VCU College of Engineering

MULT 25-607 ML for RF Spectrum Sensing

Preliminary Design Report

Prepared for

Chandler Barfield, Riley Stuart, Robie John

Vectrus V2X

By

Baaba Jeffrey, Daniel Hartman, Shane Simes, Kush Patel

Under the supervision of

Yanxiao Zhao & Tamer Nadeem

12/09/2024

Executive Summary

This project aims to further the previous team's efforts in using machine learning to identify features of radio frequency (RF) signals to be able to distinguish between WiFi and Bluetooth signals. The need for a project such as this has become increasingly apparent with the rapid propagation of devices cluttering the RF spectrum. This necessitates a change in the way communication systems operate for organizations such as the Navy. Since the use of these systems is currently a human driven task, this project aims to utilize machine learning to speed up the information processing, and potentially even the decision making process.

We will be using a USRP B210 software defined radio (SDR) to capture the signals and transform them into IQ (in-phase and quadrature) data for easier sampling. This data will then be transformed using an FFT (fast fourier transform) in order to provide the frequency information that the machine learning algorithm will use to differentiate between the RF signals. The algorithm will be a trinary classifier outputting whether the signal is WiFi, Bluetooth, or neither. The fall semester and winter break will be the first phase of the project where we are setting up and training the algorithm to recognize these signals. The spring semester will be the second phase where we will be looking at how far we can take this project. This will include things like testing with real world data as well as looking at options such as unsupervised training of the algorithm.

The project is sponsored by V2X, a military contractor who is helping to provide resources as well as background information for this project. At the end of the project we will have a combined system that utilized our RF dataset and the machine learning model, along with any documentation, to deliver to V2X.

Table of Contents

Section A. Problem Statement

Figure 1. USS Carl Vinson [1]

Military organizations, particularly the U.S. Navy, rely extensively on sophisticated communication systems that employ radio frequency (RF) signals for a wide array of critical operations. These RF signals are essential for the proper functioning of various systems, including radar, communication networks, and electronic warfare technologies. However, the electromagnetic spectrum's increasing congestion has presented significant challenges in the efficient identification, monitoring, and classification of RF signals. This challenge is becoming increasingly pronounced as both friendly and adversarial RF signals grow in complexity and number, especially in modern warfare scenarios where speed, precision, and reliability are paramount.

Naval vessels, which operate in dense RF environments, are particularly affected by this issue. They often encounter a cluttered electromagnetic spectrum, with signals from a wide variety of devices—both civilian and military—overlapping and competing for bandwidth. The traditional manual identification and classification of RF signals, performed by human operators, has become a bottleneck in this complex environment. This process is not only time-consuming but also highly susceptible to human error, particularly in mission-critical situations where stress and time constraints are significant factors.

An illustrative example of the risks involved is the identification of adversarial jamming or spoofing attempts designed to disrupt radar or communication systems. If an adversary successfully jams or manipulates an RF signal without immediate detection and counteraction, critical systems aboard a naval vessel could be temporarily disabled. In extreme scenarios, such disruptions could lead to catastrophic outcomes, including the potential loss of life or even the sinking of a vessel. Given the rapid and dynamic nature of electromagnetic warfare, it is clear that real-time response is essential. However, the traditional human-in-the-loop approach is increasingly inadequate to meet the speed required for effective counteraction.

The challenges posed by congested RF spectrums are not unique to the U.S. Navy. Many other military organizations, as well as civilian sectors such as aviation and emergency services, are grappling with similar issues. However, in high-stakes environments like military operations, the need for rapid and accurate RF signal classification is especially pressing. The growing sophistication of electronic warfare (EW) systems among adversaries has only exacerbated this issue. Today, many nations possess advanced EW capabilities that can effectively disrupt

RF-dependent operations, underscoring the global importance of developing robust RF signal classification and countermeasure systems. As a result, military forces around the world are investing heavily in solutions designed to enhance their RF signal processing and electronic warfare resilience.

In response to these challenges, this research project, sponsored by V2X—a Vectrus company—focuses on the intersection of electrical and computer engineering with computer science. The project's primary objective is to address the critical challenges of RF signal capture, processing, and classification by leveraging advanced machine learning techniques. V2X, a leading provider of mission-critical solutions to defense and government clients, is supporting this initiative to enhance real-time RF signal identification and classification for U.S. Navy assets. The project aims to significantly improve operational efficiency and reduce the reliance on human operators, ultimately increasing both the accuracy and speed of RF signal classification.

As part of this design, we plan to utilize an RTL-SDR V3 for signal capture, specifically targeting the 2.4 GHz band commonly used for Wi-Fi and Bluetooth communications. This RF data will be processed in real-time, starting with a Fast Fourier Transform (FFT) to break down the spectrum and extract key features. The data will then be further analyzed by focusing on the in-phase (I) and quadrature (Q) components of the signal. Similar approaches have been validated in previous research. For example, Wang, Chen, and Zhang (2021) successfully captured and processed I/Q data under real-world engineering conditions, including random noise, which demonstrated the feasibility of this approach. However, their implementation of a neural network resulted in a relatively low classification accuracy of 79%, highlighting the need for further refinement.

Recent advancements in end-to-end machine learning systems offer promising solutions. For instance, Alam et al. (2023) explored the use of deep learning techniques to detect and identify drones using RF signals. Their research successfully developed an end-to-end system that integrates RF signal processing with advanced machine learning algorithms, resulting in significant improvements in both detection speed and classification accuracy. Notably, their system achieved sub-millisecond detection times and a classification accuracy rate of 98% for unmanned aerial vehicles (UAVs), largely due to the high-quality data available from the CardRF dataset. This underscores the critical importance of access to large, high-quality datasets for training machine learning models and achieving superior performance in RF signal classification.

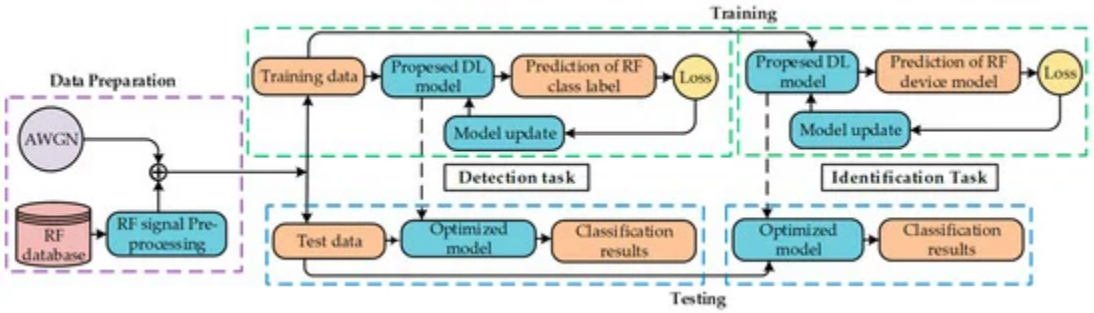


Figure 2. Architecture of signal identification in Alam et al. (2023) [3]

Building on these advancements, deep neural networks (DNNs) have emerged as powerful tools in RF signal identification, particularly in complex and crowded electromagnetic environments. Unlike traditional hand-engineered methods, which are limited by the need for domain-specific knowledge, DNNs can automatically learn features from raw RF data. This capability is crucial in environments where many transmitters share a channel or where high data rates are involved, such as naval operations. Research by Youssef et al. (2018) into DNNs with multi-stage training has shown especially promising results, achieving 100% classification accuracy for 12 different transmitters, demonstrating remarkable scalability for larger transmitter populations. By focusing on the intrinsic physical characteristics of RF signals, rather than just transmitted data, these networks offer a robust approach for rapidly identifying unknown or rogue transmitters. Incorporating DNNs into the current project can significantly enhance real-time RF signal classification, further improving both speed and accuracy while reducing reliance on human operators.

In choosing between power spectral density (PSD) and in-phase/quadrature (I/Q) neural training, research indicates that I/Q data can effectively capture intricate time and phase variations, enhancing classification in dynamic RF environments. Conversely, PSD is more suitable for frequency-based analysis, particularly for detecting jamming or interference. The decision to focus on one method will depend on the specific operational requirements of the RF environment (Elyousseph & Altamimi, 2021).

This project will enhance V2X's capabilities in navigating complex RF signal environments, laying the groundwork for future advancements in signal processing. By focusing on either power spectral density or in-phase/quadrature analysis as the training method, the project aims to address critical challenges in RF signal classification. The anticipated outcomes will significantly improve operational efficiency, demonstrating the potential impact of innovative solutions in real-world applications while providing valuable insights into the effectiveness of machine learning techniques in this domain.

Section B. Engineering Design Requirements

This section outlines the goals, objectives, specifications, and constraints that will guide the design and development of our RF spectrum sensing device. The requirements have been established through an understanding of client needs, existing technology, and thorough research. These requirements ensure the intended outcomes of client expectations and ensure the project remains aligned with previous work.

