NSWC Crane RF Challenge: Final Design Document

Grayden Johnson, Sean Page, Andrew Greiner, Vighnesh Pawaskar, Jacob Risser

Indiana University Purdue University Indianapolis

Todd M Kuebelbeck, Jr

Mechanical Engineer

Naval Surface Warfare Center, Crane Division, (NSWC Crane)

Expeditionary Electronic Warfare — Counter UxS (WXQX)

(630) 886 - 5161

Table of Contents

Table of Contents Continued

List of Figures

List of Tables

1. Project Motivation

NSWC Crane created this project with the goal to collect, identify, organize, and display various RF signals to the user. The objective is to give the nation's warfighters, the US Marines, a method to simulate RF traffic in a large city while training in remote or rural locations. The project was spurred from the Depart of Defense's interest in emulating an urban environment's electromagnetic traffic. The hardest hurdle to overcome is the compilation of the data. Even with the extreme amount of data that is collected, Crane was relying on human power to identify and classify all signals received. This is extremely inefficient because there could be terabytes of data. The solution devised was to create an AI-ML script that could identify, classify, and categorize all the data collected at an improved rate. Once this project is completed the Electronics Warfare Department at NSWC Crane will expand upon the project by integrating the design into a handheld module that can be used in the field.

1.1. Background Information

This project relies heavily on concepts of RF signals, or radio frequency signals. At the most basic level, an antenna can produce electromagnetic oscillations, or RF waves, simply by using a varying electrical signal. These RF waves are measured in units called hertz (Hz), which are the number of cycles per second when the radio wave is transmitted. One hertz is equal to one cycle per second. In a radio wave, the wavelength is inversely proportional to the frequency. Even though radio frequencies are not visible to the human eye, the RF spectrum is very broad ranging from 30 Hz to 300 GHz.

There are many types of wireless devices that make use of the RF spectrum. Some of the most common are radio and television broadcast stations, cellphones, Wi-Fi and Bluetooth, satellite communications systems, air traffic control systems, and even key fobs. Anything that uses a form of wireless communication will make use of RF waves and the RF spectrum.

The second big concept this project relies on is AI-ML, or Artificial Intelligence Machine Learning, a new area of computer engineering which allows the computer to think for itself. AI-ML has a multitude of applications. In our case we are using our designed AI-ML model to receive data from our previously mentioned SDR, then run it through our trained model to classify and organize RF signals detected within the immediate

environment. From a high level view, the AI-ML is a black box that data goes into and predicted results come out of.

1.2. Stakeholders

Team Members:

Sean is the designated communication manager, with being assigned to organizing meetings with the sponsor, coordinating dates for deadlines, and keeping the team on schedule. Aside from the team role Sean has very good experience with Python Scripts and programming in general. This being said he shall be spearheading the development of the AI-ML Script and helping with the GUI development.

Grayden is the team manager, which means he is responsible for helping the team work together by organizing meetings, taking notes during these meetings, and promoting an inclusive environment for everyone. Grayden has experience with hardware similar to the SDR so he will be responsible for understanding how the SDR operates and how it will integrate with the rest of the project.

Vi is the project manager, which means that he will be focusing on the technical aspects of the project. Vi will be assigning work to team members and making sure work is being done on time. If there is a standstill on a decision, Vi will make the final call.

Andrew is the resource manager, this means that he will be making sure our SDR is safe and being distributed properly to the team members that are in demand of it. This also means that he is making sure that we are bringing our laptops to meetings to make sure we are being effective. Lastly, Andrew is taking notes of our meeting and making sure our notebooks are up to date.

Jacob is second in command to Sean and also Grayden. This will mean that Jacob will make sure communication with the sponsor is staying consistent and to relay messages from other team members to our sponsor in the case that Sean was not able to. Also, being second in command as the team manager means that Jacob is making sure people are on time to our meetings and that they know all the information pertaining to the meeting and our objective.

Project Sponsors:

Todd Kuebelbeck is a Mechanical Engineer that works within the DOD. His department is within the Unmanned Aerial Vehicles at Crane Naval Base, he has a few years in his field.

Scot Hawkins has a doctorate in Electrical Engineering, he has worked at Crane Naval Base for a numerous amount of years within the Unmanned Aerial Vehicles Department.

1.3. Project Overview

Document deliverables developed:

- Requirements Document (September 8, 2023)
- Sponsor Revised Requirements Document (September 20, 2023)
- Design Options Document (October 20, 2023)
- Design Document (November 17, 2023)
- Parameter List of RF Signals (November 24th, 2023)
- Final Design Document (March 22nd, 2024)

We are scheduled to meet with our sponsor every two weeks to discuss where we were at with the project. During these meetings, we learn in greater detail the scope of the project and what Crane was looking for. We received very useful links to different websites where we could learn more about RF and complete the necessary background research for this project. Every week we do not meet with our sponsor, we meet as a team and discuss weekly progress.

Sponsor Meetings:

- Went over scope, deliverables, and expectations (September 20, 2023)
- Presented requirements document to sponsor (October 2, 2023)
- Presented design document to sponsor (November 20, 2023)
- Met in person with sponsor, received hardware, and discussed project moving forward (December 8, 2023)
- Updated sponsor on progress, discussed upcoming project management schedule (February 5, 2024)
- Officially merged Group A and B together and reassigned roles, presented updated project management schedule (February 12, 2024)
- Large update meeting, discussed what we had working and work we still needed to do (April 1, 2024)
- Weekly progress meeting, implemented features that were introduced in the previous meeting (April 5, 2024)
- Weekly progress meeting, discussed final deliverables (April 12, 2024)

1.3.1 Technical Requirements: Constraints

Ettus Research USRP B205mini-i

- 70 MHz - 6 GHz frequency range
- Up to 56 MHz of instantaneous bandwidth
- Full duplex operation
- User-programmable, industrial grade Xilinx Spartan-6 XC6SLX150 FPGA
- Fast and convenient bus-powered USB 3.0 connectivity
- Synchronization with 10 MHz clock reference or PPS time reference
- GPIO and JTAG for control and debut capabilities
- 83.3 x 50.8 x 8.4 mm form factor
- USRP Hardware Driver (UHD) open-source software API version 3.9.2 or later
- GNU Radio support maintained by Ettus Research through GR-UHR, an interface to UHD distributed by GNU Radio

AI-ML module using TensorFlow

- Classification of signals must be accomplished inside the AI-ML model

GUI

- Must start and stop the SDR from collecting data
- Must display the data after it is processed by the AI-ML model

1.3.2 Technical Requirements: Safety

For the safety of the SDR, since it is a very expensive piece of equipment, the following rules must be followed.

- Before plugging SDR into a USB port on a computer, be sure that an antenna is always attached to either the RX2 or TRX port. This is to ensure no short circuits on the FPGA
- Only every attempt to operate the SDR within the defined frequency range listed above
- When handling the B205 SDR, be sure to always have anti-static protective gear and to be grounded

1.3.3 Technical Requirements: Standards

In accordance with part 15 of the FCC broadcasting guidelines, even though there is no intent on broadcasting any signals.

- AM Bandwidth signals are limited to a max of 200 feet
- FM Bandwidth signals are limited to a max of 250 microvolts per meter, measured 3 feet from the source. Within 2.4 and 5.8 GHz, as well as 915MHz or 433MHz.

2. Documents Referenced

Introduction to RF challenge document	Here
Radio Frequency and Modulation Standards	Here
SDR Documentation and Resources	Here
GUI Reference Documentation	Here
TensorFlow Documentation	Here
GNU Radio	Here
Comparing PyTorch with TensorFlow	Here
AI-ML Learning Types	Here
IEEE 802.11 Standards	Here

Table 1: Reference Documents Table

3. Requirements

The requirements on this project are loose in definition as it is a challenge to complete a broad task. The requirements described to us are the following:

- The RF signals must be sweeped and stored with the help of the Software Defined Radio provided.
- The data must be recognized, classified and sorted (FM waves, TV satellite signals, cellular/bluetooth signals, and Key FOB signals) with the help of an AI-ML module.
- The sorted and processed data must be presented in an easy to read manner by the GUI.

3.1. System Requirements

The system requirements of this project ensures a smooth transfer of data from one element to another with errorless transition and processing to reduce the amount of bugs and improve time efficiency:

- The system must be able to start with the GUI, which tells the SDR to start collecting the RF data and format it into a .dat file. This can be done by specifying the output type within GNU Radio.
- The AI-ML module within the system must be able to read this .dat file, feed it through the program, and organize the RF data into the different types such as AM/FM, TV signals, bluetooth, etc.
- The GUI must be able to tell the SDR to stop collecting, and once the AI-ML program is done processing the data it will be displayed on the GUI in an easy to read format for the user.

