.30241906  1.31820596  1.33399286  1.34977976
  1.36556666  1.38135355  1.39714045  1.41292735  1.42871425  1.44450114
  1.46028804  1.47607494  1.49186184  1.50764874  1.52343563  1.53922253
  1.55500943  1.57079633]
AoA is at: -40.25125628140703 degrees
try to plot
done
end calculation
end common
release2
Header: BB 77 BA
OFDM PHY Header: RATE 0x0B LENGTH 957 RESERVED 1 TAIL 46 --> Parity check failed. RATE
not supported. Length invalid.  [Correct: 4/6]
grep1
[2,1] packets
release1
grep1
[2,1] packets
release1
Header: CB 09 02
OFDM PHY Header: RATE 0x0B LENGTH 78 RESERVED 0 TAIL 0 --> n_ofdm_syms 27 [Correct:
5/7]

##################
INFO: device_handler::close_device(): Disconnected from device number 0.
##################
```

Access point 4

```
Using Volk machine: ssse3_64_orc
--------------------------------------------------------------
LimeSuite Source (RX) info
```

```
##################
Connecting to device
##################
Devices found: 1
##################
Device list:
Nr.:|0|device:|LimeSDR-USB, media=USB 2.0, module=FX3, addr=1d50:6108,
serial=00090726074E1515
##################
INFO: device_handler::open_device(): no serial number. Using first device in the list.
Use "LimeUtil --find" in terminal to find prefered device serial.
##################
Device number 0 from the list is used.
##################

INFO: device_handler::enable_channels(): MIMO mode set for device number 0.
INFO: device_handler::set_samp_rate(): set sampling rate: 10 MS/s.
INFO: device_handler::set_rf_freq(): RF frequency set [RX]: 2472 MHz.
INFO: device_handler::set_analog_filter(): INFO: device_handler::set_analog_filter():
INFO: device_handler::set_gain(): set gain [RX] CH0: 60 dB.
INFO: device_handler::set_gain(): set gain [RX] CH1: 60 dB.
INFO: device_handler::set_antenna(): CH0 antenna set [RX]: LNAH.
INFO: device_handler::set_antenna(): CH1 antenna set [RX]: LNAH.
INFO: device_handler::calibrate(): INFO: device_handler::calibrate(): INFO:
source_impl::init_stream(): source channel 0 (device nr. 0) stream setup done.
INFO: source_impl::init_stream(): source channel 1 (device nr. 0) stream setup done.

##################
INFO: device_handler::close_device(): Disconnected from device number 0.
##################
```

Appendix G – GNU Radio CSI Filtering

Key GNU Radio module for filtering signals

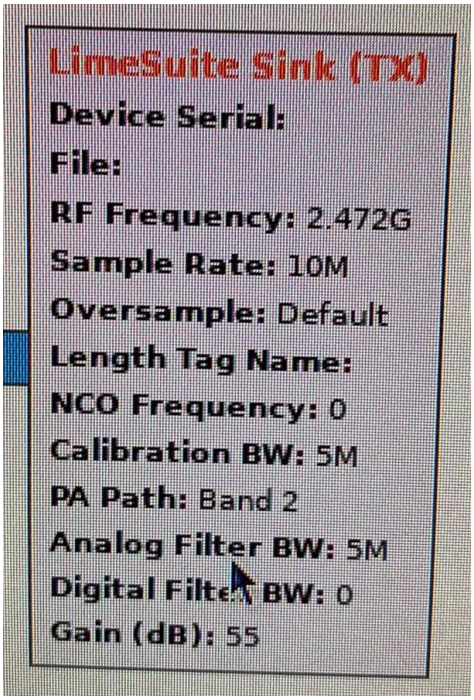


Figure 1. GNU Radio Module to Filter Signals

Coding snipped of blocks used to filter signals in GNU Radio

```
        ##################################################
        # Blocks
        ##################################################
        self.swifi_wifi_conv_decoder_0_1 = swifi.wifi_conv_decoder("phy")
        self.swifi_wifi_conv_decoder_0_0_0 = swifi.wifi_conv_decoder(24, -1)
        self.swifi_wifi_conv_decoder_0_0 = swifi.wifi_conv_decoder(24, -1)
        self.swifi_wifi_conv_decoder_0 = swifi.wifi_conv_decoder("phy")
        self.swifi_scrambler_0_0 = swifi.scrambler(False,"phy")
        self.swifi_scrambler_0 = swifi.scrambler(False,"phy")
        self.swifi_sample_marker_0_0 = swifi.sample_marker(gr.sizeof_gr_complex,"phy")
        self.swifi_sample_marker_0 = swifi.sample_marker(gr.sizeof_gr_complex,"phy")
        self.swifi_payload_parser_0_0 = swifi.payload_parser("phy",dump_filename,"text",ff,Tru
e)
        self.swifi_payload_parser_0 = swifi.payload_parser("phy",dump_filename,"text",ff,True)
        self.swifi_ofdm_frame_sync_0_0 = swifi.ofdm_frame_sync(0.9,64,0.4,0.05,0.2,"trigger")
        self.swifi_ofdm_frame_sync_0 = swifi.ofdm_frame_sync(0.9,64,0.4,0.05,0.2,"trigger")
        self.swifi_ofdm_equalizer_1_0 = swifi.ofdm_equalizer("",True,"channel_states",digital.
constellation_bpsk().base())
        self.swifi_ofdm_equalizer_1_0.set_equalization_method(equalization_method)
        self.swifi_ofdm_equalizer_1_0.set_alpha(alpha)
```