B.1 Project Goals (i.e. Client Needs)

The primary goal of this project is to improve situational awareness in radio frequency (RF) environments that are heavily contested by using machine learning to identify and categorize devices that are using a certain frequency band. This goal addresses the requirement to effectively monitor RF spectrum consumption and recognize various devices within a certain environment. The following list outlines the primary goals determined by the needs of the project to guarantee the successful implementation of this solution:

- To create a system that can identify active frequencies inside a designated RF band
- To categorize devices according to their frequency properties
- To create a machine learning model that can accurately distinguish between different kinds of devices
- To improve the efficiency of spectrum monitoring and make it more reliable and cost-efficient

B.2 Design Objectives

Design Objectives include:

- The design will detect active frequencies within the specified RF bands
- The design will be able to determine if the device is a WI-FI or Bluetooth device
- The design will process real-time data from the RF environment
- The design will use achievable software and hardware resources, including existing machine learning algorithms and RF data collection tools
- The design will be developed, tested, and validated to ensure proper effectiveness

B.3 Design Specifications and Constraints

- Design must use a USRP B210 SDR for processing signals into IQ data - (Cost constraint)
- The design will use a trinary classifier that distinguishes between Wifi, Bluetooth, or neither
- The design must have a User Interface showing which Network types are being used, ie Wifi and Bluetooth

B.4 Codes and Standards

- 802.11-2020/Cor 1-2022 - IEEE Standard for Information Technology--Telecommunications and Information Exchange between Systems
- 802.15.1-2002 - IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN

Section C. Scope of Work

The primary focus of this project is to develop a system that captures live RF spectrum data and utilizes machine learning algorithms to identify specific target signals, namely Bluetooth and Wi-Fi operating in the 2.4 GHz band. The project aims to create a reliable and efficient method for RF signal detection and classification, with the following key objectives:

1. RF Spectrum Capture: We will design and implement a system utilizing Software-Defined Radio (SDR) technology for capturing RF signals in the 2.4 GHz frequency range. The SDR will provide the flexibility to tune into various frequencies and bandwidths, allowing for real-time signal capture across multiple protocols. This capability will ensure that diverse and high-quality data is gathered, which is crucial for effective machine learning model training.

2. Signal Identification via Machine Learning: The project will develop and train machine learning models capable of accurately classifying captured Bluetooth and Wi-Fi signals. This process will include creating a data pipeline that encompasses feature extraction, model training, and evaluation. We aim to ensure the models perform reliably under various conditions, accounting for factors such as signal strength and interference.

3. System Integration and Automation: A key aspect of the project is to create an autonomous system that can detect and classify signals without human intervention. This approach will enhance the efficiency and reliability of the signal identification process, streamlining operations and reducing the potential for human error.

4. Verification and Validation: The system will undergo rigorous testing to validate its performance and accuracy. We will evaluate the machine learning models to ensure they meet the required specifications and can effectively classify target signals in diverse environments.

C.1 Deliverables

Project Deliverables

The project deliverables include:

1. RF Dataset: A comprehensive collection of captured RF spectrum data, specifically targeting Bluetooth and Wi-Fi signals in the 2.4 GHz range. This dataset will be gathered through live data collection, which can take place in a lab setting or remotely, utilizing Software-Defined Radio (SDR) technology for flexible and efficient data capture.

2. Machine Learning Model: A trained model capable of accurately identifying and classifying Bluetooth and Wi-Fi signals from the collected RF data. The coding and training processes for this model will be performed on local machines with a github repo, allowing team members to work effectively from various locations. This flexibility in development enables continuous iteration and improvement of the model.

3. Combined System: An integrated solution that merges the RF dataset and the machine learning model, facilitating real-time signal detection and classification. This system will leverage both the collected data and the trained model to provide accurate and timely results with a user interface.

4. Documentation: This includes user manuals, system architecture descriptions, and guidelines for future enhancements or modifications. Comprehensive documentation will ensure that all aspects of the project are clearly communicated and easily accessible for future reference.

5. Academic Deliverables: Additional outputs will include essential academic documents, such as:
    a. Team contract
    b. Project proposal
    c. Preliminary design report
    d. Fall poster and presentation
    e. Final design report
    f. Capstone EXPO poster and presentation

C.2 Milestones

This project has two teams that are working on their own respective aspects through the project, these being the RF/SDR team and the other being the machine learning and GUI team. At the beginning this results in the two parts of the project having separate milestones. The RF team's first major milestone will be getting to the point of capturing testing data, this means our SDR is programmed to capture signals and transform it to IQ data. The next major milestone is when we have created the process to capture and prep this data so it is ready for training the machine learning algorithm. This means we have developed a process to get clean, usable data that has been converted to the frequency domain and formatted properly.

For the machine learning team, the first major milestone will be mapping out the layers of the machine learning algorithm as well as what functions those layers will use. The next major milestone will be after the initial training of the algorithm. After this the major milestone will be when the algorithm is able to distinguish between the signals at a minimum of 90% accuracy.

These milestones up to this point are planned for the fall semester and winter break. For the spring semester we will be mapping out where we'd like to based on the progress up to that point. We would like to explore possibilities such as testing on real-world data as well as unsupervised learning.

Section D. Concept Generation

This section outlines two distinct design concepts to address the problem of real-time RF signal identification and classification, leveraging advanced machine learning techniques. Each concept is evaluated for its potential to meet the design objectives, considering its strengths, weaknesses, and associated risks. Additional sub-problems arising from each concept are also explored to identify opportunities for refinement and improvement.

1. PSD-Based Signal Classification Using LightGBM

This approach utilizes a Light Gradient Boosting Machine (LightGBM) model trained on features extracted from the Power Spectral Density (PSD) of captured RF data. The PSD provides a frequency-domain representation of the signals, and LightGBM leverages engineered features—such as spectral shape and bandwidth—to effectively distinguish between Wi-Fi and Bluetooth activity.

- Pros: LightGBM is highly efficient and well-suited for structured, tabular data derived from PSD features. It offers fast training and inference, making it suitable for real-time applications. The model also provides greater interpretability compared to deep neural networks, allowing for insight into which features are most important for classification.
- Cons: The performance of LightGBM is dependent on the quality and diversity of the engineered features. It may not capture highly complex or subtle patterns as effectively as

deep learning models without extensive feature engineering. Additionally, the model's accuracy can be affected by the presence of noise or overlapping signals in the PSD data.

Potential Risks: If the feature extraction process does not adequately capture the distinguishing characteristics of different signal types, classification accuracy may suffer. Furthermore, the model may require retraining or feature refinement to adapt to new signal environments or hardware changes. However, LightGBM's efficiency ensures that computational requirements remain manageable for real-time processing.

2. I/Q-Based Signal Classification Using Convolutional Neural Networks

This concept utilizes in-phase (I) and quadrature (Q) signal components to train a convolutional neural network. I/Q data captures intricate time and phase variations, which can enhance classification accuracy in dynamic RF environments.

- Pros: Captures detailed time-domain and phase information. Well-suited for analyzing frequency-hopping signals like Bluetooth.
- Cons: Requires significant preprocessing to manage the SDR B210's bandwidth constraints. Neural networks are computationally intensive and demand large datasets for effective training.

Potential Risks: Dynamic signal characteristics like frequency hopping could result in signal loss during capture. Noise levels may impact the quality of features extracted from I/Q data.

3. Random Forest Classifier with Feature Engineering

This concept involves leveraging a Random Forest (RF) classifier trained on manually engineered features such as PSD metrics, signal energy, or statistical attributes of the I/Q data. RFs provide a robust and interpretable alternative to neural networks, particularly in environments with limited bandwidth and noisy data.

- Pros: Computationally efficient and robust to noise. Provides interpretability and insight into feature importance. Performs well with small to medium-sized datasets.
- Cons: Requires manual feature engineering, which can be labor-intensive and domain-specific. May not fully capture the complexity of RF signals compared to neural networks.

Potential Risks: Suboptimal performance in handling dynamic features like frequency hopping without custom features. Limited scalability for extremely high-dimensional data.

Sub-Problem: SDR B210 Limitations

A key sub-problem across all concepts is the inherent limitation of the SDR B210, which can only view up to 50 MHz of bandwidth at a time. As the resolution increases, noise

levels also rise, potentially degrading signal quality. Additionally, frequency-hopping signals like Bluetooth may change channels during capture, leading to truncated or incomplete data. This issue necessitates innovative preprocessing or adaptive methods to ensure signal integrity across all design concepts.

Section E. Concept Evaluation and Selection

In this section, we evaluate each design concept based on a set of predefined criteria to determine the most suitable solution for the RF signal classification problem. A systematic decision-making process, using a Decision Matrix, helps eliminate biases and ensures a rational, structured approach to concept selection.

The following criteria were chosen to evaluate the design concepts:

- **Computational Efficiency**: This criterion measures the efficiency of each concept in terms of computational resources, including processing power, memory usage, and bandwidth requirements. Concepts that require fewer computational resources are considered more efficient and suitable for real-time applications.
- **Scalability**: Scalability refers to the ability of the design concept to handle larger datasets or increased complexity without significant degradation in performance. A scalable solution can efficiently adapt to growing data or more complex RF environments, which is critical for dynamic signal classification.
- **Development Complexity**: Development complexity evaluates the difficulty of designing, implementing, and maintaining each concept. Concepts with lower complexity are easier to develop, integrate, and maintain, which is essential for long-term success and timely project completion.

Decision Matrix

The design concepts were compared and scored using a weighted decision matrix. Each criterion was assigned a weight based on its importance to the project, and each concept was rated on a scale of 1 to 100 for each criterion. The weighted score was then calculated by multiplying the score for each criterion by its respective weight.