3.2. Hardware/Data Collection System Requirements

The hardware/data collection aspect solely relied on the SDR. It is responsible to collect all the raw data which would be needed for further use:

- A SDR (software defined radio) shall be used to capture all of the RF frequencies in the designated area that shall be fed into the AI-ML module,
- The SDR must be able to detect and record frequencies between 88MHz (AM signals) on the low end, and up to 6GHz (wifi) on the high end. With everything in

between (TV, FM, Key Fobs, bluetooth, ADS-B and cell phones) being recorded as well.
- The SDR must sweep the frequency spectrum for all available signals when prompted by the GUI for a set amount of time, then supply the AI-ML with the collected data.

3.2.1. Hardware Requirements

We do not control the physical requirements of the SDR. It will be provided and the physical specifications are set. Computers and monitors used for GUI do not have defined requirements, as long as said computer can efficiently compute and run the AI-ML it shall be substantial.

3.2.2. Software Requirements

In order for the SDR to run, a computer needs to run the Ettus Research UHD software, which is the USRP Hardware Driver. The UHD software offers cross-platform support for multiple industry standard development environments and frameworks, including RFNoC, GNU Radio, LabView, and Matlab/Simulink. For this project, we are using GNU Radio.

In order to use the UHD software, there are multiple different dependencies that have to be downloaded and installed first. First, a computer that runs Linux as its operating system is needed. We are actually using Ubuntu 20.04, but it is based in Linux. The UHD dependencies include CMake, Boost, LibUSB, Python, Mako, Doxygen (optional), and NSIS (optional).

The SDR collects the data we need to analyze, it is our job to figure out how to transfer the collected data to the AI-ML module we created.

The total data transfer speed for the hardware/software requirements is 5.0 GBps. This is obtained from the SDR specifications document where it details the exact method of data transfer. That being a USB 3.0 connection, with the max data transfer speed of 5.0 GBps. Realistically the data amount will be less because the USB cord also powers the SDR.

3.3. AI-ML Subsystem Requirements

Based on specifications that we give it, our AI-ML module will receive raw data supplied from the SDR and recognize and classify the different signals that were collected.

The data will be inputted in an .dat file format, the program must be able to efficiently process the data from that format before producing I/O graphs.

The AI program must be able to then classify the frequencies and if it runs into any anomalies that results in a skewed representation of the data it must be able to resolve it on its own.

The program shall then classify and sort the signals. After classifying and sorting the signals the data will be sent to the GUI to be presented.

3.3.1. Hardware Requirements

The trained AI-ML runs purely on software, an untrained model will take a computer a period of time to train and get functional. Once the model is trained it is as simple as plugging in the trained model to accept the SDR data that has been supplied by the GNU radio.

It is however, required that the computer that runs the AI-ML has a GPU that is able to support the use of the libraries and run smoothly without stalling or being overloaded which would cause the program to crash.

3.3.2. Software Requirements

The program must run the latest version of Tensorflow and Keras library. The system must be efficient in terms of time complexity considering the load of input data.

The output data produced by the program must be readable and understandable by the user. Before the AI-ML is put into action the program must be trained thoroughly to update the directory to make sure that all detected frequencies are identified.

3.4. GUI Subsystem Requirements

The GUI portion is going to be extremely simple. From the user's perspective all they shall see is a start button, a stop button, a pause button, the data after it is sorted by the program, the latitude and longitude of the signal, and the time and date. Everything shall be done in the background.

The GUI shall be the only interface the user sees and provide all the information. The GUI ties together the AI-ML, GNU radio, and the SDR so they are all controlled by these buttons listed in the GUI interface.

The end goal of the GUI is to display the data after it is run through by the program. The user shall be able to start the program, pause it, and start it back up until they do not need anymore data.

3.4.1. Hardware Requirements

The GUI shall include a start button, a stop button, a pause button, and a display. Other than these features, everything else would be complicating the project.

3.4.2. Software Requirements

The GUI shall be able to take the user input that starts the program and should be able to display the end result of the program. The user should also be able to hit a button and stop the program whenever they want. The user input of clicking start shall start the SDR, this pulls in RF signals.

After the SDR pulls in data, the GNU shall organize the data so the AI-ML can compile the data and sort out what RF signal is what. This then shall be sent to the GUI for the user.

3.5. Literature Research

Using the cited NIH(National Health Institute) research paper, it mentions how PyTorch vs TensorFlow and why it is much more user friendly for its grad students who are new to using AI-ML.

PyTorch has plenty of readily available documentation and execution time being inherently fast. This is due to their ‘’simple design is better.” They pride PyTorch on being straightforward and the meat and potatoes of AI-ML.

TensorFlow is the second listed in their list of recommendations, the consensus of TensorFlow is that mistakes are harder to find and troubleshoot, while also being adept to more experienced users of AI-ML. Where TensorFlow falls short is that it is not very user-friendly. PyTorch and TensorFlow are said to be on even ground when it comes to feeding data into the network, and data being fed out of the network.

PyTorch falls back in larger and more complex designs being a hindrance on the AI-ML, but TensorFlow excels in more complex projects.

There are 2 different types of AI-ML learning: Supervised and Unsupervised Learning.

Supervised Learning is when the AI–ML has given reference data, so when the AI-ML is performing its computation it can compare its current/computed data to its reference data. With that there are 2 functions of Supervised learning, Classification and Regression(prediction). Think of Supervised learning as taking a test with an equation sheet. You likely have all the information needed, but it's up to you to fill in the blanks.

Unsupervised Learning is just the opposite of Supervised Learning, the AI-ML is given no reference data. Where the AI-ML takes predictive guesses on groups of data, this being useful for when there is not a numerical answer or a singular answer. The 4 types of functions that Unsupervised learning are applied to are Clustering, Association, Anomaly Detection, and Artificial Neural Networks.

Other than AI-ML language and learning options. There is a more nitty gritty debate of which AI-ML algorithm to use. There are dozens of different algorithms all tailored to specific uses. In the application of our AI-ML, there are 2 that are described to be most suiting. This being the Decision-Tree algorithm and the Naive-Bayes algorithm, being a supervised learning type.

Decision-Tree algorithm is exactly as it sounds, it will compare a set of data to a standard or window that it possibly could meet. Then the AI-ML will decide whether or not it falls close enough to the parameters to be classified under this category. If not it will go up the tree comparing the data to each parameter. Once a parameter(s) has been met that the data matches, it will be classified as such. That being said this is a supervised method of AI-ML, because the AI-ML is given data and parameters to compare the data too.

Naive-Bayes algorithm is a statistical based algorithm that uses conditional probability and Bayes-Theorem to compare the data given to similar data that meets similar

criteria. Naive-Bayes is credited with being terrific for data sets with lots of complex values to be computed. This being said it seems tailored for our use, but calibrating the algorithm to peak accuracy would seem to be a time consuming task. That being said, this is also a supervised method of AI-ML. because the AI-ML is given parameters and will use statistical analysis to compare the data on how similar the data is to our parameters.

Lastly is the CNN or cognitive-neural-network. A CNN works by initially observing and guessing outcomes based on known data results. But over time it will remember and learn from its errors, progressively getting better and better at predicting the correct outcome. In our case it will be predicting the type of signal we have detected. There is a soft threshold to the ability of the CNN, AI-ML's with applications similar to this project can predict near 90% accuracy. But with that comes very fast computing, and very little in terms of computer resources. With training the AI-ML model, it takes a large period of time and is very resource intensive to train, but once the model is trained it can be saved, sent, and loaded up onto a separate computer.

GNU Radio Companion is an open source software developmental toolkit that provides block structures to implement software radio designs. These "blocks" or components are run alongside analog hardware through a computer or embedded system. This software is readily available to use various styles of software defined radio devices as long as the proper files are installed. It also has the capabilities to simulate environments through pre-installed blocks and data files.

3.6. Alternative Approaches and Formal Decision Making

Initially the plan was to use PyTorch as the approach to the AI-ML. After a large amount of work to get PyTorch properly working with no real progress we decided to approach Tensorflow. Tensorflow was simpler to get setup, but still very strenuous to get functional. Thus came the formal decision to use Tensorflow instead of PyTorch

With the GNU Radio our first approach was to use one of the Windows based laptops to use GNU Radio. But after a very long time of no progress, and a confirmation email from Ettus Research themselves it was concluded that using the SDR with Windows 10 was already difficult, and with Windows 11 it hadn't been done before. Consequently we had to research dual-booting one of our laptops to Ubuntu 20.04 linux OS, this being our only choice we pursued it and got the GNU Radio to work. So the formal decision came along to use the Ubuntu 20.04 for our GNU Radio.