```
        self.swifi_ofdm_equalizer_1_0.set_beta(beta)
        self.swifi_ofdm_equalizer_1_0.enable_foc(False)
        self.swifi_ofdm_equalizer_1 = swifi.ofdm_equalizer("",True,"channel_states",digital.co
nstellation_bpsk().base())
        self.swifi_ofdm_equalizer_1.set_equalization_method(equalization_method)
        self.swifi_ofdm_equalizer_1.set_alpha(alpha)
        self.swifi_ofdm_equalizer_1.set_beta(beta)
        self.swifi_ofdm_equalizer_1.enable_foc(False)
        self.swifi_ofdm_equalizer_0_0 = swifi.ofdm_equalizer("phy",False,"channel_states")
        self.swifi_ofdm_equalizer_0_0.set_equalization_method(equalization_method)
        self.swifi_ofdm_equalizer_0_0.set_alpha(alpha)
        self.swifi_ofdm_equalizer_0_0.set_beta(beta)
        self.swifi_ofdm_equalizer_0_0.enable_foc(False)
        self.swifi_ofdm_equalizer_0 = swifi.ofdm_equalizer("phy",False,"channel_states")
        self.swifi_ofdm_equalizer_0.set_equalization_method(equalization_method)
        self.swifi_ofdm_equalizer_0.set_alpha(alpha)
        self.swifi_ofdm_equalizer_0.set_beta(beta)
        self.swifi_ofdm_equalizer_0.enable_foc(False)
        self.swifi_ofdm_chan_est_0_0 = swifi.ofdm_chan_est(64,((0, 0j, 0, 0j, 0, 0j, -1, 1j, -
1, 1j, -1, 1j, -1, -1j, 1, 1j, 1, -1j, -1, 1j, 1, 1j, 1, 1j, 1, 1j, -1, (-0-1j), 1, -1j, -
1, 1j, 0, -1j, 1, (-0-1j), 1, -1j, 1, 1j, -1, -1j, 1, (-0-1j), -1, 1j, 1, 1j, 1, 1j, 1, 1j, -
1, -1j, 1, 1j, 1, -1j, -1, 0j, 0, 0j, 0, 0j), (0, 0, 0, 0, 0, 0, 1, 1, -1, -1, 1, 1, -1, 1, -
1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 0, 1, -1, -1, 1, 1, -1, 1, -1, 1, -
1, -1, -1, -1, -1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 1, 1, 1, 0, 0, 0, 0, 0)),0.5,
            "channel_states","frame_len",3,False)
        self.swifi_ofdm_chan_est_0_0.set_estimation_method(0)
        self.swifi_ofdm_chan_est_0 = swifi.ofdm_chan_est(64,((0, 0j, 0, 0j, 0, 0j, -1, 1j, -
1, 1j, -1, 1j, -1, -1j, 1, 1j, 1, -1j, -1, 1j, 1, 1j, 1, 1j, 1, 1j, -1, (-0-1j), 1, -1j, -
1, 1j, 0, -1j, 1, (-0-1j), 1, -1j, 1, 1j, -1, -1j, 1, (-0-1j), -1, 1j, 1, 1j, 1, 1j, 1, 1j, -
1, -1j, 1, 1j, 1, -1j, -1, 0j, 0, 0j, 0, 0j), (0, 0, 0, 0, 0, 0, 1, 1, -1, -1, 1, 1, -1, 1, -
1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 0, 1, -1, -1, 1, 1, -1, 1, -1, 1, -
1, -1, -1, -1, -1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 1, 1, 1, 0, 0, 0, 0, 0)),0.5,
            "channel_states","frame_len",3,False)
        self.swifi_ofdm_chan_est_0.set_estimation_method(0)
        self.swifi_interleaver_0_1 = swifi.interleaver(False,"phy")
        self.swifi_interleaver_0_0_0 = swifi.interleaver(False,11)
        self.swifi_interleaver_0_0 = swifi.interleaver(False,11)
        self.swifi_interleaver_0 = swifi.interleaver(False,"phy")
        self.swifi_header_payload_demux_0_0 = swifi.header_payload_demux(
                3,
                64,
                16,
                "frame_len",
                "trigger",
                True,
```

```
                gr.sizeof_gr_complex,
                "phy")
        self.swifi_header_payload_demux_0 = swifi.header_payload_demux(
                3,
                64,
                16,
                "frame_len",
                "trigger",
                True,
                gr.sizeof_gr_complex,
                "phy")
        self.swifi_constellation_decoder_0_0 = swifi.constellation_decoder("phy")
        self.swifi_constellation_decoder_0 = swifi.constellation_decoder("phy")
        self.swifi_bit_unpacker_0_0 = swifi.bit_unpacker("phy",1)
        self.swifi_bit_unpacker_0 = swifi.bit_unpacker("phy",1)
        self.swifi_bit_inserter_0_0 = swifi.bit_inserter("phy")
        self.swifi_bit_inserter_0 = swifi.bit_inserter("phy")
        self.spectrum_plotter_two_input_plotter_0 = spectrum_plotter.two_input_plotter(-
8, 105, 3)
        self.spectrum_plotter_csi_collector_1 = spectrum_plotter.csi_collector(gr.sizeof_gr_co
mplex*1)
        self.spectrum_plotter_csi_collector_0 = spectrum_plotter.csi_collector(gr.sizeof_gr_co
mplex*1)
        self.limesdr_source_0 = limesdr.source("", 2, "")
        self.limesdr_source_0.set_sample_rate(10e6)
        self.limesdr_source_0.set_center_freq(2.472e9, 0)
        self.limesdr_source_0.set_bandwidth(20e6,0)
        self.limesdr_source_0.set_bandwidth(20e6,1)
        self.limesdr_source_0.set_gain(60,0)
        self.limesdr_source_0.set_gain(60,1)
        self.limesdr_source_0.set_antenna(1,0)
        self.limesdr_source_0.set_antenna(1,1)
        self.limesdr_source_0.calibrate(5e6, 0)
        self.limesdr_source_0.calibrate(5e6, 1)