Criteria	Weight	PSD with CNN	I/Q with CNN	LightGBM
Computational Efficiency	50	75	50	80
Scalability	30	50	70	70
Development Complexity	20	75	55	50
Weighted Score	100	67.5	61.0	71.0

Evaluation and Selection

- PSD with Neural Network: This concept scored 67.5, offering good computational efficiency and moderate scalability. While neural networks can learn complex patterns from PSD data, they typically require more computational resources for both training and inference. The development complexity is also higher, as neural networks often need extensive tuning and larger datasets to achieve optimal performance.
- I/Q with Neural Network: Scoring 61.0, this approach is less favorable due to its higher computational demand and development complexity. While it offers good scalability, the large data volume from I/Q samples and the intensive processing requirements make it less suitable for real-time applications, especially in resource-constrained environments.
- LightGBM: Scoring 68.0, This concept achieved the highest weighted score of 71.0, primarily due to its strong computational efficiency and scalability. LightGBM is well-suited for real-time RF signal classification, as it can process engineered features from PSD data quickly and with minimal computational overhead. Its development complexity is moderate, as it requires careful feature engineering but is less demanding to implement and tune compared to deep neural networks. These characteristics make LightGBM an excellent fit for environments with limited processing resources and for applications requiring rapid, reliable classification.

Conclusion

Based on the decision matrix evaluation, LightGBM is selected as the most viable solution for this project. Its high computational efficiency, strong scalability, and manageable development complexity make it the best fit for our objectives and constraints. The next step is to proceed with the detailed design and implementation of the LightGBM-based classification system, focusing on optimizing feature extraction and model performance for real-time RF signal classification.

Section F. Design Methodology

In this section, we provide an overview of the design's computational methods, experimental methods, and the validation procedure used to determine whether a successful design has been achieved. The computational methods discussed include data processing through an FFT, data classification using a LightGBM model, and signal detection via our event detector. Our experimental method focuses on data collection and model refinement, while the validation section outlines how we will assess the effectiveness of the final design.

F.1 Computational Methods

There are two main computational methods utilized in this project: the FFT and the LightGBM model, along with an event detector that may increase in complexity as the project progresses. The FFT is used to process the input IQ data into Power Spectral Density (PSD) data, transforming the signal from the time domain to the frequency domain. Specifically, we collect 1024 samples at two different center frequencies (2.425 GHz and 2.475 GHz) with a 50 MHz bandwidth, then combine them to perform a 2048-sample FFT. We use NumPy's FFT function for this calculation, which is defined as:

$$A_k = \Sigma_{m=0}^{n-1} a_m exp\{-\ 2\pi i \frac{mk}{n}\}$$

where *n* is the 2048 sample length, *m* is the sample index, and *k* is the frequency bin. The result of this process is illustrated in Figure 3, which shows the IQ data as voltage over time, and Figure 4, which displays the PSD data after FFT, revealing frequency content from 2.4 to 2.5 GHz with a clear signal near 2.45 GHz.

After obtaining the PSD data, features are extracted to characterize the spectral properties of the signals. These features are then input to the LightGBM model, which performs classification to distinguish between Wi-Fi, Bluetooth, and background noise. The event detector identifies regions of interest within the PSD, ensuring that only relevant signal segments are analyzed. This combination of FFT-based preprocessing, feature engineering, and LightGBM classification forms the core of our computational methodology.

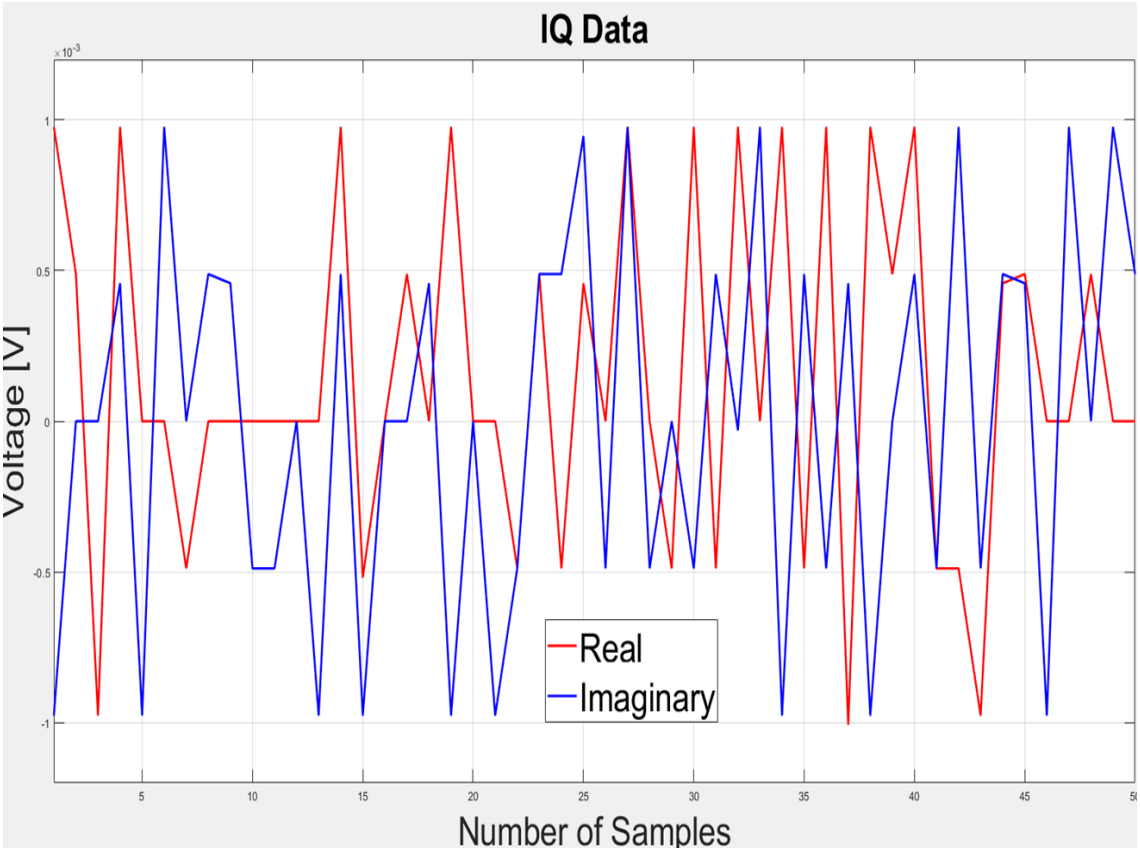


Figure 3. Sample IQ Data

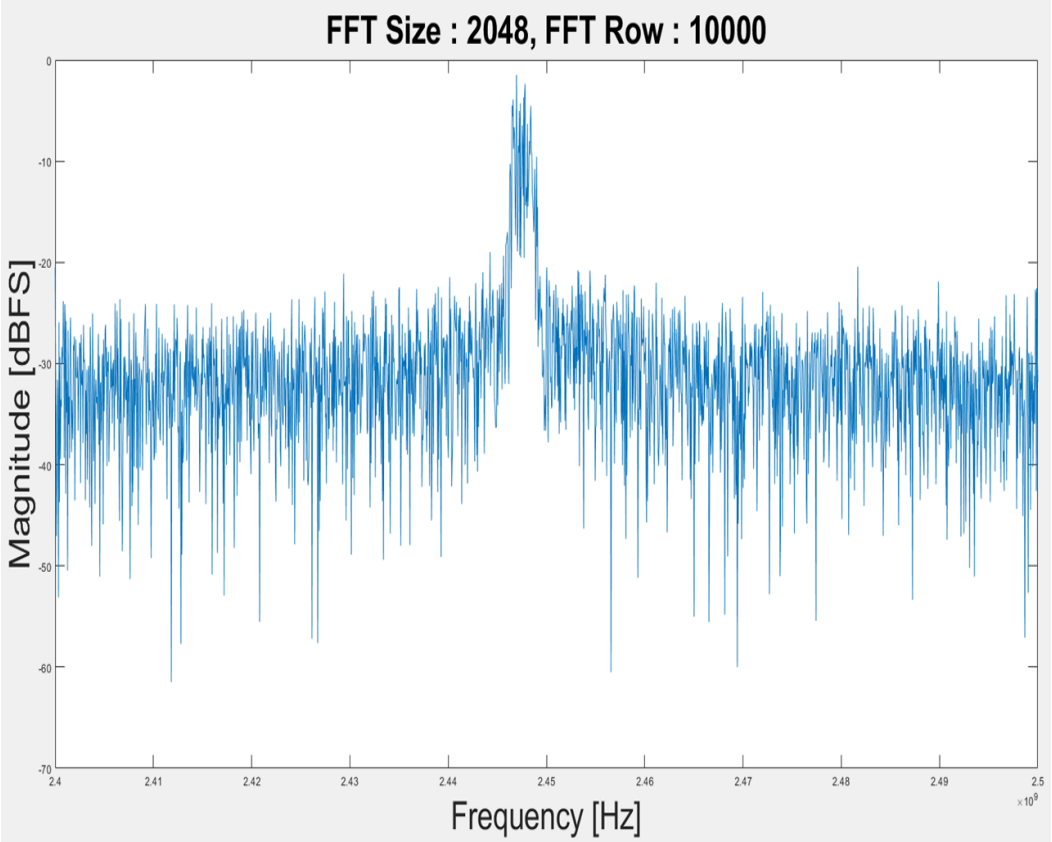


Figure 4. Sample PSD Data

The RF signal classification system is built around a LightGBM model, a state-of-the-art gradient boosting framework that excels at handling structured, tabular data. Instead of relying on deep learning architectures like convolutional neural networks, this approach leverages domain-specific feature engineering to extract meaningful characteristics from the power spectral density (PSD) data of radio frequency signals. The classifier is designed to distinguish between WiFi and Bluetooth transmissions, which differ primarily in their spectral width and energy

distribution.