4. Design Overview

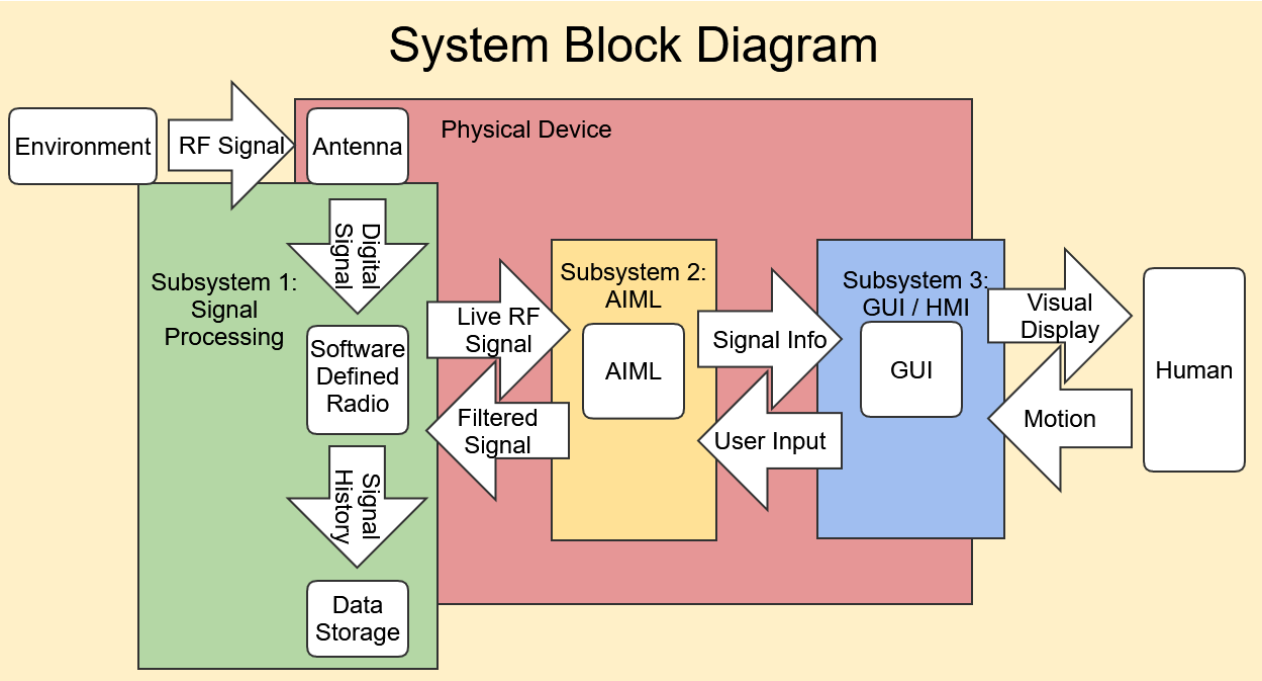


Figure 1: High Level Representation of RF Challenge

4.1. Hardware/Data Collection Subsystem Design Overview

The purpose of this subsystem is to use the SDR and GNU Radio to capture all of the RF frequencies in the designated area. The SDR will take RF signals as input using an antenna and then output a .dat file for the AI-ML to process.

There are no design options for this subsystem as it is provided to us.

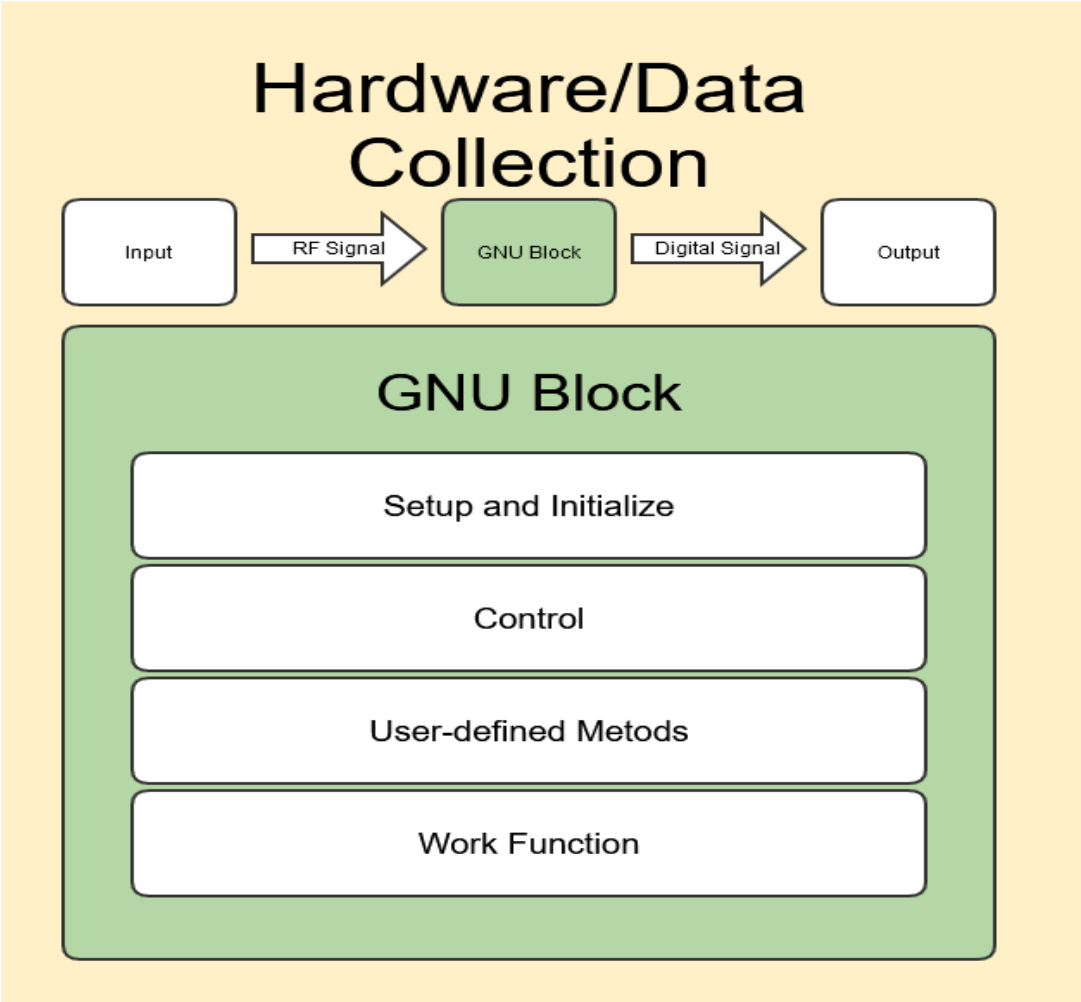


Figure 2: High Level Representation of Interfacing with SDR

4.1.1. Hardware Design

There is no hardware design for this subsystem as the SDR is provided to us, and the SDR is the only piece of configurable hardware. The SDR loaned to us is the Ettus Research USRP B205mini-i.

