Spectrum Sensing System

July 8, 2025

High-Level Tasks in Cognitive Radio and Spectrum Sensing

1. **Power Spectral Density (PSD) Estimation.** This task involves estimating the distribution of power across different frequency components of a signal. It is fundamental to understanding the spectral characteristics of the observed environment.
   (a) *PSD Estimation.*
   (b) *Quantitative Reconstruction of PSD.* This involves reconstructing the original PSD from noisy or undersampled data using techniques such as interpolation, smoothing, or model-based estimation.
   (c) *Spectral Parameter Estimation.* Key features derived from the PSD include:
      - Peak frequencies (locations of maximum power),
      - Bandwidth (range of occupied frequencies),
      - Power levels at specific frequencies,
      - Center frequency (mean spectral centroid).
   (d) *Demodulation Support.* PSD estimation assists in identifying carrier frequencies and modulation schemes, facilitating demodulation of the received signals.
2. **Signal Detection.** This task focuses on determining the presence of informative content within the received signal.
   (a) *Detection of Signal Presence (Thresholding).* A threshold is applied to a test statistic (e.g., energy) to decide whether a signal is present.
   (b) *Symbol Detection.* Once signal presence is confirmed, the next step is identifying the transmitted symbols, such as bits in digital communication systems.
3. **Identification and Classification of Signals.** This task involves characterizing signals based on observed features.
   (a) *Modulation Classification.* Determining the modulation format (e.g., AM, FM, QAM, PSK) is critical for enabling correct demodulation and decoding.
   (b) *Spatio-Temporal Identification of Spurious Signals.* This involves localizing and characterizing unwanted signals across both spatial and temporal domains.
   (c) *Spatio-Temporal Identification of Interference.* Identifying and profiling interference sources that affect the desired signal, considering their spatio-temporal evolution.
4. **Channel State Estimation.** Characterizing the wireless channel is essential for compensating for impairments such as fading or noise.
   (a) *Channel Impulse Response (CIR).* Estimation of how the channel distorts the temporal profile of the signal.
   (b) *Channel Frequency Response (CFR).* Estimation of the channelâĂŹs effect on signal amplitude and phase across frequency.
5. **Decision-Making.** Based on the outcomes of the previous steps, appropriate decisions are made to complete the communication or control loop.
   (a) Decoding of the received data.
   (b) Choosing an optimal action in adaptive or control-oriented systems.
   (c) Interference identification and mitigation strategies.

System	Year	Application Focus
`GBSense`	2024	Wideband RF spectrum monitoring with low-cost setup. Utilizes sub-Nyquist sampling for real-time analysis.
`Compressed-Sensing Localization System`	2024	Real-time RF emitter detection and localization for non-cooperative civilian radio monitoring using compressed sensing.

Recent civilian-focused RF spectrum sensing systems developed since 2024.

Implementation Details of *GBSense*: A GHz-Bandwidth Compressed Spectrum Sensing System

Core Architecture and Sampling Strategy

- **Periodic Non-Uniform Sampling via Time-Interleaved ADC (TI-ADC):** Utilizes multiple ADC lanes operating in parallel, interleaved in time, to form structured non-uniform sampling patterns. Enables capture of up to 2 GHz of RF bandwidth with only 400 MHz average sampling rate.
- **Clock Distribution and Synchronization:** A dedicated subsystem ensures precise timing across ADC channels, mitigating jitter and preserving temporal alignment.

Hardware Components

- **Power Splitter Subsystem:** Distributes incoming RF signals to multiple ADC lanes for parallel capture.
- **Time-Interleaved Sampling Subsystem:** Incorporates off-the-shelf ADC modules phased across time to support sub-Nyquist sampling.
- **Logic Device Subsystem:** FPGA or microcontroller-based logic handles sample alignment, decimation, buffering, and transmission to the host processor.

Real-Time Software Integration

- **Low-Power Processor:** A Raspberry Pi processes ADC output for spectral reconstruction using compressed sensing algorithms.
- **Software Pipeline:**
  1. Data decimation and formatting,
  2. Compressed sensing-based spectrum reconstruction,
  3. Spectral detection via thresholding.
- **Latency:** Real-time spectrum sensing achieved with ~30 ms frame processing latency.

Performance Metrics

- **Detection Accuracy:**
  - 100% accuracy for spectrum occupancy below 100 MHz,
  - Over 80% accuracy for 200 MHz occupancy levels.
- **Throughput:** Frame-wise analysis completed in under 30 ms.

System Design Innovations

- **Hardware-Friendly Design:** Avoids analog delay lines typical in multicoset architectures; instead, uses programmable digital timing via TI-ADC.
- **Adaptive Sampling Pattern Control:** Sampling patterns are programmable, allowing adaptation to varying spectral environments.
- **Modular and Cost-Efficient:** Constructed with commercially available components and low-cost processing, suitable for scalable and field-deployable applications.