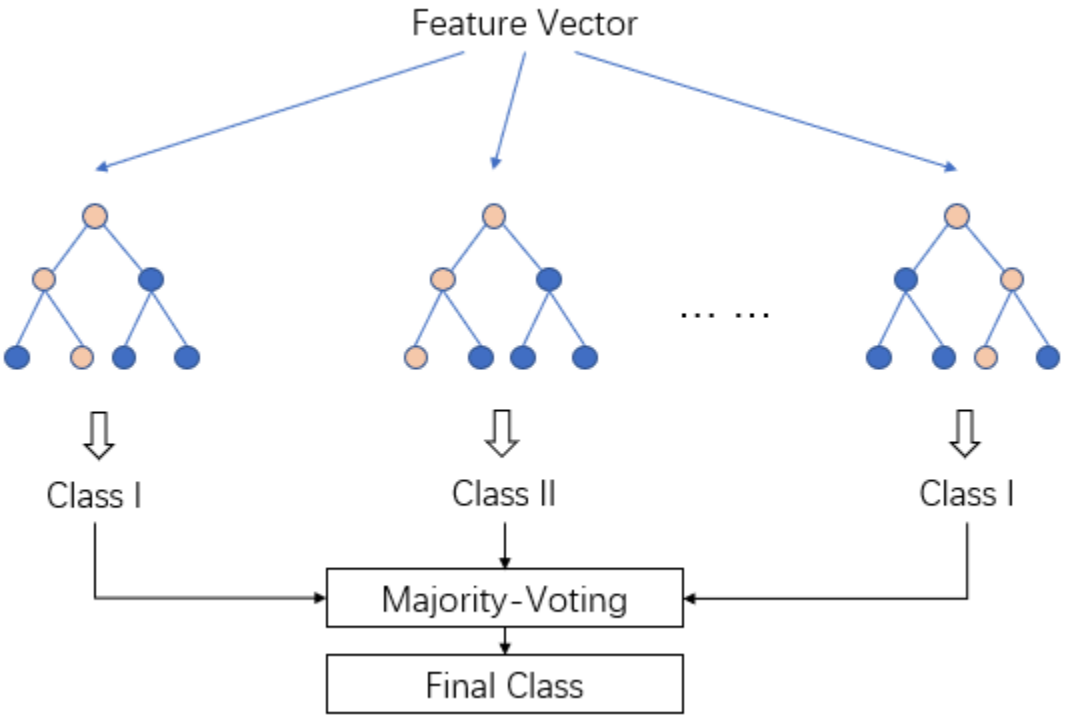


Figure 5. LightGBM Architecture

The classification pipeline begins with a preprocessing step that uses an event detector to identify regions in the PSD where a signal is present. This ensures that only the relevant portions of the spectrum are analyzed, while the rest is replaced with the estimated noise floor. By focusing on the active signal regions, the system improves its robustness to background noise and interference, which is especially important when operating in real-world environments.

Feature extraction is a critical component of the system. The classifier supports multiple feature extraction strategies, including spectral shape features, bandwidth features, and a combined approach. Spectral shape features capture information such as the number and width of peaks, energy distribution, and correlation with idealized WiFi and Bluetooth templates. Bandwidth features analyze how signal energy is distributed across different segments of the spectrum. By combining these features, the model gains a comprehensive view of the signal's characteristics, enabling more accurate classification.

Once features are extracted, the LightGBM model predicts the probability that each signal is WiFi or Bluetooth. The final classification is determined by these probabilities, with an additional rule that considers the width of the detected event if the model's confidence is low. This hybrid approach leverages both the statistical power of machine learning and domain knowledge about RF signal properties, resulting in a robust and interpretable solution for wireless signal classification.

F.2 Experimental Methods

Our experimental process begins with the data collection. All training data was collected in the lab over a wire. This means our devices generating bluetooth and WiFi signals will be connected via coax directly into our SDR. This was in order to give us the cleanest data with the best signal to noise ratio in order for the model to be trained on the clearest, highest resolution data. This resulted in an accurate model. We collected over 100,000 samples of each type of signal, this includes looking across the 11 used WiFi channels. We made sure the Wi-Fi signal was spread over all channels as this ensured that model was not looking at a frequency dependence to determine if a signal was Wi-Fi or Bluetooth.

Once we had the data collected we began training the model with the PSD data. As we fed in data, the model iterated based on whether it is successfully identifying a signal or not. These iterations are known as epochs. We measured two main outputs; loss (for training and testing data) which measures how far off the model's predictions are from the true value, and accuracy or the percentage of correct predictions. A sample image of how the values change through epochs is given below:

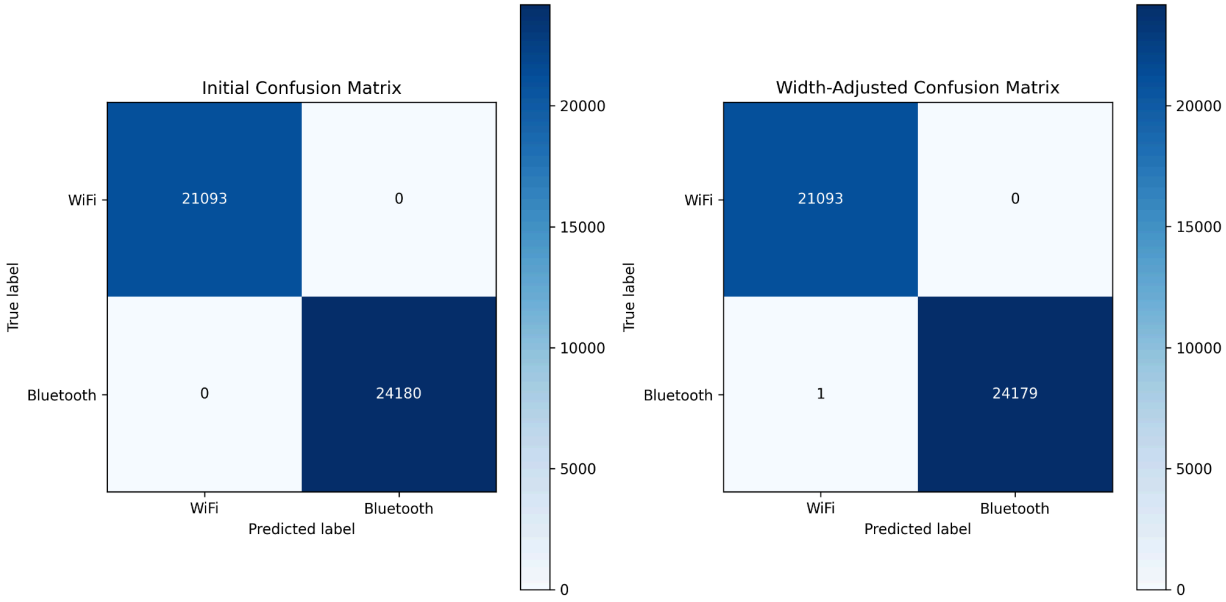


Figure 6. LightGBM Evaluation

If we do not meet our desired accuracy targets, we will need to evaluate whether our LightGBM model architecture, feature engineering, or data collection process requires modification. After making any necessary changes, we will repeat the data collection, training, and testing cycles as needed until we achieve a functional and accurate model.

F.3 Validation Procedure

TThe final implementation of the design will be considered successful in meeting our sponsor V2X's needs based on two main metrics: classification accuracy and real-time operation. Our trained LightGBM model will be tested to determine how accurately it can identify signals in the testing data. To be considered successful, we are aiming for a minimum of 95% accuracy in correctly identifying a signal as Wi-Fi, Bluetooth, or neither. After training, the model will be evaluated on collected testing data to ensure it meets this threshold. If it does not, we will revisit and improve either the data collection process or the LightGBM model's feature set and parameters.

Once we have achieved an accurate model, we will proceed to implement and validate real-time or near real-time detection. Our goal is for the system output to update at least 30 times per second (30 Hz), though higher rates are desirable. This requires efficient capture and processing of IQ data, rapid FFT computation, effective event detection, and fast classification with the LightGBM model. Once the system can reliably process and classify signals at or above this rate, we will have met our real-time performance objective.