Figure 3: USRP B205mini-i

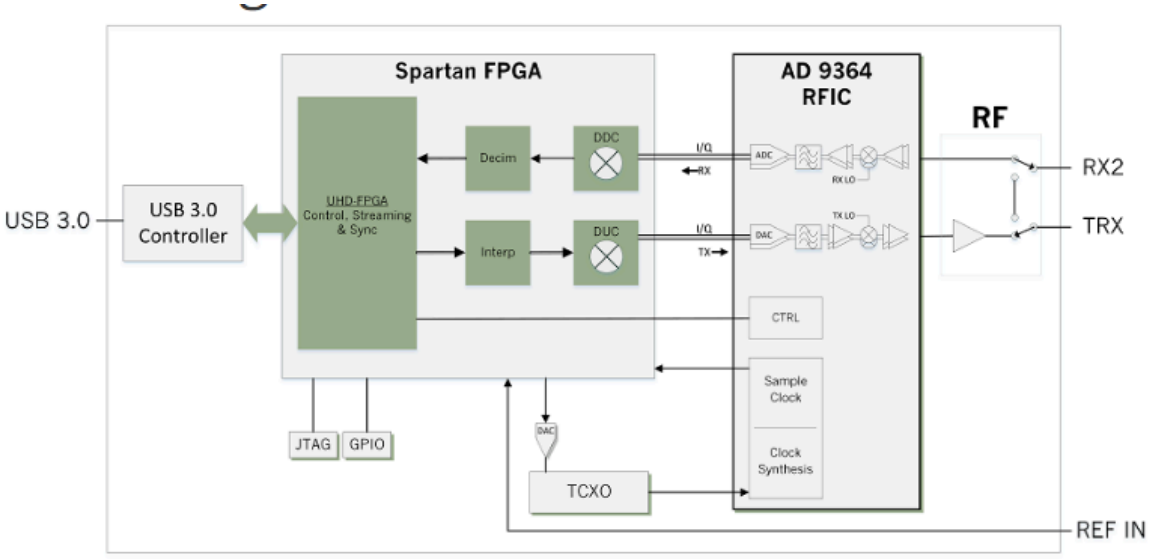


Figure 4: USRP B205mini-i Block Diagram

4.1.2. Software Design

The three softwares working with each other shall be the GNU Radio, AI-ML, and the GUI. The GUI shall give GNU Radio instructions on what to do(start/stop/filtering). Referencing the GNU Radio Companion flow graph, MPSK SNR Estimator is used to calculate estimations of the signal SNR. This is calculated using the signal and noise power estimate to determine if the SDR is receiving static noise or actual frequency data. The code within “MyPythonBlock” is used as a sweep function. The four variables that must be accounted for are center_freq_start, center_freq_stop, sample_rate, and num_samples. The center_freq_start/stop variables are used to define the frequencies in which the SDR starts and stops collecting data. The center frequency is incremented after data collection at a frequency is completed or determined to be static noise and skipped. Sample_rate is the indicated number of samples taken in by the SDR per second. Num_samples refers to the amount of samples of data taken at each qualifying frequency. While the AI-ML will run as its standalone where it imports the SDR data, categorizes data, and organizes data. Lastly the AI-ML will export the organized data to the GUI to display the final result. Block diagrams of the system(s) shown below.

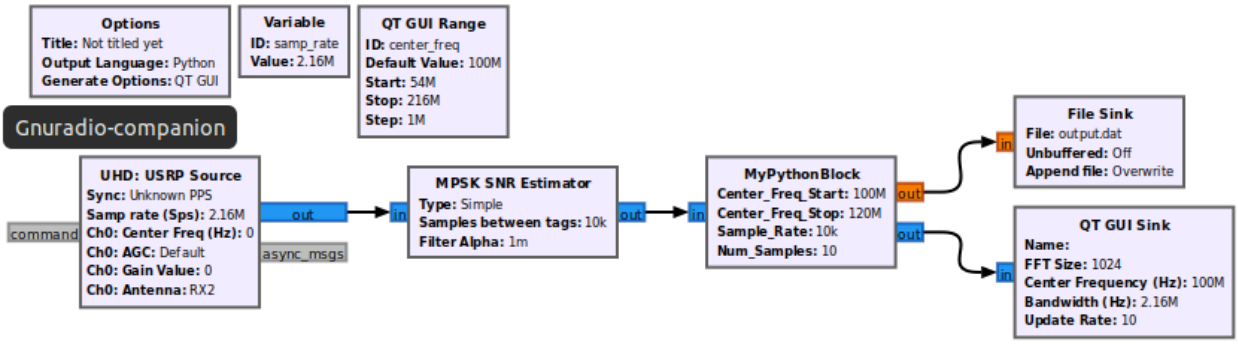


Figure 5: GNU Radio Block Diagram

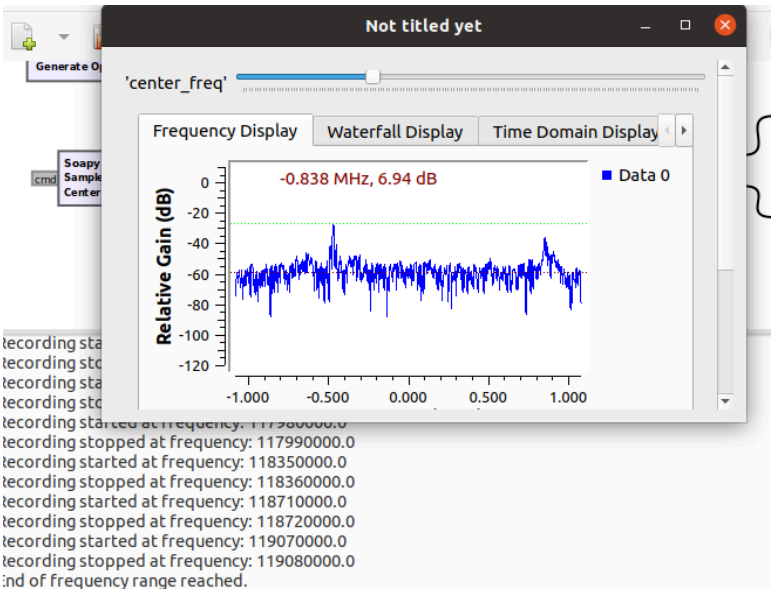


Figure 6: Output of QT GUI Sink Within GNU Radio Block Diagram

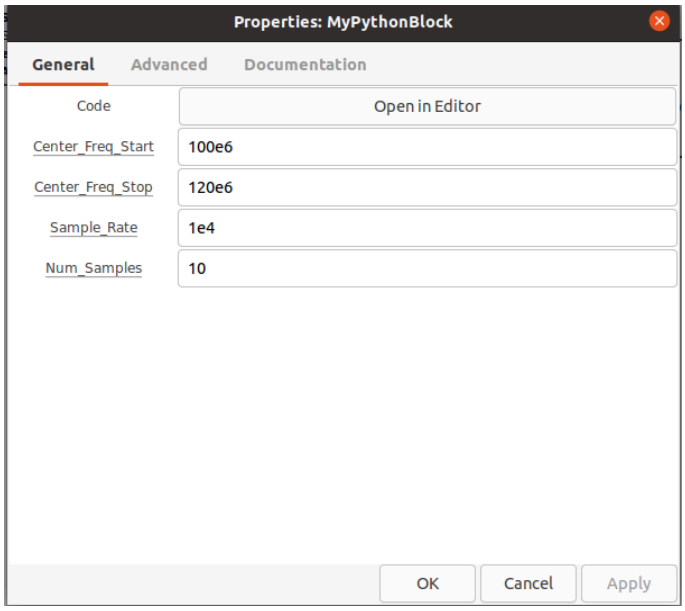


Figure 7: Variables Used Within the Custom MyPythonBlock

4.1.2.1. Software Use Case Diagram

The software use-case is not applicable as the software is required for the overall functionality of this project.

4.1.2.2. Software Modules

The three overarching software modules that will work together are GNU Radio, the GUI, and AI-ML. The GUI and GNU Radio will work together to start and stop the data collection, while GNU Radio sends the data it collects to the AI-ML. Outlined on the diagrams below.

4.1.2.3. Software Sequence Diagram

The diagram below is the sequence diagram of how each subsystem of software will work and display the data.

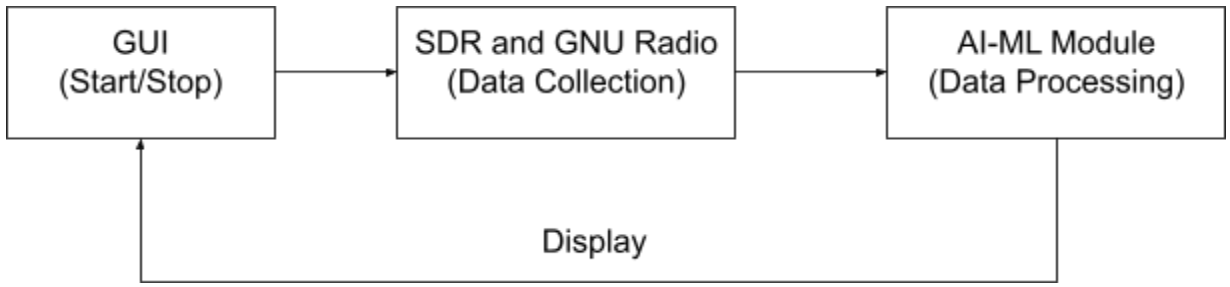


Figure 8: Software Sequence Diagram

4.1.2.4. Software tools

GNU Radio will be used to automate the data collection process of the project. We are going to use it to sweep through the RF spectrum and gather the prominent RF signals

as it goes. Once it is done collecting it will format the data into a .dat file and send it over to the AI-ML module for processing.