        (self.limesdr_source_0).set_max_output_buffer(100)
        self.fft_vxx_0_1 = fft.fft_vcc(64, True, (window.rectangular(64)), True, 1)
        self.fft_vxx_0_0_0 = fft.fft_vcc(64, True, (window.rectangular(64)), True, 1)
        self.fft_vxx_0_0 = fft.fft_vcc(64, True, (window.rectangular(64)), True, 1)
        self.fft_vxx_0 = fft.fft_vcc(64, True, (window.rectangular(64)), True, 1)
        self.digital_packet_headerparser_b_0_0_0 = digital.packet_headerparser_b(header_format
ter_2.base())
        self.digital_packet_headerparser_b_0_0 = digital.packet_headerparser_b(header_formatte
r_2.base())
```

```
        self.digital_constellation_decoder_cb_0_0 = digital.constellation_decoder_cb(header_mo
d.base())
        self.digital_constellation_decoder_cb_0 = digital.constellation_decoder_cb(header_mod.
base())
        self.blocks_vector_to_stream_1_0_0_0_0 = blocks.vector_to_stream(gr.sizeof_gr_complex*
1, 48)
        self.blocks_vector_to_stream_1_0_0_0 = blocks.vector_to_stream(gr.sizeof_gr_complex*1,
 48)
        self.blocks_vector_to_stream_1_0_0 = blocks.vector_to_stream(gr.sizeof_gr_complex*1, 4
8)
        self.blocks_vector_to_stream_1_0 = blocks.vector_to_stream(gr.sizeof_gr_complex*1, 48)
        self.blocks_unpacked_to_packed_xx_0_1 = blocks.unpacked_to_packed_bb(1, gr.GR_LSB_FIRS
T)
        self.blocks_unpacked_to_packed_xx_0_0_0 = blocks.unpacked_to_packed_bb(1, gr.GR_LSB_FI
RST)
        self.blocks_unpacked_to_packed_xx_0_0 = blocks.unpacked_to_packed_bb(1, gr.GR_LSB_FIRS
T)
        self.blocks_unpacked_to_packed_xx_0 = blocks.unpacked_to_packed_bb(1, gr.GR_LSB_FIRST)
```

Appendix H – Position Estimation

Below is the function used to estimate the transmitter position. An equivalent line for each angle of arrival is generated and the points of intersection is then calculated. A normal distribution was used to "plot" probabilities of position as you move out from each intersection point and the standard deviation was set to be 1/3rd of the maximum distance between points of intersection, that lied within the testing grip.

```
def calculatePosition(receiverData, resolution=Resolution.Centimeters):
    positionMatrix = np.zeros((yDimension, xDimension))
    intersectionPoints = np.array(np.shape(0))

    # Update Lines
    directionLines = np.array(np.shape(0))
    for i in range(len(receiverData)):
        data = receiverData[i]
        receiverId = data.get_receiverId()
        equivalentSlope = math.tan(data.get_angleOfArrival())
        receiverPosition = receivers[receiverId].get_receiverPosition()

        updatedLine = Line(equivalentSlope, findYIntercept(
            equivalentSlope, receiverPosition))
        print(str(updatedLine))
        directionLines = np.append(directionLines, updatedLine)

    # Calculate Intersection points
    numLines = len(directionLines)
    avgPosition = Position(0, 0)
    for i in range(numLines - 1):
        for j in range(i + 1, (numLines)):
            # TODO: Think more about case where the object is outside the 10x10 meter grid
            line1 = directionLines[i]
            line2 = directionLines[j]

            # Check if parallel lines
            if (line1.slope - line2.slope) == 0:
                continue

            xIntersection = (
                (line2.intercept - line1.intercept) / (line1.slope - line2.slope))
            intersectionPoint = Position(
                xIntersection, (line1.slope * xIntersection + line1.intercept))

            if isInBounds(intersectionPoint):
                intersectionPoints = np.append(
                    intersectionPoints, intersectionPoint)

                numIntersectionPoints = len(intersectionPoints)
                if numIntersectionPoints == 1:
                    avgPosition = intersectionPoint
                else:
                    avgPosition = Position((avgPosition.x * (numIntersectionPoints - 1)) / numIntersectionPoints +
                     intersectionPoint.x / numIntersectionPoints,
                                           (avgPosition.y * (numIntersectionPoints - 1)) / numIntersectionPoints +
                                            intersectionPoint.y / numIntersectionPoints)
```

```
    # print(str(intersectionPoints[0]))
    # print("Avg intersection point: ", str(avgPosition))
    # Estimate position of transmitter
    maxRange = 100
    sigma = (maxRange / 3.0)
    # print ("sigma", sigma)
    for i in range(maxRange + 1):
        for j in range(maxRange + 1):
            distance = calculateMagnitude(Position(i, j))
            density = calculateGaussian(sigma, distance)

            for point in intersectionPoints:
                for update in range(4):
                    if update == 0:
                        y = int(point.y) + j
                        x = int(point.x) + i
                    elif update == 1:
                        y=int(point.y) - j
                        x=int(point.x) - i
                    elif update == 2:
                        y=int(point.y) + j
                        x=int(point.x) - i
                    elif update == 3:
                        y=int(point.y) - j
                        x=int(point.x) + i

                    if isInBounds(Position(x,y)) and (update == 0 or (update == 1 and (i != 0 or j != 0)) or (update > 1 and (i != 0 and j != 0))):
                        positionMatrix[y, x] += density

    maxVal = positionMatrix.max()
    # Take top 90%
    for i in range(positionMatrix.shape[0]):
        for j in range(positionMatrix.shape[1]):
            if positionMatrix[i][j] < 0.1*maxVal:
                positionMatrix[i][j] = 0

    # Show plot of position estimation
    plotHeatmap(positionMatrix)
    positionMatrix = changeResolution(positionMatrix, resolution)
    print("changed resolution from Centimeters to ", resolution)
    plotHeatmap(positionMatrix)

    return positionMatrix
```

Below are the custom objects/types used to manage data:

The position class is class is used to store and represent the coordinates of a position in 3 dimensions.