Summary of Key Specifications

Feature	**Specification**
RF Bandwidth	2 GHz
Average Sampling Rate	400 MHz
ADC Methodology	Time-Interleaved ADC (TI-ADC)
Processor	Raspberry Pi
Frame Processing Latency	~30 ms
Detection Accuracy @ <100 MHz Occupancy	100%
Detection Accuracy @ 200 MHz Occupancy	>80%

Key technical characteristics of the GBSense system.

Implementation Details of the Compressed-Sensing Localization System

System Overview

A prototype designed for *real-time monitoring and localization of non-cooperative RF emitters*, using compressed sensing combined with TDoA (Time Difference of Arrival) measurements :contentReference[oaicite:1]index=1.

Hardware Architecture

- **Multiple Sensing Nodes:** Distributed SDR units capture wideband RF signals.
- **Compressed Sampling at Nodes:** Each node applies a measurement matrix (e.g., Gaussian) to compress incoming signals before transmission to a fusion center :contentReference[oaicite:2]index=2.
- **Fusion Node:** Collects compressed samples from nodes and performs joint reconstruction and localization.

Signal Processing Pipeline

1. **Compressed Sensing Reconstruction:** Implements greedy algorithms (e.g., OMP) or convex recovery to reconstruct wideband signals from under-sampled measurements :contentReference[oaicite:3]index=3.
2. **TDoA Estimation:** Uses the reconstructed signals at multiple nodes to extract arrival-time differences for source localization :contentReference[oaicite:4]index=4.
3. **Localization Algorithm:** Estimates emitter coordinates using TDoA multilateration based on reconstructed timing differences.

Software and Computational Aspects

- **Localization Logic:** Fusion center runs CS recovery and TDoA multilateration algorithms in real time.

- **Mapping Interface:** Localization results are displayed on a digital/interactive map, even in offline settings :contentReference[oaicite:5]index=5.

Performance and Evaluation

- **Sample Compression Ratio:** Significant data reduction at nodes via CS before transmission.
- **Detection Performance:** ROC curves indicate that CS-enabled sensing achieves performance comparable to full-rate sampling even at moderate SNRs :contentReference[oaicite:6]index=6.
- **Localization Accuracy:** TDoA-based multilateration yields precise emitter positions; specific error metrics vary by node geometry and quality.

Innovations and Advantages

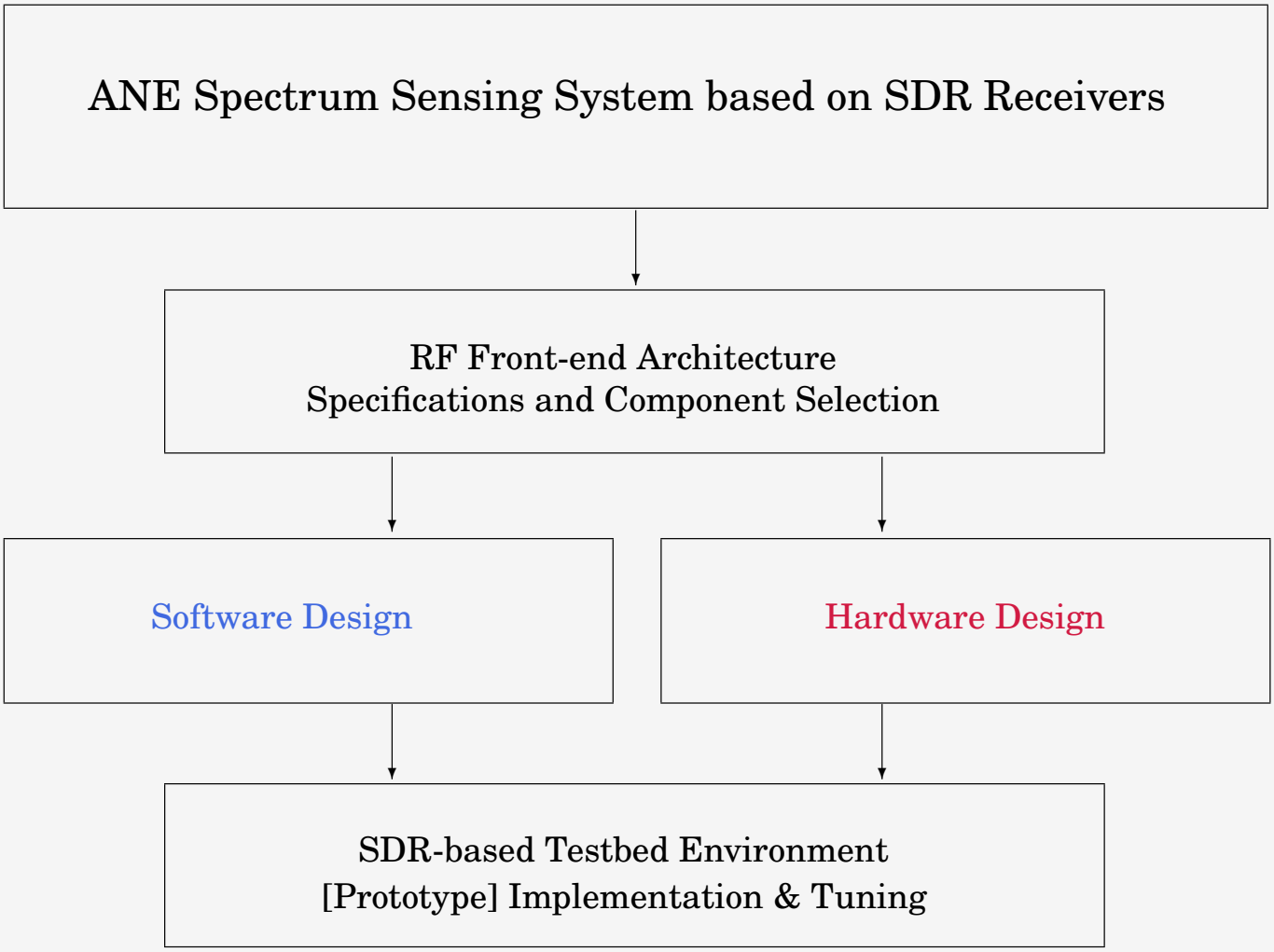
- **Efficient Use of Bandwidth:** Combines sub-Nyquist sampling and compressed sensing to reduce node-to-fusion bandwidth.
- **Scalable Architecture:** Easily extensible by adding SDR nodes to improve localization precision.
- **Real-Time Mapping:** Integration with digital map interfaces enables near real-time emitter tracking.