Section G. Results and Design Details

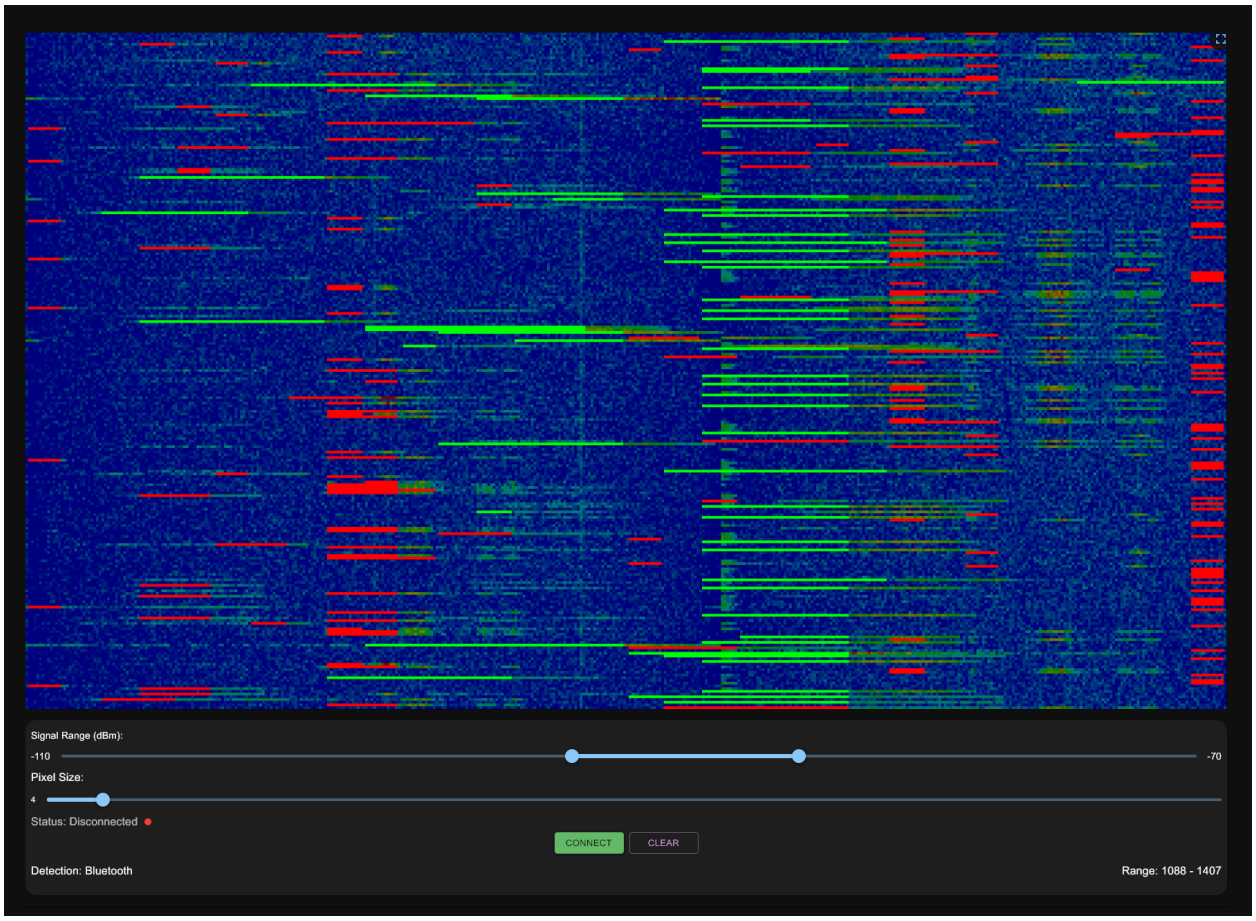
This section highlights the key result, analysis, and design details that reflect our current implementation of the RF spectrum sensing device. The project goals, objectives, and constraints were met through systematic design, modeling, and testing, ensuring the solution is both efficient and reliable for live measures.

G.1 Modeling Results

Training a model is a crucial step in any machine learning system, especially for tasks such as signal classification. In our approach, we utilized a Light Gradient Boosting Machine (LightGBM) model, which is well-suited for structured data and offers high accuracy and efficiency. Training the LightGBM model involves providing it with a large set of labeled RF signal data, allowing it to learn the distinguishing features of Wi-Fi, Bluetooth, and background noise. Through this process, the model identifies patterns in the extracted features—such as spectral shape and bandwidth—that are characteristic of each signal type. Detailed code for our SDR data acquisition and machine learning pipeline is included in Appendix 3 and 4.

G.2 Experimental Results

As the PSD data is processed and features are extracted, the LightGBM model classifies each detected signal event. The results are visualized in a real-time waterfall plot, as shown in Figure 8. A waterfall plot displays frequency, time, and signal power through color intensity, providing a comprehensive view of how signals vary across different frequencies over time. For this design, the visualization allows us to observe clear patterns associated with Wi-Fi and Bluetooth signals. The model accurately recognizes Wi-Fi signals with 20 and 40 MHz bandwidths and Bluetooth signals with 1 and 2 MHz bandwidths.



(Green Wifi) (Red Bluetooth)

Figure 8. RF Data Waterfall Plot

G.4. Final Design Details/Specifications

Our constraints for this project were that the design must use a USRP B210 SDR for processing signals into IQ data, the design must use a trinary classifier that distinguishes between Wifi, Bluetooth, or neither and the design must have a User Interface showing which Network types are being used, i.e Wifi and Bluetooth.

Our design effectively incorporates all key elements for successful signal recognition. The system utilizes a USRP B210 for RF data capture, operating within the 2.4 GHz - 2.5 GHz frequency range at a 50 MHz sampling rate. To generate the PSD, the design uses a 2048-point FFT. The classifier is a binary system comprising an input layer, a hidden layer, and an output layer. A LightGBM processes the PSD features to accurately classify signals. The system includes a power threshold-based event detector for initial signal detection and identification. As shown in the waterfall plot above, our model confirms the UI showing which network types are wifi or Bluetooth.

Section H. Societal Impacts of Design

H.1 Public Health, Safety, and Welfare

The project incorporates several critical safety features designed specifically for naval applications. Spectrum interference mitigation ensures reliable communication by identifying and managing nearby signals, which is essential for naval ships operating in complex RF environments. Anomaly detection helps identify unauthorized or malicious spectrum use, protecting sensitive naval operations from potential breaches or harmful interference. Adaptive learning capabilities allow the system to adjust to changing RF conditions in maritime environments, ensuring consistent performance and safety. Fail-safe mechanisms are implemented to prevent unsafe conditions by reverting to secure defaults if the system encounters anomalies. Compliance with regulatory standards from organizations like the FCC and ITU ensures the system operates within legal and safety guidelines. These measures enhance public safety, protect naval communication infrastructure, and ensure robust, interference-free spectrum monitoring, reducing risks through thorough testing.

H.2 Societal Impacts

The project significantly enhances naval communication capabilities by enabling ships to detect and analyze nearby signals efficiently. This capability supports naval security, ensures reliable operations in maritime environments, and contributes to global maritime safety. By optimizing spectrum usage, the system promotes secure and efficient communication within and between naval fleets. These advancements strengthen the safety of naval personnel and contribute to broader societal benefits by protecting trade routes and enhancing maritime operations worldwide.

H.3 Political/Regulatory Impacts

By aligning with national and international spectrum policies, the project ensures compliance with regulatory standards critical for naval operations. Its deployment demonstrates adherence to legal frameworks and supports secure maritime communications. The project's success could influence policy developments in spectrum management for defense applications and foster international collaboration on maritime security. These advancements strengthen political and regulatory frameworks for global naval operations, ensuring harmonized practices.

H.4. Economic Impacts

Efficient spectrum management reduces operational costs for naval forces by optimizing the use of available RF resources, minimizing the need for additional infrastructure. The technology fosters innovation in naval communication systems, supporting industries involved in defense technology development. By improving the effectiveness of naval operations, the project also enhances economic stability by securing trade routes and maritime resources. The creation and deployment of such technologies contribute to job opportunities in defense, technology, and regulatory sectors.

H.5 Environmental Impacts

Optimized spectrum usage in naval operations reduces energy consumption by ensuring efficient communication and signal management, even in challenging maritime environments. The project's design extends the lifespan of existing naval communication equipment, reducing electronic waste and supporting environmental sustainability. These efficiencies align with global efforts to reduce the environmental footprint of military operations while maintaining operational effectiveness

H.6 Global Impacts

The project enhances global naval communication systems by enabling effective spectrum sharing and utilization in international waters. It supports maritime security and disaster response, fostering international collaboration on maritime safety initiatives. By improving communication capabilities, the project strengthens global maritime operations and contributes to the protection of international trade and resources. Its adherence to international standards positions it as a leader in naval spectrum management, influencing technological advancements worldwide.

H.7. Ethical Considerations

Ethically, the project ensures that spectrum monitoring does not compromise the privacy of nearby signals or vessels. It addresses potential biases in machine learning models to ensure accurate and equitable signal detection. Transparency in how the system operates and makes decisions is prioritized to maintain trust among stakeholders. Accountability mechanisms are incorporated to address any operational failures, ensuring reliability and ethical integrity. These considerations uphold the highest standards of fairness and respect for international maritime communication norms.

Section I. Cost Analysis

Part Name	Part Number	Vendor	Unit	Qty.	Total	Delivery	Date

		Name	Cost		Cost	Time	Received
SDR	ETTUS USRP B210	Provided by VCU	$2322	1	$2322	Unknown - previous year	Unknown - previous year
Bluetooth Receiver	Maxuni BT 5.3	Unknown - previous year	$12.79	1	$12.79	Unknown - previous year	Unknown - previous year
NETGEAR Router	Unknown	Unknown - previous year	$28.00	1	$28.00	Unknown - previous year	Unknown - previous year
GL iNet Router	GL-AR300 M16-EXT	Amazon	$25.52	1	$25.52	10 days	12/2/24
SDR	RTL SDR Blog V3	Unknown - previous year	$43.95	1	$43.95	Unknown - previous year	Unknown - previous year

Section J. Conclusions and Recommendations

The primary objective of this project is to develop a system that uses machine learning to accurately classify W-Fi and Bluetooth signals within the RF spectrum. This initiative builds upon the work of last year's design group, which encountered challenges related to model overfitting, resulting in poor signal classification performance. This year's approach focuses on mitigating overfitting by implementing selective model training techniques to ensure the model generalizes effectively and consistently meets the classification requirements.

