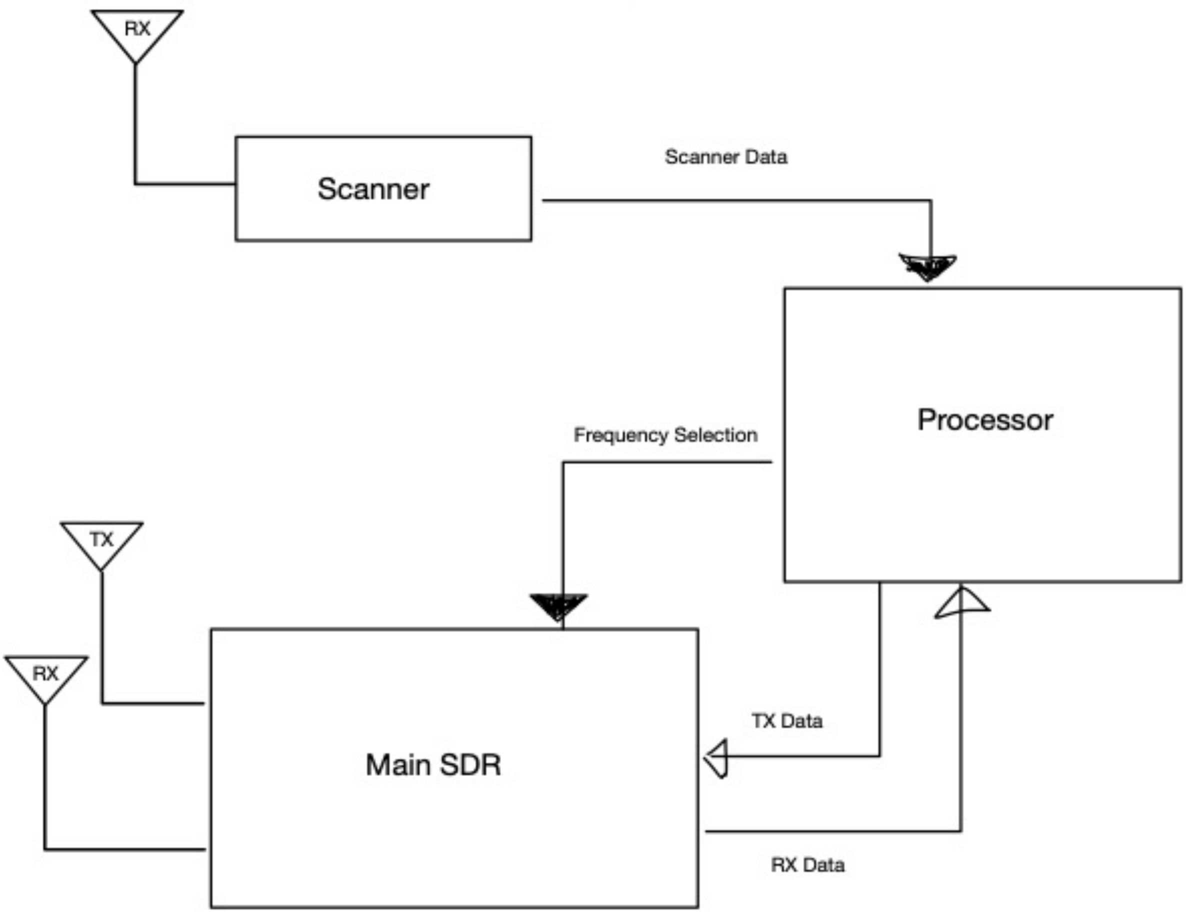


# **Outline**

Goal: Implement a cognitive radio that changes frequencies based on a decision from a ML algorithim. Decision is based on throughput optimization or to remain undetected (using DSSS).

Start with simulating in MATLAB.

- Test spectrum sensing algorithms.
- Validate machine learning (ML) models for frequency selection.
- Analyze the feedback loop between scanning SDR and main SDR.
- Simulate a multi-user wireless network with:
  - Primary users (licensed users) transmitting at specific frequencies.
  - Secondary users (your cognitive radio) that adapt based on spectrum availability.
- Incorporate real-world parameters: noise, fading, interference, and dynamic spectrum occupancy.

  - Use FFT to analyze power spectral density (PSD).
  - Experiment with energy detection, matched filtering, or cyclostationary feature detection.

  - Noise filtering (e.g., using a Gaussian filter).
  - Feature extraction (e.g., SNR, bandwidth utilization).

Develop and Train the ML Model

A. Collect Simulated Data

- Record spectrum usage patterns, PSD data, and interference levels from the simulation.
- Label data with optimal frequency decisions based on predefined rules.

B. Train Initial ML Models

1. Model Type:
   - Start with a supervised learning model like Random Forest or XGBoost for quick prototyping.
   - Move to Reinforcement Learning (e.g., Deep Q-Networks) for real-time decision-making.

2. Features for Training:
   - Frequency bands and their occupancy.
   - SNR, SINR, and channel quality.
   - Time-based spectrum usage patterns.

3. Performance Metrics:
   - Accuracy of frequency selection.

  - Spectrum efficiency.
  - Interference avoidance.

C. Test ML Model in Simulation
- Validate the ML model by integrating it into the simulation.
- Measure latency and accuracy in spectrum decision-making.

---

Prototype the Feedback Loop
A. Simulate the Main SDR
- Model the main SDR as a transmitter-receiver pair in the simulation.
- Implement dynamic frequency tuning based on ML outputs.

B. Realize the Feedback Mechanism
- Simulate a loop where the main SDR reports back metrics (e.g., BER, throughput).
- Allow the ML model to update its decisions based on this feedback.

---

Transition to Hardware Prototyping
A. Select Hardware
- Scanning SDR: Use an affordable RX-only SDR like RTL-SDR for initial tests.
- Main SDR: Use a more capable SDR like Ettus USRP or LimeSDR with TX/RX capabilities.

B. Replicate the Simulation in Hardware
1. Port spectrum sensing algorithms to the scanning SDR using tools like GNU Radio or SDR-specific libraries.
2. Use Python, TensorFlow, or MATLAB to run the ML model on a local PC or edge device.
3. Implement the feedback loop in real-time using protocols like UDP or ZeroMQ for communication between SDRs.

C. Field Test in a Controlled Environment
- Test in a lab setting with controlled spectrum usage (e.g., using signal generators).
- Measure system performance metrics in real-time.

---

Optimize and Scale the System
A. Real-World Testing
- Deploy in real-world spectrum environments to gather diverse data.
- Use this data to retrain and fine-tune the ML model.

B. Optimize Hardware and Software
- Minimize latency by using FPGAs or GPUs for ML inference.
- Optimize SDR firmware for faster retuning and data transfer.

C. Add Advanced Features
- Implement advanced spectrum sensing techniques (e.g., cooperative sensing using multiple scanning SDRs).
- Introduce energy-efficient transmission strategies.

---

Step 7: Final Deployment
- Deploy the system in its intended environment with full compliance to regulatory requirements.
- Continuously monitor performance and update ML models through online learning.