Key Specifications and Summary

Feature	**Specification**
Number of Nodes	Multiple SDR-based sensing units
Sampling Technique	Compressed sensing (e.g., using random Gaussian projections)
Signal Recovery	Greedy (OMP) or convex CS algorithms
Localization Method	TDoA multilateration on reconstructed signals
Interface	Real-time display on digital/offline map
Performance	ROC curves similar to full-rate systems at moderate SNR

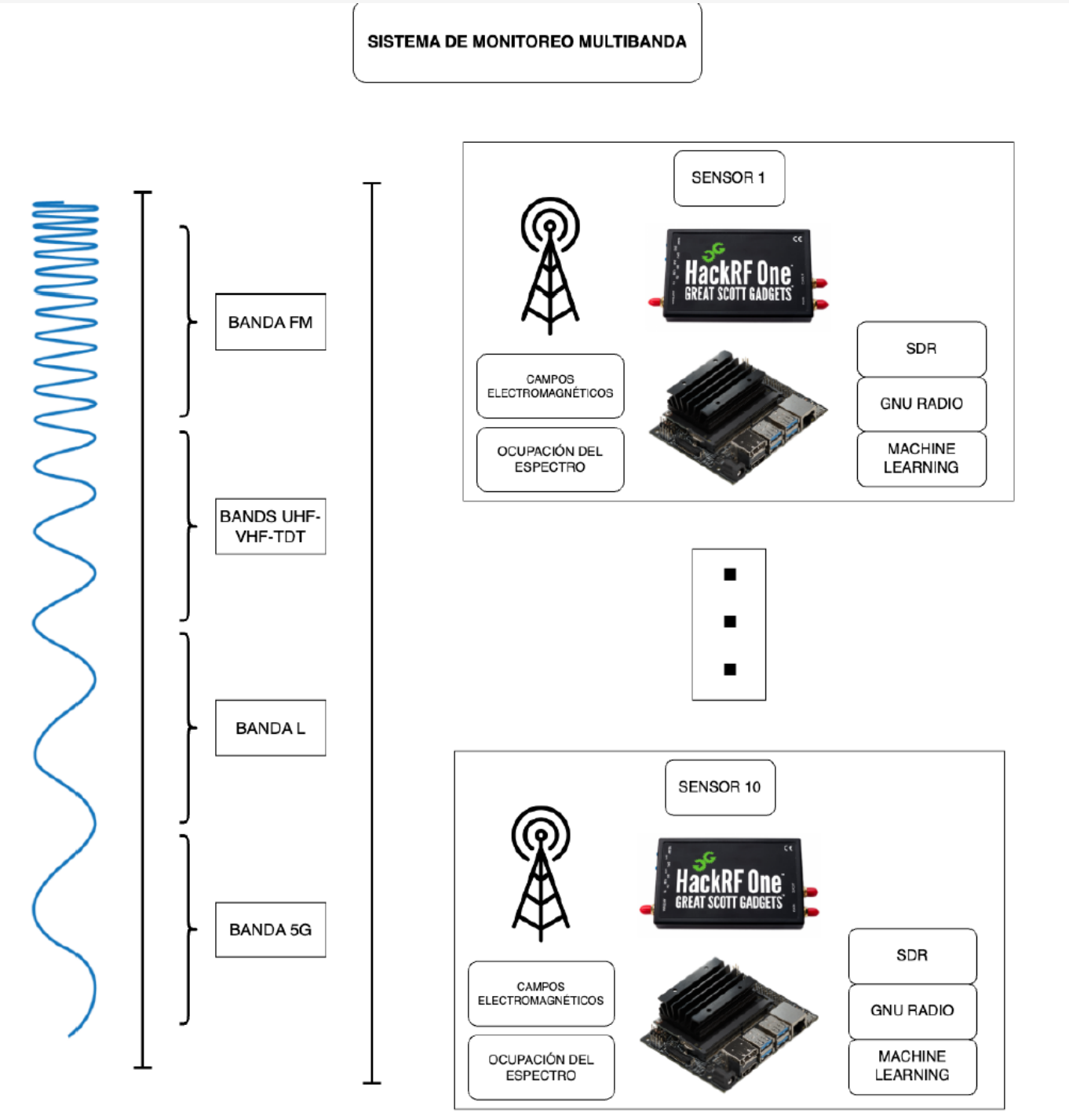
Technical summary of the compressed-sensing localization system.