Our team operates collaboratively while using our specialized expertise by diving into two focused groups. The RF team handles data collection working closely with the SDR, conversion of raw IQ data to PSD data, and the transfer of this data to the ML model. The ML team is responsible for developing the classification model and integrating the results into the frontend interface. Our design process was shaped through weekly meetings with our sponsors and advisors, who provided guidance and recommendations at each stage of the project. This structured approach ensures we remain on track and make informed decisions throughout the development process.

This design could be further enhanced by developing a mobile application capable of detecting and classifying RF signals directly from a mobile device. Such an app would provide a portable, user-friendly solution for real-time RF spectrum analysis, enabling users to quickly

identify and classify surrounding signals on the go. This advancement would not only streamline the signal classification process but also broaden accessibility to the technology, making it practical for use in various environments. Integrating features like real-time visualization, data logging, and alerts could further improve the usability of the tool. In the case that there would be a continuation of this project, this could be their focus or other ways to optimize the model.

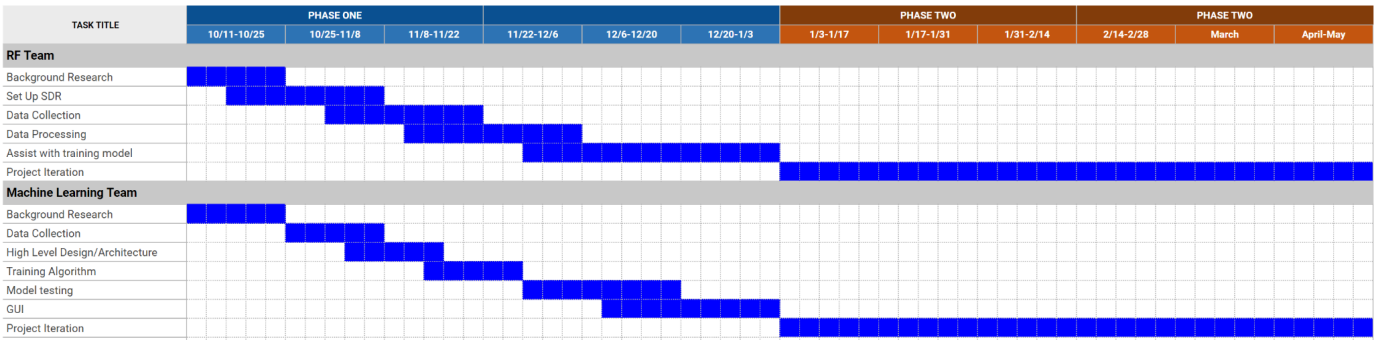
We achieved our goal of real-time signal capture, classification, and visualization. Our live-waterfall graph was able to operate above our target of 30 Hz, and the classification accuracy remained high even in environments with significant signal density. However, one limitation of our current system is that, during inference, we are only able to access the first detected signal event as we scan from left to right across the spectrum. This means that in cases where multiple signals are present simultaneously, only the first detected event is classified, potentially missing additional overlapping or adjacent signals.

To further improve our classification performance, the next steps would include acquiring data from multiple Wi-Fi and Bluetooth sources to introduce greater variation into the training set. One of the main sources of error observed was the presence of Bluetooth signals with a larger bandwidth than those represented in our training data. By incorporating a wider variety of emitters, we can better capture the diversity of real-world signals and reduce misclassification. Additionally, collecting training data that includes both Bluetooth and Wi-Fi signals present simultaneously would be necessary to maximize accuracy in complex environments.

This enhancement would also require transitioning to a trinary classification system, capable of identifying when both Bluetooth and Wi-Fi are present in the same spectrum window. Finally, including scenarios with multiple Bluetooth or Wi-Fi signals in the training data would further improve the model's robustness in noisy or crowded environments. Achieving this would require access to multiple emitters during data collection, but would significantly strengthen the system's real-world performance.

Our SDR code is illustrated in appendix 3 and our ML model code is illustrated in appendix 4.

Appendix 1: Project Timeline



Appendix 2: Team Contract (i.e. Team Organization)

Step 1: Get to Know One Another. Gather Basic Information.

Task: This initial time together is important to form a strong team dynamic and get to know each other more as people outside of class time. Consider ways to develop positive working relationships with others, while remaining open and personal. Learn each other's strengths and discuss good/bad team experiences. This is also a good opportunity to start to better understand each other's communication and working styles.

Team Member Name	***Strengths each member bring to the group***	***Other Info***	***Contact Info***
Shane Simes	Adaptability, Strong organization skills	Strong skills in many different coding languages, good at document/report drafting	simess@vcu.edu
Daniel Hartman	Flexibility, communication skills	Knowledgeable in HFFS, proficient at soldering, knowledgeable in multiple programming languages	hartmand2@vcu.edu
Kush Patel	Adaptable, quick to learn.	Technical knowledge in embedded systems, strong skills in various languages and tools.	Patelku2@vcu.edu
Baaba Jeffrey	Organization, adaptability, collaborative skills	Skillful in multiple programming languages	Jeffreybt@vcu.edu

Other Stakeholders	***Notes***	***Contact Info***
Yanxiao Zhao		yzhao7@vcu.edu

Tamer Nadeem	Was on similar project last year	tnadeem@vcu.edu
Riley Stuart John Robie		Riley.Stuart@gov2x.com John.Robie@gov2x.com

Step 2: Team Culture. Clarify the Group's Purpose and Culture Goals.

Task: Discuss how each team member wants to be treated to encourage them to make valuable contributions to the group and how each team member would like to feel recognized for their efforts. Discuss how the team will foster an environment where each team member feels they are accountable for their actions and the way they contribute to the project. These are your Culture Goals (left column). How do the students demonstrate these culture goals? These are your Actions (middle column). Finally, how do students deviate from the team's culture goals? What are ways that other team members can notice when that culture goal is no longer being honored in team dynamics? These are your Warning Signs (right column).

Resources: More information and an example Team Culture can be found in the Biodesign Student Guide "Intentional Teamwork" page (webpage | PDF)

Culture Goals	***Actions***	***Warning Signs***
Respect and Support	- Encourage open communication and active listening during meetings. - Provide constructive feedback and recognize each other's strengths.	- Team members interrupt each other frequently. - Criticism is not constructive or lacks respect.
Accountability	- Clearly define each member's responsibilities and deadlines. - Regularly review progress and address any issues promptly.	- Missed deadlines are frequent without explanation. - Lack of follow-through on assigned tasks.

Acknowledging Contributions	- Recognize and praise individual achievements during team meetings. - Provide positive feedback and encourage peers.	- Team member's efforts are consistently overlooked. - No acknowledgment of significant contributions during team discussions.
Constructive Feedback	- Offer feedback that is specific, actionable, and supportive. - Encourage a culture where feedback is welcomed and acted upon.	- Feedback is vague, overly critical, or not delivered in a constructive manner. - Team members become defensive or disengaged during feedback sessions.
Collaborative Problem-Solving	- Engage in open discussions to address issues collaboratively. - Foster an environment where all opinions are considered and valued.	- Problems are ignored or handled in isolation. - Team members shut down or dismiss others' suggestions without discussion.

Step 3: Time Commitments, Meeting Structure, and Communication

Task: Discuss the anticipated time commitments for the group project. Consider the following questions (don't answer these questions in the box below):

- What are reasonable time commitments for everyone to invest in this project?
- What other activities and commitments do group members have in their lives?
- How will we communicate with each other?
- When will we meet as a team? Where will we meet? How Often?
- Who will run the meetings? Will there be an assigned team leader or scribe? Does that position rotate or will the same person take on that role for the duration of the project?

Required: How often you will meet with your faculty advisor advisor, where you will meet, and how the meetings will be conducted. Who arranges these meetings?
See examples below.

Meeting Participants	*Frequency* *Dates and Times / Locations*	*Meeting Goals* *Responsible Party*
Students Only	Every Tuesday at 7 pm at the library in person	Actively work on projects. Bring up questions about the project.
Students + Faculty advisor	May join weekly 7 pm meetings through Zoom	Update faculty advisor and get answers to our questions (Any of the team members will scribe; Baaba will create meeting agenda and lead meeting)
Project Sponsor	Meeting Monday 9/9 in person then meeting online after that through zoom	Update project sponsor and make sure we are on the right track (Shane will create meeting agenda and lead meeting; Kush/Daniel will present prototype so far)

Step 4: Determine Individual Roles and Responsibilities

Task: As part of the Capstone Team experience, each member will take on a leadership role, ***in addition to*** contributing to the overall weekly action items for the project. Some common leadership roles for Capstone projects are listed below. Other roles may be assigned with approval of your faculty advisor as deemed fit for the project. For the entirety of the project, you should communicate progress to your advisor specifically with regard to your role.