4.1.2.5. Software State Machine Diagram

The diagram below is a state machine of how each portion of the software interacts with each other.

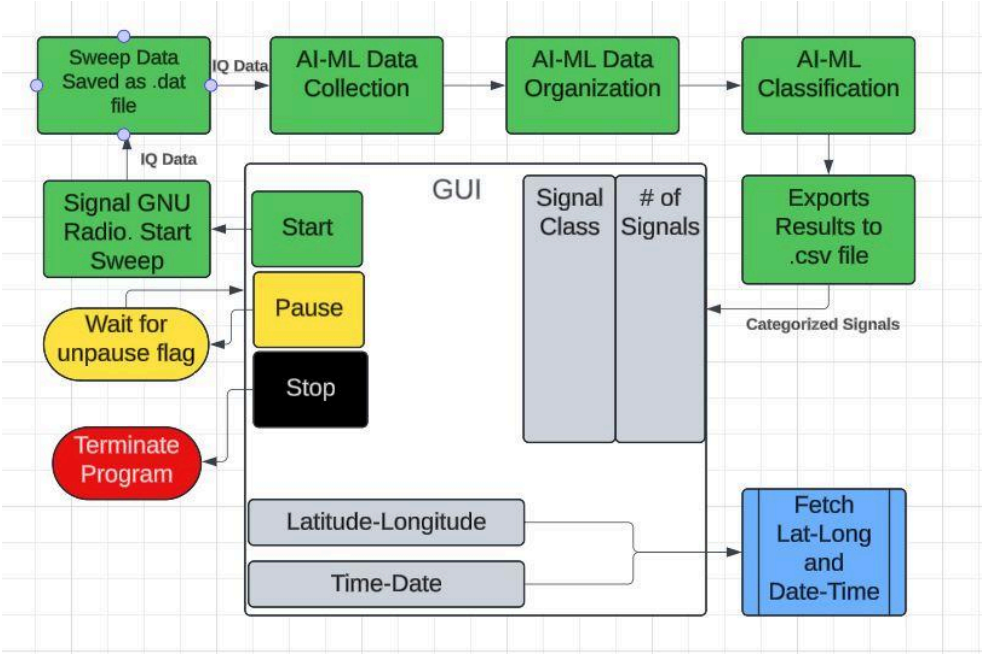


Figure 9: Software State Machine Diagram

4.1.3. Integration Specification

The first thing to do is formulate a parameter list from the given data/material, this shall accelerate our initial stages of creation when it comes to the AI-ML development.

The background research on the SDR itself shall need to be performed before we even receive it from our sponsor. The more we know about it before we have it the easier the next steps will be.

Once we actually have the SDR, we need to understand how it physically records the data we need and how it is stored within the SDR.

After we know how everything is stored, the final step of this subsystem is to figure out how to export the data we categorized to our AI-ML module in the form of a .dat file, as well as display the results to the GUI.

4.1.4. Subsystem Test Plan

The GUI system has a start button, pause button, a stop button, as well as a section to display the results of the AI-ML. The GUI will need full functionality before it is integrated to control either the data collection or the AI-ML for the classification. The verification process is detailed below.

The AI-ML system has 2 subsections(DataGeneration and Accuracy Testing) that shall need to work in unison to perform the overarching task, each subsection will be individually tested and confirmed to work. Once the 2 are completed the total AI-ML will be verified and can be integrated into the full system. The full verification process is listed below.

The GNURadio/SDR system contains the block diagrams and functionality in order to receive and collect the data. This data is then fed into the AI-ML in order to be parameterized and displayed for the user to view. In order to test the functionality of the GNU Radio Companion program, the program will be constructed as mentioned previously. After construction and compiling, the program will be run for a minute then powered off. The created data file will be inspected to ensure data is collected properly.

Reference appendix F for VCRM.

4.1.4.1. Software Verification

To test the GUI and verify that it is functional, we will first test to see if the GUI pulls in data from a test file when clicking on the start button. This shall pull up the data file’s information and close it when the user clicks stop. That is the first step to test the GUI. The next step would be to use a simplified device that pulls in information and see if the GUI is able to keep up. Once we follow these steps, this will ensure that the GUI is fully capable.

Subsystem 1 for the AI-ML is the DataGenerator. The data generator works by pulling data in increments instead of pulling large amounts of data at once. This eases the stress on the computer's memory and opens the door to less robust computers. This test is purely functional, either it works or it does not work.

Next is the AI-ML training and testing. The AI-ML will be fed data with known results, as to let the AI-ML learn how to predict results. The testing of the AI-ML comes when it is done being trained. It will test itself against known data, essentially predicting the results and then telling the user the accuracy percentage. This testing portion is completely up to the user, if the user decides the accuracy is adequate enough to integrate, then that will be the AI-ML model used.

4.1.4.2. Hardware Verification

The SDR can realistically be tested anywhere. Once we understand how the SDR collects and stores the data and how to access it, we can take it any place where RF signals are present.

4.2. AI-ML Subsystem Design Overview

The purpose of this subsystem is to identify and classify the RF signals that are collected by the SDR. Once the signals are collected, the AI-ML module will then sort the signals based upon a list of parameters that we specify. It makes use of various dynamic sorting methods with the help of neural networks to sort the signals and group them based on similarity and make classifications to identify the frequency type. After the signals are sorted it will then send the list to the GUI for the user to read.

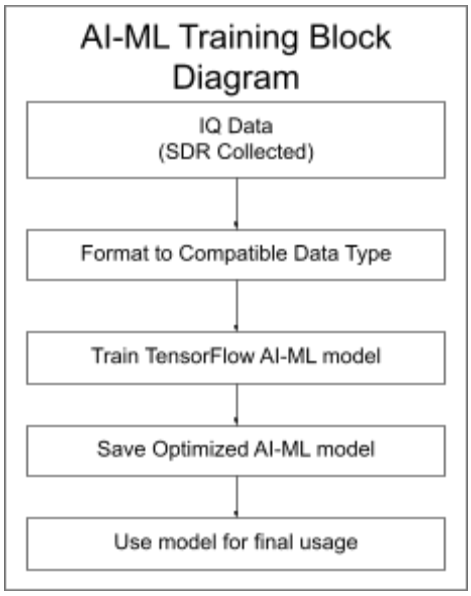


Figure 10: Roadmap for Successful Implementation of AI-ML

```
model = Sequential()
model.add(Conv1D(64, 3, activation='relu', padding='same', input_shape=(1024, 2)))
model.add(BatchNormalization())
model.add(Bidirectional(LSTM(100, return_sequences=True)))
model.add(Dense(200, activation='relu'))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Creating Data Generators
#play around with the batch size, recoomend keepig shuffle on.
train_generator = CustomDataGenerator('training_data.dat', batch_size=8, shuffle=True)
test_generator = CustomDataGenerator('testing_data.dat', batch_size=4, shuffle=True)

# Train the model
model.fit(train_generator, validation_data=test_generator, epochs=1000)
#change the epochs when dealing with more or less data
```

Figure 11: Visualization of Training the AI-ML Model (Layers)

4.2.1. Hardware Design

There is no hardware design in this section due to the subsystem containing strictly software.

4.2.2. Software Design

The three softwares working with each other are the AI-ML, GNU Radio, and the GUI. The GUI shall give the GNU Radio the instruction to start its sweep, while also telling the AI-ML will run. Where it collects the SDR data, Categorizes Data, and Organizes Data. Lastly the AI-ML will export the organized data to the GUI to display the final result.

4.2.2.1. Software Modules

The three submodules of the AI-ML are the Data Generation, Data Categorization, and Data Organization. Data Generation is used to regulate how much data is fed at once to the AI-ML model. Data Categorization is the bulk of the AI-ML itself, this is where the Data that was priorly mentioned will be fed into the model. There is not much to talk about for this module since it is a black box, the user does not have much influence over it. Lastly is Data Organization, once the data is categorized it will be organized into a format that will be easily readable by the user. As well as easily imported into the GUI.

4.2.2.2. Software tools

The AI-ML has a lot of tools within it to use. To start we are developing the code within Microsoft VSCode. Within VSCode there is a large number of libraries to installNumpy, Tensorflow, Keras, h5py, Pandas, ScikitLearn, etc. Each one is a python library that has a set of tools used within just the AI-ML model. Each software subsystem has its own libraries needed to accomplish its given task. Reference Appendix H for a full picture.