ANE Spectrum Sensing System



Global Diagram.

Spectrum Sensing Specifications:



Multiband Spectrum Sensing

Spectrum Sensing Networks:

RMER Red de monitoreo de Espectro Radiolectricoc

RMTDT Red de monitoreo de TDT

RCEM Red de Monitoreo de Campos Electromagneticos

Spectrum Sensing Bands and Services: RMER + RMTDT

Band	Notation	Service	Subservice	Sub. Bandwidth
VHF	VHF 1	RDS FM	Radio difusión	88MHz - 108MHz
	VHF 2	Fijo	Fijo y Móvil	137MHz - 144MHz
		MÃşvil		
	VHF 3	Fijo	Fijo y Móvil	148MHz - 174MHz
		Móvil		
UHF	UHF 1	Fijo	Fijo y Móvil	400MHz - 470MHz
		Móvil		
	UHF 2	TDT	TV digital terrestre	470MHz - 512MHz
	UHF 3	Banda L	Fijo y Móvil	1GHz - 2GHz
SHF	SHF 1	5G	Móvil	2555 MHz - 2560 MHz 2675 MHz - 2680 MHz 3300 MHz - 3620 MHz
	SHF 2			

Tabla de bandas y servicios: RMER + RMTDT

Spectrum Sensing Bands and Services: RMEM

Band	Notation	Bandwidth
L Band	L1	1427-1429 MHz
		1429-1452 MHz
		1452-1492 MHz
		1492-1518 MHz
		1518-1525 MHz
	L2	1660.5-1668 MHz
		1700-1710 MHz
		1710-1890 MHz
		1890-1910 MHz
		1910-1930 MHz
		1970-1980 MHz
		1980-1990 MHz

Tabla de Banda L RMEM

Prototype of Electromagnetic Spectrum Monitor

The following components are needed for the design of each module of multipurpose monitoring:

* CanaKit Raspberry Pi 5 Basic Kit (8 GB RAM): This microcomputer features an ARM processor, 8 GB of RAM, and a 32 GB SD card for storage. It serves as the central component of the system.
    - **Power Connection:** Use one of the USB-C cables to connect the power port of the Raspberry Pi 5 to a waterproof USB-C connector mounted on the Dexson Junction Box. This allows a safe external power supply to the system.
    - **Data Connection:** Connect another USB-C cable from the data port of the Raspberry Pi to a second waterproof USB-C connector in the box. This connector can be used for programming or data transfer to the Raspberry Pi from external devices.
    - **Network Connection:** If a local network connection is required, connect an Ethernet cable from the RJ45 port of the Raspberry Pi to one of the waterproof RJ45 connectors in the box.
* CanaKit Pi 5 Case for Raspberry Pi 5 with Active Cooler: A case specifically designed for the Raspberry Pi 5, which includes an active cooler to maintain optimal operating temperature.
* Nooelec HackRF One: An SDR (Software Defined Radio) device that enables reception and transmission of radio signals across a broad range of frequencies.
    - **Antenna Connection:** Two external antennas are connected to the system, which will be used by the HackRF One for signal reception and transmission.
    - **Wiring:**
        - **Coaxial Cable x2:** Connect each of the two antennas to their respective waterproof SMA connectors mounted on the box. These connectors maintain the external connection secure and protected against environmental conditions.
        - **Switch:** Internally, connect the coaxial cables from the SMA connectors to a 2-position switch. This switch will allow selecting which of the two antennas will be active at any given time.
        - **HackRF One:** From the switch, connect another coaxial cable to the SMA connector of the HackRF One, allowing the selected antenna's signal to be transmitted or received by the SDR device.
    - **Switch Control:**
        - **Integration with Raspberry Pi:** The 2-position switch will be directly controlled by the Raspberry Pi, using a GPIO (General Purpose Input/Output) of the Raspberry Pi to activate the antenna change based on system needs.
        - **Automation:** A script can be programmed on the Raspberry Pi to automatically select the most appropriate antenna according to signal conditions or frequencies of interest.
    - **Connection to Raspberry Pi:** The HackRF One connects to the Raspberry Pi via an additional USB-C cable (if necessary, a USB-A to USB-C adapter could be used).
* ANT500: An adjustable telescopic antenna designed for use with SDR devices like the HackRF One, covering frequencies from 75 MHz to 1 GHz.
* Taoglas TG.66.A113: A compact GPS antenna intended for navigation and geolocation applications, compatible with the Mini GPS/BDS Unit (AT6558).
* Mini GPS/BDS Unit (AT6558): A GPS/Beidou module that provides position, velocity, and time data via a serial connection.