- **Before meeting with your team**, take some time to ask yourself: what is my "natural" role in this group (strengths)? How can I use this experience to help me grow and develop more?
- **As a group,** discuss the various tasks needed for the project and role preferences. Then assign roles in the table on the next page. Try to create a team dynamic that is fair and equitable, while promoting the strengths of each member.

Communication Leaders

Suggested: Assign a team member to be the primary contact for the client/sponsor. This person will schedule meetings, send updates, and ensure deliverables are met.

Suggested: Assign a team member to be the primary contact for faculty advisor. This person will schedule meetings, send updates, and ensure deliverables are met.

Common Leadership Roles for Capstone

1. **Project Manager:** Manages all tasks; develops overall schedule for project; writes agendas and runs meetings; reviews and monitors individual action items; creates an environment where team members are respected, take risks and feel safe expressing their ideas.
   Required: On Edusourced, under the Team tab, make sure that this student is assigned the Project Manager role. This is required so that Capstone program staff can easily identify a single contact person, especially for items like Purchasing and Receiving project supplies.
2. **Logistics Manager:** coordinates all internal and external interactions; leads in establishing contact within and outside of organization, following up on communication of commitments, obtaining information for the team; documents meeting minutes; manages facility and resource usage.
3. **Financial Manager:** researches/benchmarks technical purchases and acquisitions; conducts pricing analysis and budget justifications on proposed purchases; carries out team purchase requests; monitors team budget.
4. **Systems Engineer:** analyzes Client initial design specification and leads establishment of product specifications; monitors, coordinates and manages integration of subsystems in the prototype; develops and recommends system architecture and manages product interfaces.
5. **Test Engineer:** oversees experimental design, test plan, procedures and data analysis; acquires data acquisition equipment and any necessary software; establishes test protocols and schedules; oversees statistical analysis of results; leads presentation of experimental finding and resulting recommendations.
6. **Manufacturing Engineer:** coordinates all fabrication required to meet final prototype requirements; oversees that all engineering drawings meet the requirements of machine shop or vendor; reviews designs to ensure design for manufacturing; determines realistic timing for fabrication and quality; develops schedule for all manufacturing.

Team Member	***Role(s)***	***Responsibilities***
Shane Simes	Project Manager	- Primary contact for sponsor - Manage scheduling - Schedule monthly meeting with sponsor - Help with any task that's needed -
Kush Patel	Systems Engineering	- Overview of product design
Daniel Hartman	Test Engineering Finances	- Data acquisition - Design testing plan - Research for purchasing decisions - Track budget

Baaba Jeffrey	Logistics Manager	- Primary contact for advisor - Ensure all deadlines are met - Ensure everyone is on the same page and up to date with the project

Step 5: Agree to the above team contract

Team Member: Shane Simes *Signature: Shane Simes*

Team Member: Kush Patel *Signature: Kush Patel*

Team Member: Baaba Jeffrey *Signature: Baaba Jeffrey*

Team Member: Daniel Hartman *Signature: Daniel Hartman*

Appendix 3: SDR Code

```python
import uhd
import numpy as np
import sys
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm
sys.path.append("../../model")
from MLRF.datatools import h5kit, psdkit
import time


usrp = uhd.usrp.MultiUSRP()

num_samps = 1024  # number of samples received
sample_rate = 50e6  # Hz
Fs = 50e6
collects = 100000
bandwidth = 50e6


def receive_iq_data(center_freq):
    usrp.set_rx_freq(uhd.libpyuhd.types.tune_request(center_freq), 0)

    # Set up the stream and receive buffer
    st_args = uhd.usrp.StreamArgs("fc32", "sc16")
    st_args.channels = [0]
    metadata = uhd.types.RXMetadata()
    streamer = usrp.get_rx_stream(st_args)
    recv_buffer = np.zeros(1024, dtype=np.complex64)

    # Start Stream
    stream_cmd = uhd.types.StreamCMD(uhd.types.StreamMode.start_cont)
    stream_cmd.stream_now = True
    streamer.issue_stream_cmd(stream_cmd)

    # Receive Samples
    samples = np.zeros(num_samps, dtype=np.complex64)
    for i in range(num_samps // 1024):
        streamer.recv(recv_buffer, metadata)
        samples[i * 1024 : (i + 1) * 1024] = recv_buffer

    # Stop Stream
    stream_cmd = uhd.types.StreamCMD(uhd.types.StreamMode.stop_cont)
    streamer.issue_stream_cmd(stream_cmd)
```

```
    #print(center_freq)
    #current_time=time.time_ns()
    #print(current_time)
    # print(len(samples))
    # print(samples[0:10])
    return samples


def calculate_psd(x, center_freq):
    N = 1024
    # x = x * np.hamming(len(x))  # apply a Hamming window
    PSD = np.abs(np.fft.fft(x)) ** 2 / (N * Fs)
    PSD_log = 10.0 * np.log10(PSD)
    PSD_shifted = np.fft.fftshift(PSD_log)

    f = np.arange(Fs / -2.0, Fs/2.0, Fs / (N))  # start, stop, step.  centered around 0 Hz
    f += center_freq  # now add center frequency
    # plt.plot(f, PSD_shifted)
    # plt.show()
    return (PSD_shifted, f)



def event_detector(PSD, threshold):
    window_size = 16
    noise_average = (sum(PSD[0:2048])/len(PSD))
    i = 0


    # Loop through the array to consider
    # every window of size 3
    while i < len(PSD) - window_size + 1:

        # Store elements from i to i+window_size
        # in list to get the current window
        window = PSD[i : i + window_size]

        # Calculate the average of current window
        window_average = sum(window) / window_size
        if window_average > threshold and noise_average > -120:
            #print(noise_average)
            #print(window_average)
            return True
        else:
            i += window_size
            # print(i)
```

```python
from matplotlib.animation import FuncAnimation

try:
        data = h5kit('PSD_WiFi_CH1_and_Bluetooth_3-25_2.h5')
        z=1
        PSD= []
        rx_freq = 2.45e9  # Frequency in Hz (e.g., 2.45 GHz)
        rx_antenna = '0'  # The antenna name (can be '0', '1', etc.)
        usrp.set_rx_rate(sample_rate, 0)
        usrp.set_rx_agc(True, 0)
        usrp.set_rx_bandwidth(bandwidth, 0)
        #usrp.set_rx_lo_freq(rx_freq, rx_antenna)
        for i in tqdm(range(collects)):
            iq_data = receive_iq_data(center_freq=2.425e9)
            iq_data_2 = receive_iq_data(center_freq=2.475e9)
            event1, frequency1 = calculate_psd(iq_data, center_freq=2.425e9)
            event2, frequency2 = calculate_psd(iq_data_2, center_freq=2.475e9)
            event = np.concatenate((event1,event2))
            frequency = np.concatenate((frequency1,frequency2))

            #print(len(frequency))
            #print(frequency)

            if ((event_detector(event, -65))):
                #print(len(frequency))
                #print(len(x))
                #plt.figure(figsize=(20,5))
                #plt.grid(['both'])
                data.write(f"{z}", event.astype(np.float32))
                #print("Pass")
                z+=1
                #plt.plot(frequency, event)
                #plt.show()


except Exception as e:
    print(f"An error occurred: {e}")
```