4.2.2.3. Software State Machine Diagram

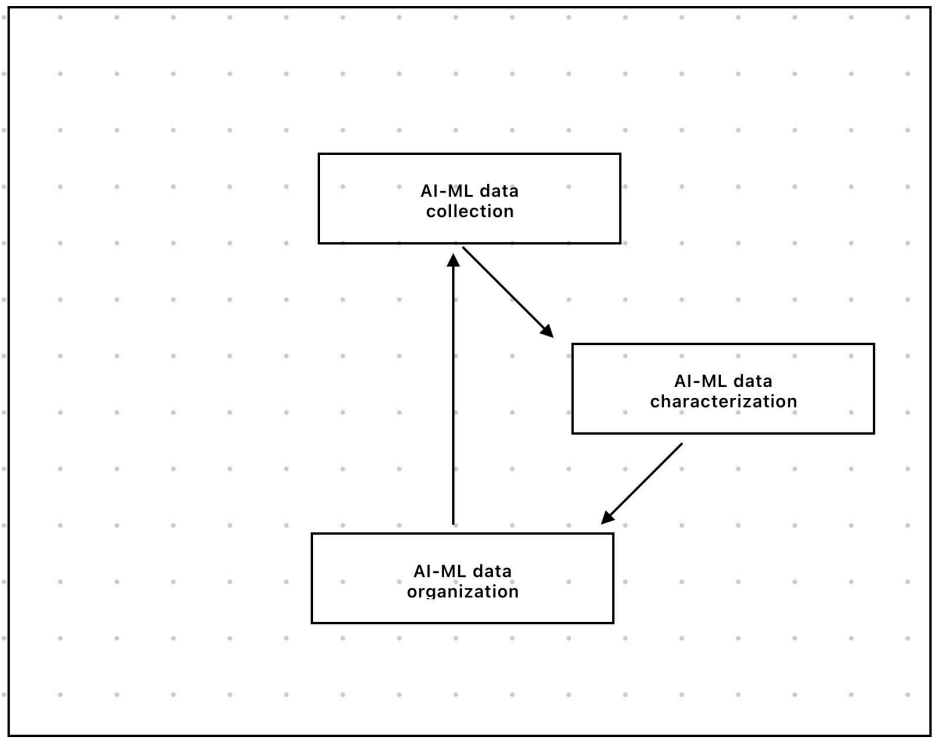


Figure 12: Software State Machine Diagram

4.2.3. Subsystem Test Plan

The AI-ML system has three subsections that shall need to work in unison to perform the overarching task, each subsection will be individually tested and confirmed to work. Once all three are completed the final AI-ML will be verified. Refer to Appendix F for the VCRM

4.2.3.1. Software Verification

The AI-ML has multiple subprograms that fall under it, including the Data Generation, Data Categorization, and Data Organization. All of these subsystems can be allocated under the Automation Software Subsystem and can only be truly verified by testing their ability.

Data Generationis first to be tested and completed, this shall be as simple as collecting the data through GNU Radio, formatting the data in a format that is compatible to the AI-ML model. Then being able to visually see the collected data. Something as simple as a print() statement can confirm our test here.

Next would be the Data Categorization, this is where the bulk of the AI-ML will be put to use. The AI-ML shall take the collected data and use the developed and trained model to compare and categorize the fed data sets. This section will take the longest to calibrate and testing shall be extremely thorough.

Lastly is Data Organization, this is a less intensive portion in terms of computing power. Once the data is categorized, the data shall be organized depending on the category put under. The verification of this step will be purely visual, being able to print and view the organized data will be easy enough.

4.2.3.2. Hardware Verification

There is no hardware verification since this subsystem is strictly software. Every piece of the system is software based, and no hardware components used are interchangeable. Essentially there was no hardware selection or design, this project has been purely software sided in terms of creation.

4.3. GUI Subsystem Overview

The purpose of this subsystem is to display the data after it goes through the AI-ML module in a manner that is easy for the user to read. The user shall be able to start the program from the GUI as well as stop it after it starts. Below is shown a sample of the skeleton framework of the GUI, as you can see there are start, stop, and refresh buttons. As well as a section dedicated to the display of results. Note that for now we have loaded the GUI with fake data just to test that the framework is working correctly.

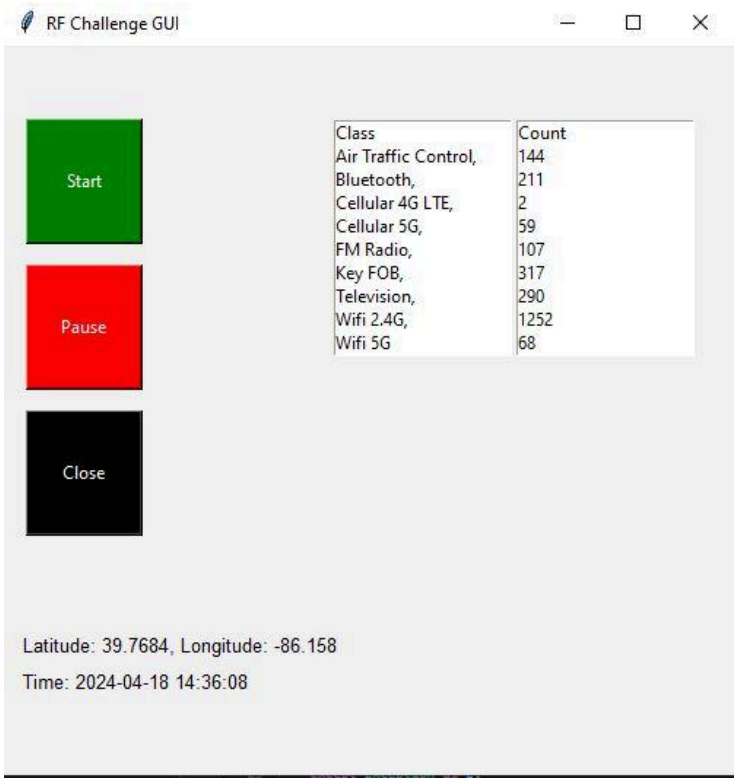


Figure 13: GUI Diagram

4.3.1. Hardware Design

We originally had a GUI that was going to have a wavelength display, parameter list, list of present signals, filter, and a start/stop button. After talking with our sponsor they told us they wanted this to be much simpler. We then cut it down to just the list of present signals, start button, pause button, and stop button. We later added the latitude and longitude of the signal as well as the time and date.

4.3.2. Software Design

The three softwares working with each other shall be the AI-ML, the GNU Radio, and the GUI. The GUI shall give the GNU Radio to start the sweep and simultaneously give the AI-ML the instruction to start doing its three step process. Once the AI-ML

completes and organizes the data, the AI-ML will export the organized data to the GUI to display the final result.

4.3.2.1. Software Modules

The two overarching software modules that will work together are the GUI and AI-ML. While the GUI and AI-ML will work together to send and receive data from each other, the GUI will be displaying these updates as shown above. While the GNU Radio/SDR will just act as a function, it will execute and produce data in a .dat format. It does not return any meaningful data to the program itself, but instead produces it and stores it separately.

4.3.2.2. Software Tools

The GUI uses python to build the interface and connect the other softwares to it. Within python, the group uses tkinter as a GUI toolkit. This allows our python script to be used as a GUI.

4.3.3. Integration Specification

The simplest portion of the GUI is the ability to start/stop the program. This is the first thing that we need to get working before we worry about anything else.

The next step is to design the filter on the GUI. The user needs to be able to select the parameters they need before they run the program.

Lastly, and most importantly, we need to figure out how to display the signals on a screen for the user to see. This needs to be the “final result” screen of the project that shows exactly what RF signals are in the environment.

4.3.4. Subsystem Test Plan

The GUI has two different subsections. The first is the GUI being able to interact and control the AI-ML script, and the second is the GUI receiving and displaying categorized/organized data to the user to view.

4.3.4.1. Software Verification

The GUI has two separate parts within it. The interaction/control of AI-ML and the displaying of the categorized and organized data.

The control of the AI-ML is simply verified by being able to start/stop the AI-ML script and SDR control by a simple press of a start/stop button. While also being able to select the filters we want to use, this shall reduce the total amount of signals categorized and organized by the AI-ML. The verification of this subsystem shall just be simple as a demonstration.

The GUI receiving data from the AI-ML and displaying it, shall be tested simply by demonstration.

4.3.4.2. Hardware Verification

Software and hardware verification for the GUI are almost identical. The start/stop button should simply start/stop the program and the display should display the present signals. Both of these shall be tested simply by demonstration.