    - **Connection to Raspberry Pi:** The GPS unit connects to the Raspberry Pi via a serial port or using a USB to Serial adapter, depending on the configuration. Use a USB-C cable for data connection if an adapter is required.

* Dexson Junction Box: A box designed for mounting components and waterproof connectors.
    - **Component Mounting:** The main components (Raspberry Pi, HackRF One, and Mini GPS/BDS) should be mounted inside the Dexson Junction Box using mounts and adapters that can be fabricated via 3D printing.
    - **Waterproof Connector Installation:** The waterproof SMA connectors (for the antennas), USB-C connectors (for power and data), and RJ45 connectors (for network connection) will be mounted on the walls of the box. Ensure they are well sealed to maintain waterproofing.
* Waterproof RJ45: RJ45 connectors intended for Ethernet connections, offering protection against water and dust in outdoor environments.
* Waterproof USB-C: Waterproof USB Type-C connectors are used for power and data transfer between the Raspberry Pi and other devices.
* Waterproof SMA: SMA connectors are designed for antenna connections, ensuring waterproofing in outdoor applications.
* Coaxial Cable: Coaxial cable are used for connecting antennas to radio and GPS devices within the system.
* Coaxial Cable (3FT): A 3-foot long coaxial cable suitable for longer or external connections.
* USB-C Cable: USB Type-C cables for power and data communication between the Raspberry Pi and other modules.
* 2-Position Switch: A manual switch that allows for safe powering on and off of the monitoring system.
    - **Installation and Connection:** The 2-position switch should be mounted in the box in an accessible location. This switch allows selecting which of the two antennas will be active at any given time. It is internally connected to the coaxial cables of the SMA connectors coming from the antennas, directing the signal to the HackRF One. This allows the Raspberry Pi to control the antenna change according to system needs, ensuring that the most appropriate antenna is always used for signal reception or transmission.
* 3D Printing and Tools: Materials and tools necessary for the 3D printing of the mounts, adapters, and other custom components used in the assembly.

Operating System and Software Configuration on Raspberry Pi

* Operating System
    - **Arch Linux ARM:**
        - **Installation:** A version of Arch Linux ARM is used on the Raspberry Pi, a distribution known for its flexibility and minimalism. This approach allows creating an optimized and lightweight environment, suitable for dedicating all Raspberry Pi resources to monitoring tasks.
        - **Configuration:**
            * **Base Packages:** Only the essential packages are installed to reduce resource consumption and ensure optimal performance.
            * **Kernel:** Consider compiling an optimized kernel focused on support for communication devices and the necessary modules for HackRF One and other hardware interfaces.
* Development and Processing Environment

- **Python:**
  - **Installation:** Python is installed along with the necessary packages for signal processing and hardware control, such as `numpy`, `scipy`, and `pySerial`.
  - **Development:** Python scripts are developed for the following purposes:
    - Control the antenna switch using GPIO.
    - Capture and process radio frequency signal data using the HackRF One.
    - Handle the interface with the GPS unit and process geolocation data.

* Neural Network Implementation with TensorFlow Serving
  - **TensorFlow Serving:**
    - **Installation:** TensorFlow Serving is configured on Arch Linux ARM to deploy neural network models used in real-time signal processing.
    - **Neural Network Model:**
      - **Development:** Neural network models are trained in a separate environment (e.g., on a server with greater computing capacity) and then deployed on the Raspberry Pi using TensorFlow Serving.
      - **Integration:** A Python API is implemented to interact with TensorFlow Serving to process real-time data, such as signal classification, pattern detection, or any other task related to monitoring.

* System Automation and Management
  - **Automation Scripts:** Python or Bash scripts are developed to automate system startup, network interface configuration, antenna selection, and TensorFlow Serving startup.
  - **System Monitoring:** Monitoring and logging tools are implemented to track system performance and detect possible failures or connection issues.