```
class Position:
    def __init__(self, x = None, y = None, z = 0):
        self.x = x
        self.y = y
        self.z = z

    def get_x(self):
        return self.x

    def get_y(self):
        return self.y

    def get_z(self):
        return self.z

    def __str__(self):
        return "({0},{1},{2})".format(str(self.x), str(self.y), str(self.z))
```

The two classes below are used to represent the receivers (their ID, position and a calibration angle offset) and the angle of arrival data received for a receiver.

```
class Receiver:
    def __init__(self, receiverId = None, receiverPosition = None, angleOffset = 0):
        self.__receiverId = receiverId
        self.__receiverPosition = receiverPosition
        self.__angleOffset = angleOffset

    def __str__(self):
        return "{0}: Position -> {1}; Angle Offset -> {2}".format(str(self.__receiverId), str(self.__receiverPosition),
        str(self.__angleOffset))

    def get_receiverId(self):
        return self.__receiverId

    def get_receiverPosition(self):
        return self.__receiverPosition

    def get_angleOffset(self):
        return self.__angleOffset
```

```
class ReceiverData:

    def __init__(self, receiverId = None, timeStamp = None, angleOfArrival = None):
        self.__receiverId = receiverId
        self.__timeStamp = timeStamp
        self.__angleOfArrival = angleOfArrival

    def __str__(self):
        return "{0}.{1}: {2}".format(str(self.__receiverId), str(self.__timeStamp), str(self.__angleOfArrival))

    def get_receiverId(self):
        return self.__receiverId

    def get_timeStamp(self):
        return self.__timeStamp

    def get_angleOfArrival(self):
        return self.__angleOfArrival
```

The class below is used to represent a line in 2 dimensions.

```
class Line:
    def __init__(self, slope = 0, intercept = 0):
        self.slope = slope
        self.intercept = intercept

    def __str__(self):
        return "y = {0}*x + {1}".format(str(self.slope), str(self.intercept))

    def calculateX(self, y):
        return (y - self.intercept)/self.slope

    def calculateY(self, x):
        return self.slope * x + self.intercept
```

The class described below is used to manage the connections with the position calculation module and the backend-server that connects to the DB. This allows for data to be received/stored

```
import requests
import json
from  enum import Enum

class RequestType(Enum):
    GET = 1
    POST = 2

class ConnectionManager:
    def getData(self, year=None, month=None, day=None, timeStart=None, timeEnd=None):
        rawData = None

        if year != None and month != None and day != None and timeFrom != None and timeTo != None:
            data = [
            ('year', year),
            ('month', month),
            ('day', day),
            ('timeFrom', timeFrom),
            ('timeTo', timeTo)
            ]
            url = "http://localhost:8888/analysis/byTime"
            rawData = self.__executeApiCall(RequestType.GET, url, data)
        else:
            url = "http://localhost:8888/analysis/allData"
            rawData = self.__executeApiCall(RequestType.GET, url)

        return rawData

    def sendPositionData(self, data):
        url = "http://localhost:8888/analysis/storeAnalysisData"
        response = self.__executeApiCall(RequestType.POST, url, {'data':data})

    def __executeApiCall(self, callType, url, data=None):
        if callType == RequestType.GET:
            if data != None:
                req = requests.get(url = url, data = data)
            else:
                req = requests.get(url = url)
        elif callType == RequestType.POST:
            return 0 # here just to prevent call in testing
            req = requests.post(url = url, data = data)

        if req:

            return req.json()
        else:
            print("Error in exectuting api call: {}".format(str(url)))
            return None
```

Appendix I – Express.js Server

Instructions to Set Up:
Once the git repository has been cloned and downloaded from the repository provided under: https://github.com/mohaimenhasan/Indoor-Localization-Capstone-Project, the server can be installed under the backend folder. All the packages can be installed through **npm install** command under the /backend/ folder with the packaged libraries. The server can be started with **npm start** or the test server can be run with **npm run dev**.

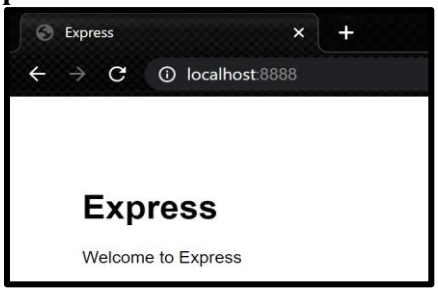


Figure 1. Server starting under http://localhost:8888/

Packages:
The main server boots up under localhost in the port 8888 available under http://localhost:8888/, which starts with the npm command start, defined under */backend/package.json*, consisting of all the packages. The server packages have been defined and packaged with all the necessary libraries, which can be set up on any machine with npm installed [19].

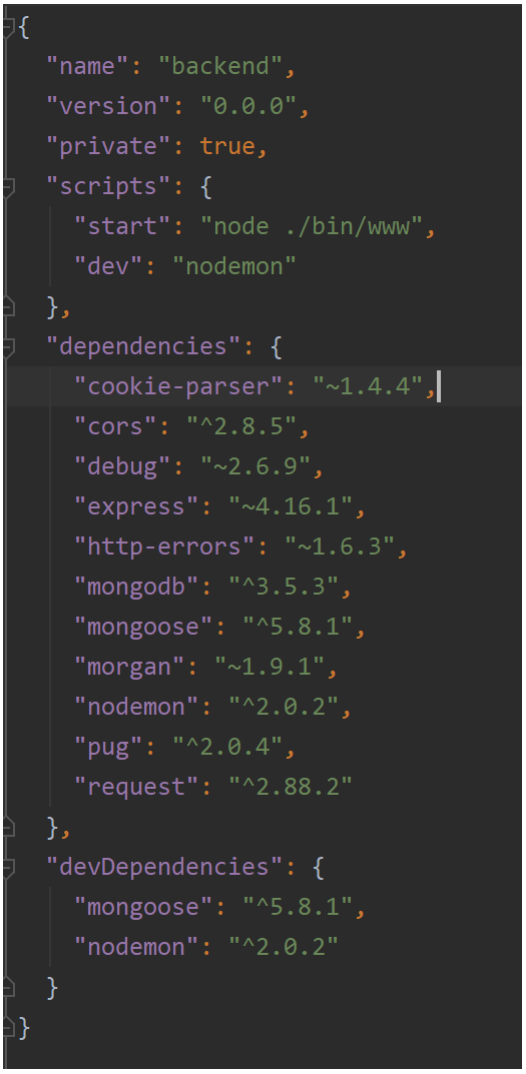


Figure 2. NPM Packages defined under package.json

Max Limit: The max for the payload has been defined under /backend/app.js

```
app.use(bodyParser.json({limit: '50mb'}));
app.use(bodyParser.urlencoded({ limit: '50mb', extended: true }));
```

Figure 3. Maxiumum Payload Limit

Main Routes:
The following images identify the main 2 routes for the backend routes of the backend server:
In the */backend/app.js* the following routes have been defined:

```
// create routes here
app.use( fn: '/', indexRouter);
app.use( fn: '/analysis', analysisRouter);
app.use( fn: '/receiver', receiverRouter);
```

Figure 1. Main routes available

Corresponding to the routes the index pages of the routes are as following:

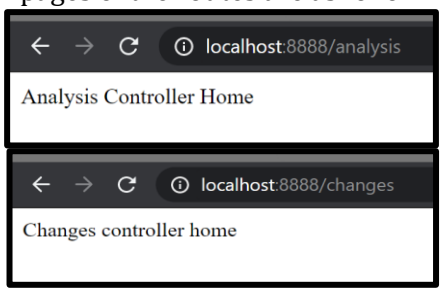


Figure 2. Index Pages for the routes

The defined routes can be shown under */backend/routes/*. The routes functions have been broken in the following order.

Routes Functions:

1. **Analysis.js**

```
var express = require('express');
var router = express.Router();

let analysisController = require('../controllers/analysis.controller');
/* GET home page. */
router.get('/', analysisController.home);
router.get('/allData', analysisController.getAllData);
router.get('/byTime', analysisController.parseByTime);
router.post('/storeAnalysisData', analysisController.storeAnalysisData);
router.post('/sendAnalysisData', analysisController.sendAnalysisData);
router.get('/getAllDates', analysisController.getAllDates);
router.get('/getCurrentRun', analysisController.runPythonCur);
module.exports = router;
```

Figure 3. Analysis Routes Functions

The analysis functionalities are coded as following:

Store Data:

```
exports.storeAnalysisData = async function(req, res, next){
    // sanity check
    try{
        let val = req.body.data;
        val = JSON.parse(val);
        let gridData = new Array(val["position"]);
        let timeFrom = new Date(val["timefrom"]);
        let timeTo = new Date(val["timeto"]);
        let allReceivers = val["receivers"];
        let griddim = new Array(val["gridDim"]);
        await saveAnalysisData(gridData, timeFrom, timeTo, griddim,
            allReceivers["receiver0"]["position"], allReceivers["receiver0"]["angle_of_arrival"], allReceivers["receiver0"]["line"],
            allReceivers["receiver1"]["position"], allReceivers["receiver1"]["angle_of_arrival"], allReceivers["receiver1"]["line"],
            allReceivers["receiver2"]["position"], allReceivers["receiver2"]["angle_of_arrival"], allReceivers["receiver2"]["line"],
            allReceivers["receiver3"]["position"], allReceivers["receiver3"]["angle_of_arrival"], allReceivers["receiver3"]["line"])
            .then(response => {
                if (response === null){
                    res.send("Data could not be stored");
                }else{
                    res.send("Completed");
                }
            })

    }catch(err){
        console.log("Check Error: ", err);
        res.send("Bad Data");
    }
};
```

```
try{
    let analVal = new AnalysisVal({
        position: gridData,
        timefrom: from,
        timeto: to,
        receivers: {
            receiver1: {
                position: receiver1Pos,
                angle_of_arrival: receiver1aoa,
                line: receiver1line
            },
            receiver2: {
                position: receiver2Pos,
                angle_of_arrival: receiver2aoa,
                line: receiver2line
            },
            receiver3: {
                position: receiver3Pos,
                angle_of_arrival: receiver3aoa,
                line: receiver3line
            },
            receiver4: {
                position: receiver4Pos,
                angle_of_arrival: receiver4aoa,
                line: receiver4line
            }
        },
        griddim: griddim
    });
    return await analVal.save(function(err){
        if (err){
            throw err;
        }
        return analVal;
    });
```

Getting Data by time:

```
async function getAnalysisData(from, to){
    try{
        // console.log(from);
        // console.log(to);
        return await AnalysisVal.find()
            .then(data=>{
                //console.log(data);
                let vals = [];
                for(let i=0; i < data.length; ++i){
                    if (data[i].timefrom >= from && data[i].timeto <= to){
                        vals.push(data[i]);
                    }
                }
                return vals;
            })
    }catch (err) {
        console.log("ERROR");
        return undefined;
    }
}
```

Retrieving All Data:

```
exports.parseByTime = async function(req, res, next) {
    await allData()
        .then(data => {
            // once you have the data you want to parse it here
            // time - // year, month, day, hour, min, second
            console.log(req.body.day+" "+req.body.month+" "+req.body.year + " " + req.body.timeFrom);
            console.log(req.body.day+" "+req.body.month+" "+req.body.year + " " + req.body.timeTo);
            let timeFrom = new Date( value: req.body.day+" "+req.body.month+" "+req.body.year + " " + req.body.timeFrom);
            let timeTo = new Date( value: req.body.day+" "+req.body.month+" "+req.body.year + " " + req.body.timeTo);
            let retVal = [];
            for (let i=0; i < data.length; ++i){
                console.log(data[i].timestamp)
                console.log(timeFrom);
                console.log(timeTo);
                if (new Date(data[i].timestamp) >= timeFrom && new Date(data[i].timestamp) <= timeTo){
                    retVal.push(data[i]);
                }
            }
            res.send(retVal);
        })
};

async function allData() {
    try{
        return await ReceiverVal.find()
            .then(data => {
                return data;
            })
    }catch(err){
        console.log("Nothing could be received");
        return null;
    }
```

2. Receiver.js

```
var express = require('express');
var router = express.Router();

let receiverController = require('../controllers/receiverval.controller');
/* GET home page. */
router.post('/addData', receiverController.postVal);

module.exports = router;
```

Figure 4. Receiver Routes Functions

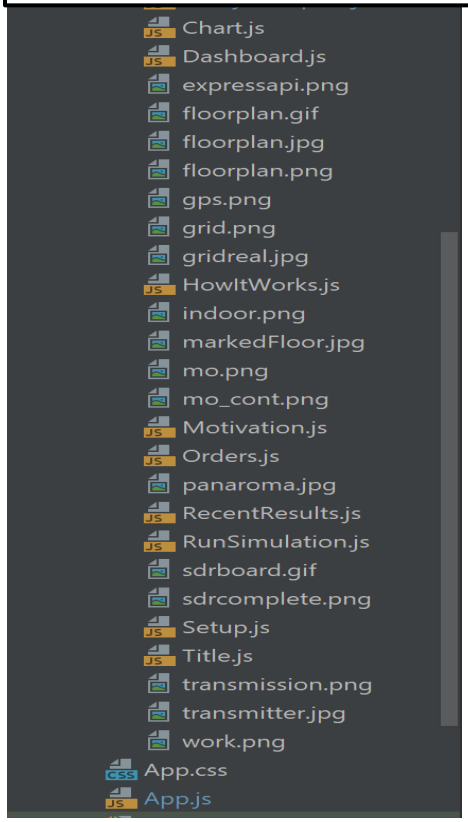
Receiver functionality has been coded as following:

Store Values from all Receivers:

```
let accessPt = req.body.access_point;
let timestamp = req.body.timestamp;
let AoA = req.body.angle_of_arrival;
let csi_frame = req.body.csi_frame;
let grid = req.body.grid;
console.log("The request is: ");
console.log(req.body);
let val = await retrieveValues(accessPt, timestamp, AoA, csi_frame, grid);
if (val === null){
    res.send({
        code: 500,
        message: "ERROR !!! Did not receive any data"
    });
}else{
    res.send({
        code: 200,
        message: val
    })
}
```

Appendix J – React.js Server on Frontend

Initial Server Design:



React.js Server Coding:

App.js:
By default, the app lands on Dashboard as it has been set under app.js:

```
import React, {Component} from 'react';
import Dashboard from "./components/Dashboard";
import './App.css';

class App extends Component{
    constructor(props){
        super(props);
        this.state = {
            currentScreen: []
        }
    }

    componentDidMount() {
        this.setState( state: {
            currentScreen: <Dashboard appContext={this}/>
        })
    }

    render(){
        return (
            <div className="App">
                {this.state.currentScreen}
            </div>
        );
    }
}

export default App;
```

Other screens are navigated through props:

```
changeDashboard(event){
    this.props.appContext.setState( state: {
        currentScreen: <Dashboard appContext={this.props.appContext}/>
    })
}

changeToCurrent(event){
    console.log("Current Screen picked");
    this.props.appContext.setState( state: {
        currentScreen: <Current appContext={this.props.appContext}/>
    });
}
```

Fetching Data from backend through API calls:

```
handleSubmit = async (event) => {
    const classes = this.props.classes;
    await fetch( url: api_base_url+'analysis/sendAnalysisData',
        opts: {
            method: 'POST',
            headers: {
                "Content-Type": "application/json",
                "Access-Control-Allow-Origin": '*'
            },
            body: JSON.stringify( value: {
                    from: this.state.fromDate,
                    to: this.state.toDate,
                }
            )
        }).then(res => res.json())
        .then(response => {
            //console.log(data);
            let heatMaps = [];
            for (let k in response){
                let positionMatrix = response[k]["position"][0];
                let xDim = response[k]["griddim"][0][0];
                let yDim = response[k]["griddim"][0][1];
                positionMatrix = Array.from(positionMatrix);
                const xLabels = new Array(Math.round(xDim)).fill( value: 0).map((_,i : number ) => `${i+1}m`);
                const yLabels = new Array(Math.round(yDim)).fill( value: 0).map((_,i : number ) => `${i+1}m`);
```

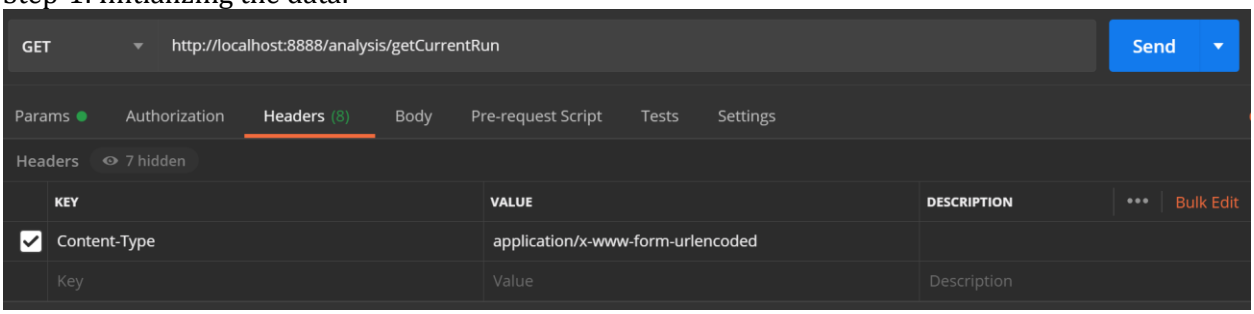
Appendix K – Assessment of the Design

Calculation of number of access points as payload:

The maximum size of payload file observed from any request is: 19 KB – receiverfiles.txt [Appendix-F]
Maximum payload from any server: 50 MB
Minimum Number of requests/second:
$\frac{(Maximum\ Payload{=}50\ MB)}{(Max\ observed\ file\ size{=}18KB)}$ = 2844 requests / second

Running the Calculation in Real Time:

Step-1: Initializing the data:



Step-2: Running the module:

```
POST /analysis/storeAnalysisData 200 72.219 ms - 9
{
  receivers: {
    receiver1: {
      position: [Array],
      line: 'y = -0.039487749395*x + 335.0',
      angle_of_arrival: -0.03946724439183158
    },
    receiver2: {
      position: [Array],
      line: 'y = 0.183570608548*x + 0.0',
      angle_of_arrival: 0.18154932420242534
    },
    receiver3: {
      position: [Array],
      line: 'y = -0.846600130229*x + 773.538867459',
      angle_of_arrival: -0.7025169501746019
    },
    receiver4: {
      position: [Array],
      line: 'y = -0.134999901212*x + 69.9299488276',
      angle_of_arrival: -0.13418863093222733
    }
  },
  position: [ [ [Array], [Array], [Array] ] ],
  griddim: [ [ 5.18, 3.35 ] ],
  _id: 5e7a5110de7b763bac6ee6a1,
  timefrom: 2020-03-24T18:27:20.967Z,
  timeto: 2020-03-24T18:27:23.995Z,
  __v: 0
}
GET /analysis/getCurrentRun 200 8662.042 ms - 745
[nodemon] starting `node ./bin/www`
```

Expanding on the results it can found that:

{"receivers":{"receiver1":{"position":[0,335],"line":"y = -0.039487749395*x + 335.0","angle_of_arrival":-0.03946724439183158},"receiver2":{"position":[0,0],"line":"y = 0.183570608548*x + 0.0","angle_of_arrival":0.18154932420242534},"receiver3":{"position":[518,335],"line":"y = -0.846600130229*x + 773.538867459","angle_of_arrival":-0.7025169501746019},"receiver4":{"position":[518,0],"line":"y = -0.134999901212*x + 69.9299488276","angle_of_arrival":-0.13418863093222733}},"position":[[[0,0.00177166540520327,0.0050361926223562265,0,0],[0,0.00001439674525677260 2,0.00009150748328134457,0,0],[0,0,0,0,0]]],"griddim":[[5.18,3.35]],"_id":"5e7a5110de7b763bac6ee6a1","timefrom":"2020-03-24T18:27:20.967Z","timeto":"2020-03-24T18:27:23.995Z","__v":0}

Figure 1. Results from Postman [24]

Results from the run:


```
{
   "receivers": {
      "receiver1": {
         "position": [0,335],
         "line": "y = -0.039487749395*x + 335.0",
         "angle_of_arrival": -0.03946724439183158
      },
      "receiver2": {
         "position": [0,0],
         "line": "y = 0.183570608548*x + 0.0",
         "angle_of_arrival": 0.18154932420242534
      },
      "receiver3": {
         "position": [518,335]
         "line": "y = -0.846600130229*x + 773.538867459",
         "angle_of_arrival": -0.7025169501746019
      },
      "receiver4": {
         "position": [518,0],
         "line": "y = -0.134999901212*x + 69.9299488276",
         "angle_of_arrival": -0.13418863093222733
      }
   },
   "position": [
      [
         [0,0.00177166540520327,0.0050361926223562265,0,0],
         [0,0.00001439674525677260 2,0.00009150748328134457,0,0],
         [0,0,0,0,0]
      ]
   ],
   "griddim": [
      [5.18,3.35]
   ],
   "_id": "5e7a5110de7b763bac6ee6a1",
   "timefrom": "2020-03-24T18:27:20.967Z",
   "timeto": "2020-03-24T18:27:23.995Z",
   "__v": 0
}
```

Appendix L – Transmitting Data from the laptop

Laptop Module Installation Failures:

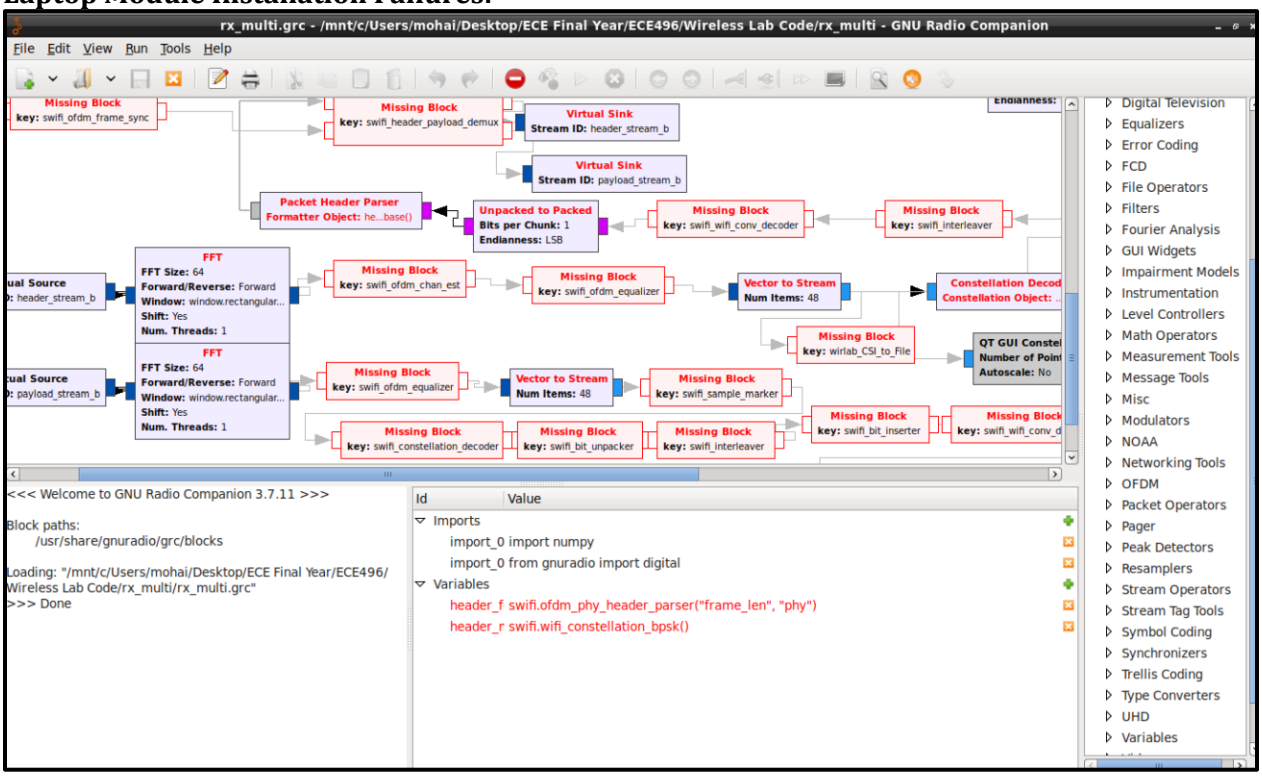


Figure 1. Module Failure

Custom Modules not installable:

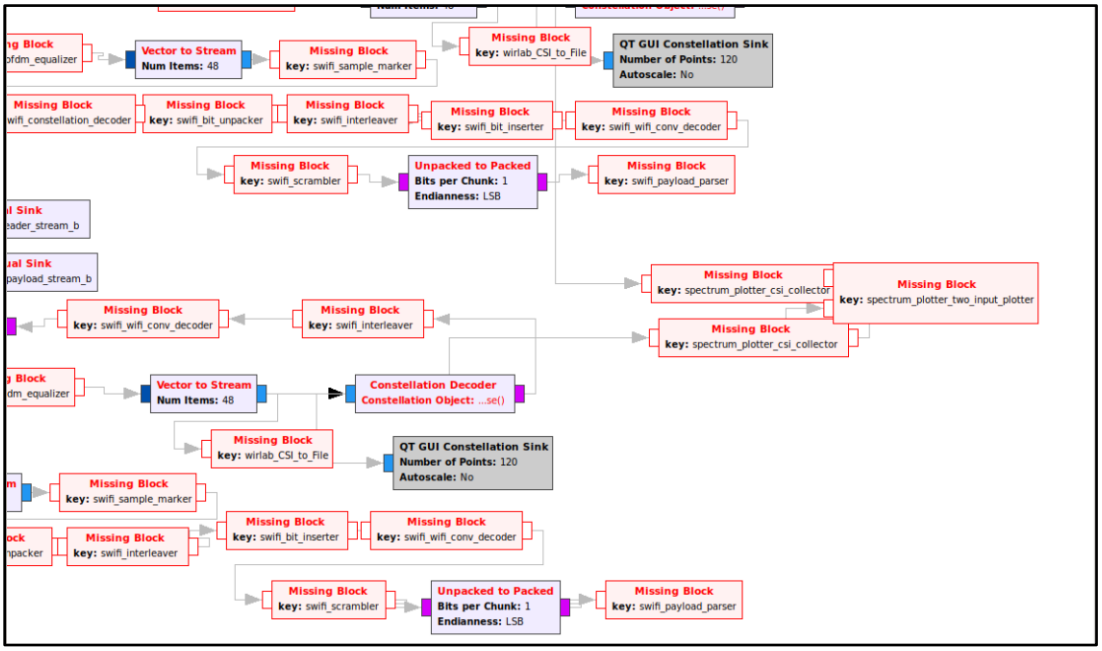


Figure 2. Custom Modules Missing