4.4 Design Challenges

Design challenges have come in many shapes and sizes throughout the duration of the project. Namely the design requirement of this entire project being controlled through 1 singular GUI presents its own dictionary of problems. Namely with python compatibility between the 2 other complex subsystems.

Other design challenges that have arisen are items like the GNU Radio being on linux, it was not our first intention to be executing the final project on linux but with the compatibility to windows our list of choices was short. With that comes a blessing because there are a multitude of resources online relating to our applications that have helped with overcoming the learning curve and getting GNU Radio to work.

Lastly with the AI-ML model. There has been a library of design challenges that came with just creating a working model that will compile and classify modulation types took

weeks of experimentation with data types as well as structure of the model in general. As well as the fine tuning to produce a reasonably accurate prediction.

5. System Test Plan

When the group is entering the testing phase, we will begin by picking up radio waves in a controlled environment such as a basement to avoid exterior interference. As the project progresses, the tests will contain more input devices as well as various radio frequencies. The final stage of testing will involve accompanying our sponsor to a US Military base to ensure proper space for testing. This station for testing will also require the use of the AIML to identify hidden radio waves.

5.1. System Integration Test Plan

There are four aspects in this project that need to be tested before the whole system can be tested all together:
1. SDR functioning: make sure that the SDR is able to read and detect frequency
2. GNU Radio frequency sweep: The GNU radio should be able to perform successful frequency sweep and produce quantifiable frequency output.
3. AI-ML functioning: The AI-Ml should be able to take in the input files from the radio and output back with identifiable frequencies. Too many unidentified frequencies would mean the model hasn't been trained enough.
4. GUI:The GUI should be able to display all the necessary output from the AI-ML and without any bugs.

5.2. System Validation

The demo for the system will take place at 9 am, 26 April, 2024 in the lecture hall in the IO building. This will be demonstrated to the sponsors and the class. Beforehand there will be testing and validation as a team to ensure that the system is working adequately prior to a demonstration to our shareholders and peers.

6. Potential Problems and Bottlenecks

If there is a large range of electromagnetic spectra then there is a change of excess input for the RAM to handle which may result in data loss.

- Mitigation: Use a specific bandwidth one at a time in order to be able to crunch the data one at a time.

1. If the timing between the components is not in sync then there is a chance of mismatch in data between components leading to a false result
    - Mitigation: Use synchronized clocks that oscillate uniformly through all of the devices
2. If the environmental factors are too harsh then the readings could be interfered with or a complete device failure
    - Mitigation: use protective equipment on the sensitive devices and mostly try to use it on a clear day.
3. If there isn't enough sample rate then the IQ/frequency resolution would be inadequate
    - Mitigation: Use sample rate for the highest frequency of interest

6.1. System-Level Potential Problems and Bottlenecks

The main risk at a system level would be that the whole system cannot be controlled by one button. This would mean that the GUI cannot start the SDR which then leads to the AI-ML not being able to parameterize the data and then not being able to show the collected data to the user on the GUI. Another possibility would be it not working at the AI-ML level and this would then pull data in but not parameterize the data. This would relay unorganized data to the user and not be effective.

Another risk at the system level would be the python languages not being compatible with each other. With every group member working in python but certain softwares are not using fully updated python libraries. This could put the project at a standstill if these languages and libraries are not compatible.

Another risk at a system level would be one single laptop not being able to hold all libraries and softwares at one time. Similar to the previous listed risk, every member is working on their own software on their own device.

6.2. SDR/GNU Radio Subsystem Risks and Opportunities

Condition	Effect	Likelihood	Factored Risk
If the USRP B205 Mini-i is non-compatible	Then the schedule will be delayed by 5 days while an alternate SDR is delivered and an additional expense of $150 will be incurred to purchase a new SDR that is compatible.	10%	12 hours $15.00
If the selected python version is not compatible with intended use	Then the schedule will be delayed by 2 weeks to convert into a new version.	25%	24 hours $0.00
If the system is not accurately enough parameterizing radio signals	Then the schedule will be delayed by 1 week in order to communicate with the sister-group and sponsor.	20%	1.4 days $0.00

Table 2: Risks and Opportunities Table

6.3. AI-ML Subsystem Risks and Opportunities

The AI-ML has a lot of risks within the scope of its ability. Firstly is producing an accurate, fast, and reliable form of classification. If the AI-ML can be trained to an effective level of accuracy then it will be fine. On the contrary, it is very tedious and difficult to predict and train an effective model. Think of the AI-ML as a black box of computing power. The user does not have much information about what is going on within the model as it trains, the most a user gets in information is the current estimated accuracy. Which is only based on training data, so as to deliver an effective AI-ML model it will require a large amount of modifying and testing to get the most effective results.

Once the model is trained, it will be fairly straightforward to integrate into the system. The largest obstacle is getting the data from the SDR in a format that is native to the model. Since the model will only accept the same data format as it was trained with.

Within the communication of the GUI, there are no foreseeable problems for its integration with the AI-ML. Being straightforward and no real problems foreseeable.

6.4. GUI Subsystem Risks and Opportunities

The GUI does not have as many risks as the other subsystems. The main risk with the GUI is not being able to tie all applications together to show the data. The risks the user to not see any data that has been pulled and parameterized. This could go wrong during integration but once it has been coded properly, it should not differ at any point. The main limitation with this subsystem is that it can not be continuously updated while data is being pulled. This could hinder information shown to the user however, with the data always changing, this could complicate the data shown

7 ABET Outcomes

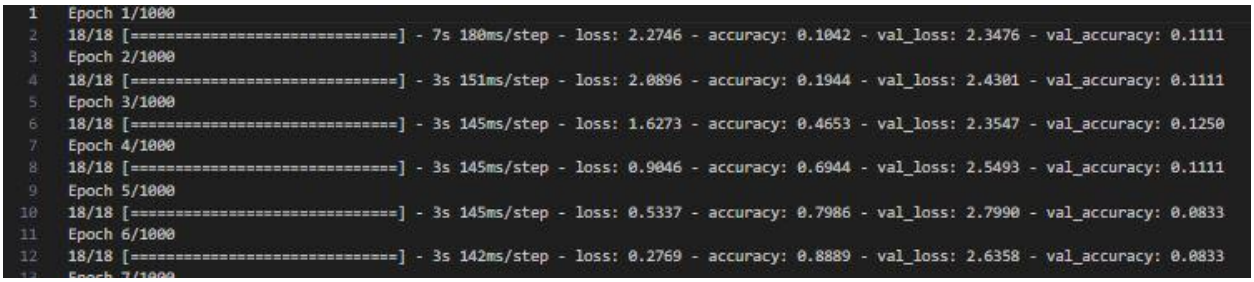
When it comes to ABET outcome 2, which is an ability to apply engineering design to produce solutions that meet specified needs with a consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors, the entire motivation behind the progress is supported. The whole purpose of this project is to give our US Marines a method to simulate RF traffic in a large city while training in remote or rural locations. Completing this project in turn will make United States citizens as well as citizens around the world safer because our marines will be better trained. This will then directly correlate to economic benefits because our military is better equipped to deal with situations when they arise.

As far as ABET outcome 7, which is an ability to acquire and apply new knowledge as needed using appropriate learning strategies, essentially every portion of this project is applicable. None of us had much prior knowledge about RF signals, and most of us did not have any experience with using SDRs, GNU Radio, TensorFlow, etc. As we all went along through this project we had to learn things that we did not know beforehand. Looking back to where we were when we started, we have a much greater understanding of the concepts we had to use during the course of this project.

7.1 Analysis and Results

We were successfully able to integrate the GUI with the AI-ML model as well as the SDR. At the push of the start button on the GUI, the SDR will sweep through the RF spectrum and collect data in a .dat file. This data will then be fed to the AI-ML model and it will sort, classify, and organize the data. This data will then be displayed on the GUI for the user to see. At this time, the AI-ML is classifying data with close to a 90% accuracy. The model works by pulling 1024 I and Q samples from the .dat file at a time, then feeding them to the model. Keep in mind the data used to train the model is a small set of sample data(about 144 sets of samples) that was collected using our SDR, with that the complete accuracy and furthermore the legitimacy of the data isn't quite known.