* Configuration Summary
  - **Arch Linux ARM** will be the base operating system, providing a minimalist and efficient environment for the Raspberry Pi.
  - **Python** will be the primary language for signal processing and hardware control, including antenna switch management and data capture.
  - **TensorFlow Serving** will be used to deploy and run neural network models, enabling advanced processing of the captured signals in real-time.

Direction-Of-Arrival (DOA) Estimation Method The method presented in this paper for DOA estimation using HackRF One offers several significant advantages and challenges when compared to traditional approaches [**?**].

- Advantages. One of the primary advantages of this method is the cost-effectiveness of the system. Traditional DOA estimation systems typically require expensive, high-precision RF equipment that can accurately maintain phase coherence and synchronization across multiple channels. By using HackRF One, a low-cost and open-source Software Defined Radio (SDR) platform, the authors have significantly reduced the financial barriers to implementing such systems. This makes the technology more accessible to a broader range of applications, including educational, research, and low-budget operational environments.

Another key advantage is the flexibility provided by SDR technology. The reconfigurability of HackRF One allows for adjustments in frequency, bandwidth, and other parameters via software, rather than hardware. This flexibility is particularly valuable in scenarios where the system must be adapted to different signal environments or experimental setups without the need for significant hardware changes.

The authors have also demonstrated that, despite the lower cost, the system is capable of achieving synchronization errors within one sampling period, which is crucial for maintaining the phase accuracy required for effective DOA estimation. The use of the MUSIC algorithm further enhances the system's ability to accurately estimate the direction of signals, even in complex scenarios involving multiple signal sources.

- Challenges and Limitations

  However, the method is not without its challenges. One of the primary limitations is the resolution and accuracy of the DOA estimation when multiple signal sources are involved. As noted in the experimental results, when the angle difference between two sources becomes small, the system struggles to distinguish between them, resulting in reduced resolution. This limitation is partly due to the inherent phase noise and multipath effects that are more difficult to mitigate in lower-cost SDR systems compared to high-end RF equipment.

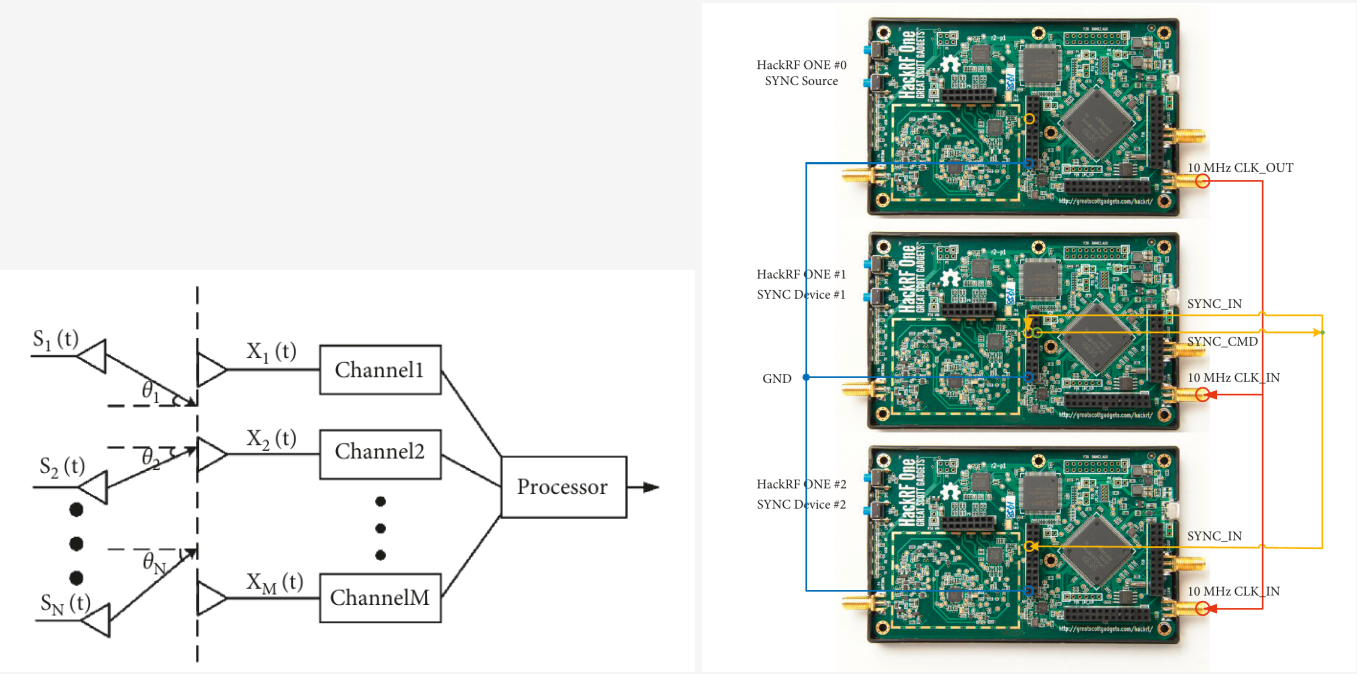


Diagram of the Electromagnetic Spectrum Monitoring System

Additionally, the need for precise synchronization between multiple HackRF One devices introduces complexity to the system setup. Although the authors have implemented effective solutions for clock and time synchronization, this process requires careful calibration and may be susceptible to environmental factors that can introduce synchronization errors.