Appendix 4: ML Model Code

Python

```
import numpy as np
from scipy import signal as scipy_signal
from .utils import event_detector


class RFClassifier:
     """RF Signal Classifier for WiFi vs Bluetooth using LightGBM"""

     def __init__(self, model, feature_method="combined", width_threshold=200):
     self.model = model
     self.feature_method = feature_method
     self.width_threshold = width_threshold  # ~10 MHz in bins
     self.wifi_width = 410  # ~20MHz in bins
     self.bt_width = 20  # ~1MHz in bins

     def preprocess_signal(self, X, noise_floor):
     """Apply event detection and noise floor replacement"""
     X_processed = X.copy()
     event_widths = []

     for i in range(X.shape[0]):
             events = event_detector(X[i])
             signal_mask = np.ones(X.shape[1]) * noise_floor
             total_width = 0
             for start, end in events:
             if start != 0 or end != 0:
                    signal_mask[start : end + 1] = X[i, start : end + 1]
                    total_width += end - start + 1
             event_widths.append(total_width)
             X_processed[i] = signal_mask
     return X_processed, event_widths

     def _extract_spectral_shape_features_direct(self, X):
     """Direct spectral shape feature extraction for model prediction"""
     features = []
     for i in range(X.shape[0]):
             signal = X[i]
             feature_vector = []
             peaks, properties = scipy_signal.find_peaks(
             signal, height=np.median(signal), distance=self.bt_width // 2
             )

             if len(peaks) == 0:
             peak_heights = [0] # Ensure defined even if no peaks
             peak_prominences = [0] # Ensure defined
             widths_3db = [0] # Ensure defined
             else:
```

```
        widths_3db = scipy_signal.peak_widths(
                signal, peaks, rel_height=0.5
        )[0]
        widths_6db = scipy_signal.peak_widths(
                signal, peaks, rel_height=0.75
        )[0]
        widths_10db = scipy_signal.peak_widths(
                signal, peaks, rel_height=0.9
        )[0]
        peak_heights = properties["peak_heights"]
        peak_prominences = scipy_signal.peak_prominences(
                signal, peaks
        )[0]

        if len(peaks) > 0:
        bt_like_peaks = np.sum(
                (widths_3db > 5) & (widths_3db < self.bt_width * 2)
        )
        wifi_like_peaks = np.sum(
                (widths_3db > self.bt_width * 2)
                & (widths_3db < self.wifi_width * 1.5)
        )
        wide_peaks = np.sum(widths_3db > self.wifi_width)
        width_height_ratios = widths_3db / peak_heights
        feature_vector.extend([
                len(peaks),
                np.mean(widths_3db) if len(widths_3db) > 0 else 0,
                np.std(widths_3db) if len(widths_3db) > 0 else 0,
                np.mean(widths_6db) if len(widths_6db) > 0 else 0,
                np.std(widths_6db) if len(widths_6db) > 0 else 0,
                np.mean(widths_10db) if len(widths_10db) > 0 else 0,
                np.std(widths_10db) if len(widths_10db) > 0 else 0,
                bt_like_peaks, wifi_like_peaks, wide_peaks,
                (
                        np.mean(width_height_ratios)
                        if len(width_height_ratios) > 0
                        else 0
                ),
                (
                        np.max(width_height_ratios)
                        if len(width_height_ratios) > 0
                        else 0
                ),
        ])
        width_bins = [
                0, self.bt_width / 2, self.bt_width, self.bt_width * 2,
```

```
                        self.wifi_width / 2, self.wifi_width, self.wifi_width *
1.5, 2048,
                ]
                width_hist, _ = np.histogram(widths_3db, bins=width_bins)
                feature_vector.extend(width_hist)
                peak_indices = np.argsort(peak_prominences)[-3:]
                top_widths = [
                        widths_3db[i] if i < len(widths_3db) else 0
                        for i in peak_indices
                ]
                feature_vector.extend(top_widths)
                else:
                feature_vector.extend([0] * 12)
                feature_vector.extend([0] * 7)
                feature_vector.extend([0] * 3)

                bt_energy, wifi_energy = [], []
                for start in range(
                0, len(signal) - self.wifi_width, self.wifi_width // 4
                ):
                if start + self.wifi_width <= len(signal):
                        wifi_window = signal[start : start + self.wifi_width]
                        wifi_energy.append(np.sum(10 ** (wifi_window / 10)))
                for start in range(
                0, len(signal) - self.bt_width, self.bt_width // 2
                ):
                if start + self.bt_width <= len(signal):
                        bt_window = signal[start : start + self.bt_width]
                        bt_energy.append(np.sum(10 ** (bt_window / 10)))
                feature_vector.extend([
                np.max(wifi_energy) if wifi_energy else 0,
                np.std(wifi_energy) if wifi_energy else 0,
                np.max(bt_energy) if bt_energy else 0,
                np.std(bt_energy) if bt_energy else 0,
                (
                        np.max(wifi_energy) / np.max(bt_energy)
                        if wifi_energy and bt_energy and np.max(bt_energy) > 0
                        else 0
                ),
                ])

                x_wifi = np.linspace(-10, 10, self.wifi_width)
                wifi_template = np.exp(-0.5 * x_wifi**2 / 9)
                x_bt = np.linspace(-10, 10, self.bt_width)
                bt_template = np.exp(-0.5 * x_bt**2)
                wifi_corr, bt_corr = [], []
                for start in range(
```

```
            0, len(signal) - self.wifi_width, self.wifi_width // 2
            ):
            if start + self.wifi_width <= len(signal):
                  window_signal = signal[start : start + self.wifi_width]
                  if np.std(window_signal) > 0:
                        corr = np.corrcoef(window_signal, wifi_template)[0,
1]
                        wifi_corr.append(corr)
            for start in range(
            0, len(signal) - self.bt_width, self.bt_width // 2
            ):
            if start + self.bt_width <= len(signal):
                  window_signal = signal[start : start + self.bt_width]
                  if np.std(window_signal) > 0:
                        corr = np.corrcoef(window_signal, bt_template)[0, 1]
                        bt_corr.append(corr)
            feature_vector.extend([
            np.max(wifi_corr) if wifi_corr else 0,
            np.mean(wifi_corr) if wifi_corr else 0,
            np.max(bt_corr) if bt_corr else 0,
            np.mean(bt_corr) if bt_corr else 0,
            ])
            feature_vector.extend([
            np.mean(signal), np.std(signal), np.max(signal), np.min(signal),
            np.median(signal), np.percentile(signal, 25),
np.percentile(signal, 75),
            ])
            features.append(feature_vector)
      return np.array(features)

      def _extract_bandwidth_features_direct(self, X):
      """Direct bandwidth feature extraction for model prediction"""
      features = []
      for i in range(X.shape[0]):
            signal = X[i]
            feature_vector = []
            num_segments = 8
            segment_size = len(signal) // num_segments
            for j in range(num_segments):
            start = j * segment_size
            end = (j + 1) * segment_size
            segment = signal[start:end]
            peaks, properties = scipy_signal.find_peaks(
                  segment,
                  height=np.median(segment),
                  distance=self.bt_width // 2,
            )
```

```
                if len(peaks) == 0:
                        feature_vector.extend([0, 0, 0, 0, 0])
                        continue
                widths = scipy_signal.peak_widths(
                        segment, peaks, rel_height=0.5
                )[0]
                bt_count = np.sum(
                        (widths > 5) & (widths < self.bt_width * 1.5)
                )
                wifi_count = np.sum(
                        (widths > self.bt_width * 2)
                        & (widths < self.wifi_width * 1.2)
                )
                avg_width = np.mean(widths) if len(widths) > 0 else 0
                heights = properties["peak_heights"]
                hw_ratio = (
                        np.mean(heights / widths)
                        if len(widths) > 0 and np.all(widths != 0) # Avoid division
by zero
                        else 0
                )
                peak_energy = 0
                for p, w in zip(peaks, widths):
                        left = max(0, int(p - w / 2))
                        right = min(len(segment), int(p + w / 2))
                        peak_region = segment[left:right]
                        peak_energy += np.sum(10 ** (peak_region / 10))
                total_energy = np.sum(10 ** (segment / 10))
                energy_ratio = (
                        peak_energy / total_energy if total_energy > 0 else 0
                )
                feature_vector.extend([
                        bt_count, wifi_count, avg_width, hw_ratio, energy_ratio,
                ])
                features.append(feature_vector)
        return np.array(features)

        def extract_features(self, X):
        """Extract features for model prediction using direct methods"""
        if self.feature_method == "spectral_shape":
                return self._extract_spectral_shape_features_direct(X)
        elif self.feature_method == "bandwidth":
                return self._extract_bandwidth_features_direct(X)
        elif self.feature_method == "combined":
                spectral = self._extract_spectral_shape_features_direct(X)
                bandwidth = self._extract_bandwidth_features_direct(X)
                return np.hstack((spectral, bandwidth))
```

```
else: # Default to combined
        spectral = self._extract_spectral_shape_features_direct(X)
        bandwidth = self._extract_bandwidth_features_direct(X)
        return np.hstack((spectral, bandwidth))

def predict(self, X, noise_floor=-190.0):
"""Predict WiFi (0) or Bluetooth (1) from raw PSD data"""
X_clean = X.copy()
X_clean[X_clean == -np.inf] = noise_floor - 10
X_processed, event_widths = self.preprocess_signal(
        X_clean, noise_floor
)
X_features = self.extract_features(X_processed)
y_proba = self.model.predict_proba(X_features)[:, 1]
y_pred = (y_proba > 0.5).astype(int)

for i in range(len(y_pred)):
        width = event_widths[i]
        confidence = max(y_proba[i], 1 - y_proba[i])
        if confidence < 0.85:
        if width < self.width_threshold:
               y_pred[i] = 1  # Bluetooth
        else:
               y_pred[i] = 0  # WiFi
return y_pred

def predict_proba(self, X, noise_floor=-190.0):
"""Predict class probabilities with event detection preprocessing"""
X_clean = X.copy()
X_clean[X_clean == -np.inf] = noise_floor - 10
X_processed, _ = self.preprocess_signal(X_clean, noise_floor)
X_features = self.extract_features(X_processed)
return self.model.predict_proba(X_features)
```

References

[1] Reilly, J. C., Scheina, R. L., Friedman, N., Eller, E. M., & Guilmartin, J. F. (2024, January 1). Warship. Encyclopedia Britannica. https://www.britannica.com/technology/naval-ship

[2] Wang, L., Chen, Z., & Zhang, Y. (2021). Wireless signal modulation identification method based on RF I/Q data distribution. *Scientific Reports*, *11*(1), 21383. doi:10.1038/s41598-021-00723-7

[3] Alam, S. S., Chakma, A., Rahman, M. H., Bin Mofidul, R., Alam, M. M., Utama, I. B. K. Y., & Jang, Y. M. (2023). RF-Enabled Deep-Learning-Assisted Drone Detection and Identification: An End-to-End Approach. *Sensors*, *23*(9). doi:10.3390/s23094202

[4] Youssef, K., Bouchard, L., Haigh, K., Silovsky, J., Thapa, B., & Valk, C. V. (2018). Machine Learning Approach to RF Transmitter Identification. IEEE Journal of Radio Frequency Identification, 2(4), 197–205. doi:10.1109/JRFID.2018.2880457

[5] Medaiyese, O.O.; Ezuma, M.; Lauf, A.P.; Adeniran, A.A. Hierarchical Learning Framework for UAV Detection and Identification. IEEE J. Radio Freq. Identif. 2022, 6, 176–188