Figure 14: Accuracy Percentage Screenshot



7.2 Conclusions and Further Work

The GUI subsystem is able to control both the AI-ML model as well as the GNU Radio/SDR subsystem, while also being able to temporarily pause the entire system. Once the user presses the start button on the GUI, the process is all automated to collect data via the SDR, then the AI-ML model takes the data in, classifies it into 1 of 9 possible signals, then the results are displayed onto the GUI in order of highest occurrence. Along with Date-Time, and Latitude-Longitude coordinates to keep record of position. Our next steps include integrating the power level of the signal to be recorded and saved for the user to view. We also would like to include better latitude-longitude positioning, currently the system uses the users IP address to bring data in for the latitude-longitude coordinates and that is not as accurate as we would like it to be. We also would like to use a confirmed and tested data source for AI-ML training to make our AI-ML more realistic and accurate. The final stretch of this project is to finish up the full functionality of each subsystem, and combine them all into the final product. In the future, the real world model used in legitimate applications should be trained off verified data, and a large amount of it. Looking at Figure 11 and Figure 14, you can see that the model itself is fairly complex, but with the small data size the training epochs was set to 1000. With that the accuracy grew, but as a tradeoff the model gets too native to look at repeated datasets, thus likely lowering real world accuracy. The model is very easy to overtrain and overfit, especially when training off of a small set of data, so a larger set such as 5-10Gb would be a good idea in terms of real world accuracy.

8. Appendices

Appendix A: List of Abbreviations

USRP	Universal Software Radio Peripheral
SDR	Software Defined Radio
AI-ML	Artificial Intelligence-Machine Learning
GRC	GNU Radio Companion
RAM	Random Access Memory
GPU	Graphical Processing Unit
GUI	Graphical User Interface
RF	Radio Frequency
AM	Amplitude Modulation
FM	Frequency Modulation
ADS-B	Automatic Dependant Surveillance-Broadcast
Hz	Hertz

Table 3: Abbreviations Table

Appendix B: Project Schedule

Project Start Date: 22-Aug-23

Project Name: NSWC Crane - RF Challenge_2

Week starting:

#	Activity	Assigned To	Start	End	Days	Status	%Done
1	Project Kickoff	Assignee Name	Aug-22-23	Aug-25-23	4	Not Started	100%
2	Create requirements document	Jacob	Aug-25-23	Sep-22-23	21	Complete	100%
3	Create design document	Drew	Sep-22-23	Oct-20-23	21	Complete	100%
	Update design document	Vi	Nov-13-23	Nov-17-23	5	Complete	100%
4	Create a simulation	Vi/Sean	Nov-23-23	Nov-27-23	3	Blocked	N/A
5	Design AIML Software	Drew/ Grayden	Nov-27-23	Dec-1-23	5	Blocked	N/A
6	Test and run SDR	Drew/ Grayden	Jan-11-24	Jan-30-24	14	Complete	70%
7	Programming to collect SDR data	Jacob	Jan-24-24	Jan-30-24	5	In Progress	10%
8	Create AIML Algotirm	Vi/Sean	Jan-24-24	Feb-8-24	12	Complete	20%
9	Testing and debuggin the algoritm	Drew/Graden	Feb-8-24	Mar-8-24	22	Complete	0%
10	Testing algoritm with the SDR	Vi/Sean	Mar-9-24	Apr-15-24	26	Not Started	0%
11	Feeding data to directory	Vi/Sean	Mar-16-24	Apr-15-24	21	Not Started	0%
12	Creating GUI	Jacob	Mar-9-24	Apr-25-24	34	Not Started	0%
13	Final Trial, testing debugging	Vi/Sean	Apr-26-24	May-3-24	6	Not Started	0%

Figure 15: Project Schedule

Appendix C: Bill of Materials

Team: Team A: Sean Page, Grayden Johnson Date: 11/16/2023

Sponsor: NSWC Crane, Todd Kuebelbeck, Scot Hawkins Project: RF Challenge

Item	Description	Manufacturer	Part number	Qty	Unit Price	Ext. Price	Notes
1	Software Defined Radio	National Instruments	USRP B205mini-i	1	$ 1,571.00	$ 1,571.00	Provided by Sponsor
2	Laptop	N/A	N/A	1	$ 500.00	$ 500.00	Need computer to run program
3	USB 3.0 Cable(5")	SaiTech IT	B00O13HLJU	1	$ 5.99	$ 5.99	On Amazon
4	Antenna(s)	N/A	N/A	1	$ 200.00	$ 200.00	Possibly need multiple antennas to cover larger RF Spectrum
5						$ -	
6						$ -	
7						$ -	
8						$ -	
9						$ -	
10						$ -	
Total						$ 2,276.99	

Figure 16: Bills of Materials

Appendix D: Lessons Learned

We recognize the value of teamwork, collaboration and communication which would help us in our future projects in the professional world. A major lesson learned was time management. When Sean and Vighnesh were programming the AI-ML and working with other aspects such as working on the SDR which took much longer than we expected, skewing our schedule.

The AI-ML is very robust when applied in the right way, but the largest problem is formatting the data from the sample data used for initial training and the SDR into the correct data format. As well as then designing the layers of the AI-ML to be compatible with said data format. With that there are a multitude of layers to run the data through for categorization and classification. Learning how each one acts and performs in accordance with another is a very steep learning curve. This taught us to be prepared to learn, every task of this project especially the AI-ML portion included a very large learning curve to overcome before we can make much progress. Doing the correct research is important to the executing of such a project.

Grayden and Andrew realized this when we expected the SDR would work automatically without much intervention but did not go how we expected and required significantly more work to make it functional. We went through each team member’s laptops and devices to see if the softwares was compatible. The Software was only compatible with Windows 10 WSL or Linux. Every member's windows were already updated to windows 11. This led us to a standstill because we would have to borrow an outside person’s windows 10 laptop or dual boot a laptop of ours to Linux. This taught us to take into account the worst possible scenarios and be prepared for it when they may or may not happen.

Appendix E: References Cited

Introduction to RF challenge document	Here
Radio Frequency and Modulation Standards	Here
SDR Documentation and Resources	Here
GUI Reference Documentation	Here
TensorFlow Documentation	Here
GNU Radio	Here
Comparing PyTorch with TensorFlow	Here
AI-ML Learning Types	Here
IEEE 802.11 Standards	Here

Table 1: Reference Documents Table

Appendix F: Verification Cross Reference Matrix

Req #	Requirement	Allocation Level	Verification Type
1	Parameter List	System Level	Analysis
2	AI-ML	System Level	Demonstration
3	AI-ML Data Collection	--Automation Software Subsystem	Test
4	AI-ML Categorization	--Automation Software Subsystem	Test
5	AI-ML Organization	--Automation Software Subsystem	Analysis
6	GUI	System Level	Demonstration
7	Interactive GUI	--Robocontroller Subsystem	Demonstration
8	GUI Display Organized Data	--Robocontroller Subsystem	Demonstration
9	SDR	System Level	Demonstration
10			
11			

Figure 17: NSWC Crane's VCRM

Appendix G: Design Options - Pugh Matrix

25 total points - 5 points each

Hardware			
	HackRF one	Ubertooth one	USRP mini
Criteria	P1	P2	P3
Affordable	4	5	4
Mobility	5	5	5
Ease of use	5	5	5
Strength	4	4	5
Effectiveness	5	3	5
	23	22	24

AI-ML		
	Pytorch	Tensorflow
Criteria	P1	P2
Affordable	3	5
Mobility	5	5
Ease of use	4	4
Strength	5	5
Effectiveness	5	5
	22	24

Figure 18: Pugh Matrix

Appendix H: Screenshot of Python Libraries Used in AI-ML

```python
import numpy as np
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, LSTM, Flatten, Dense, Dropout, BatchNormalization, Bidirectional
from tensorflow.keras.utils import Sequence, to_categorical
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
import logging
```

```python
import tkinter as tk
from tkinter import messagebox
import numpy as np
import datetime
import threading
import subprocess
import csv
import tensorflow as tf
from tensorflow.keras.models import load_model
import geocoder
import subprocess
import os
```

```python
from packaging.version import Version as StrictVersion
from PyQt5 import Qt
from gnuradio import qtgui
from gnuradio import blocks
from gnuradio import digital
from gnuradio import gr
from gnuradio.filter import firdes
from gnuradio.fft import window
import sys
import signal
from PyQt5 import Qt
from argparse import ArgumentParser
from gnuradio.eng_arg import eng_float, intx
from gnuradio import eng_notation
from gnuradio import uhd
import time
from gnuradio.qtgui import Range, RangeWidget
from PyQt5 import QtCore
import main_epy_block_0 as epy_block_0  # embedded python block
import sip
```

Figure 19: Python Libraries. 1st is for training, 2nd is for the whole system functioning, 3rd is for the GNU Radio script itself