The dependence on software-defined solutions also means that the system's performance is closely tied to the computational capabilities of the host machine. While the system is designed to be low-cost, achieving high accuracy in real-time applications may require more powerful computing resources, particularly when processing large volumes of data or running complex algorithms like MUSIC.

- Potential Improvements

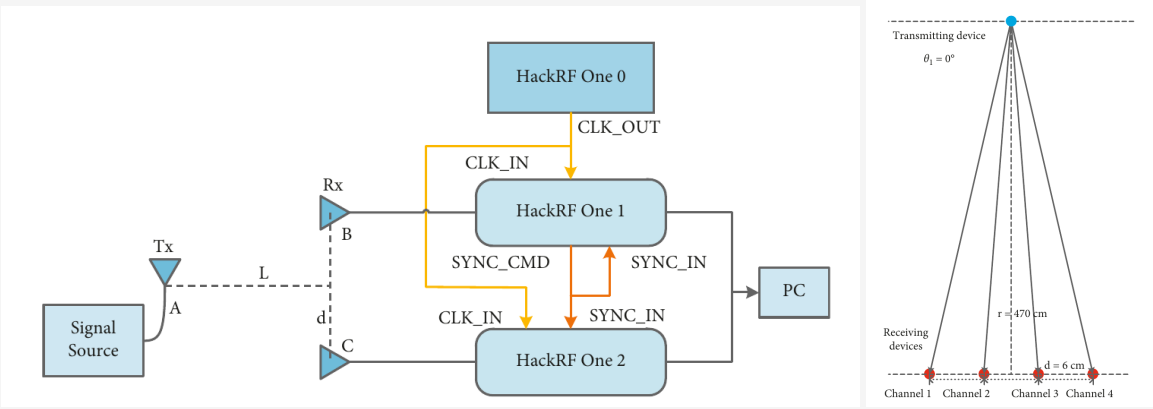


Diagram of the Electromagnetic Spectrum Monitoring System

Future work could focus on improving the system's resolution for multiple signal sources by exploring more advanced signal processing algorithms or by enhancing the synchronization mechanism to further reduce phase noise. Additionally, integrating machine learning techniques for adaptive signal processing could help mitigate some of the limitations related to multipath effects and environmental noise.

Another potential improvement is the development of more user-friendly software interfaces that simplify the setup and calibration process, making the system more accessible to non-expert users. This could involve creating automated tools for synchronization calibration or integrating real-time diagnostic tools to monitor system performance during operation.

- Conclusion

  Overall, the method presented in this paper represents a significant step forward in making DOA estimation technology more accessible and cost-effective. While there are challenges and limitations associated with the use of low-cost SDR devices like HackRF One, the potential applications and benefits of such a system are considerable. With further refinement and development, this approach could become a viable option for a wide range of DOA estimation tasks in various fields, from academic research to practical engineering applications.

Medición de Campo Eléctrico RCEM

Preparación del Equipo - Antenas

- **Antenas:**
  - Toagle Antenna: área efectiva $A_e = \frac{\lambda^2 G}{4\pi}$

  - ANT 500 Antenna:

Preparación del Equipo - Transmisor, Receptor y Software

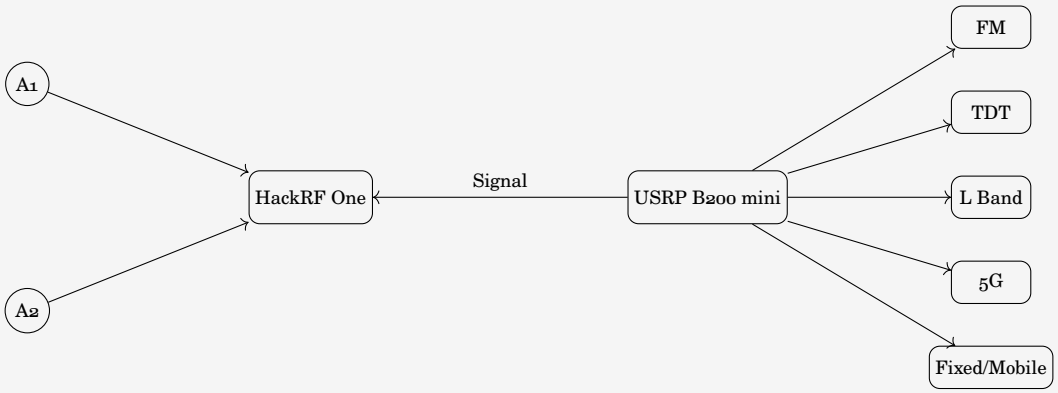
- **Transmisor:** USRP B200 mini.

- **Receptor:** HackRF One.

- **Software:** Python para control de transmisión/recepción.

Configuración del Sistema

1. Conectar las antenas al HackRF One.
2. Configurar el USRP B200 mini con Python para transmitir en las siguientes bandas:
   - Servicios Fijos y Móviles
   - FM (88-108 MHz)
   - TDT (470-790 MHz)
   - Banda L (1-2 GHz)
   - 5G (hasta 3.5 GHz)
3. Configurar HackRF One para recibir en las frecuencias correspondientes.



Procedimiento de Medición

- **Calibración inicial:** Medición de señal de referencia.
- Captura de datos con HackRF One en cada banda.
- Registro de intensidad de campo eléctrico con dispositivo manual.
- Cálculo de potencia en dBm con la ecuación:

$$P_{dBm} = 10 \log_{10} \left(\frac{P_{mW}}{1mW} \right)$$

- $\eta = 377$ ohmios (impedancia del espacio libre).

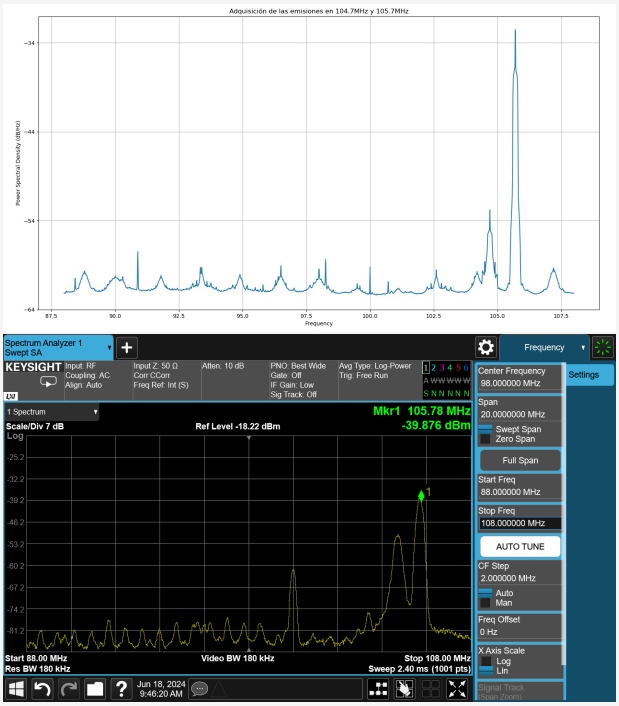
Index	Emisora	Frecuencia (MHz)	Potencia (dBm)	Campo Electrico 377ω (V/m)
0	Olímpica Stereo	89.7	-9.52	0.2052
1	Radio Nacional de Colombia	90.1	-14.29	0.1185
2	Radio Uno	91.7	-8.21	0.2386
3	La W Radio	92.5	-17.09	0.0858
4	La FM	94.3	-5.90	0.3114
5	Tropicana	95.5	-11.61	0.1612
6	RCN Radio	98.7	-15.02	0.1089
7	La Mega	100.7	-14.27	0.1188
8	Los 40	101.7	- 18.29	0.0748
9	Radio Tiempo	102.7	-15.58	0.1022

Datos de las emisoras con sus frecuencias, potencias en dBm y campos electricos a 377ω.

Validación Experimental

- Uso de analizador de espectro y dispositivo de medición de campo manual.

- Comparación de datos entre HackRF One y dispositivos externos.



- Calibración adicional si se encuentra discrepancia.
- Confirmación de precisión en bandas objetivo.

Pruebas en Bandas Específicas

- **Servicios Fijos y Móviles:**
  - Cálculo de la distancia mínima usando la fórmula de Fresnel.

$$d = \sqrt{\frac{4\lambda D}{\pi}}$$

donde λ es la longitud de onda y D es la distancia entre las antenas.

- **FM (88-108 MHz):**
  - Medición en entorno libre de interferencias.
- **TDT (470-790 MHz):**
  - Medición con filtros pasa banda.

- **Banda L (1-2 GHz):**
  - Uso del modelo de Hata para estimar pérdidas.

$$PL(dB) = 69.55 + 26.16\log_{10}(f) - 13.82\log_{10}(h_{\text{trans}}) - a(h_{\text{rec}}) + [44.9 - 6.55\log_{10}(h_{\text{trans}})]\log_{10}(d)$$

donde f es la frecuencia en MHz, h_{trans} y h_{rec} son las alturas de las antenas, y d es la distancia en km.

- **5G (hasta 3.5 GHz):**
  - Cálculo de la primera zona de Fresnel.

$$r_1 = 17.32\sqrt{\frac{d}{4f}}$$

donde r_1 es el radio de la zona de Fresnel en metros, d es la distancia en kilómetros, y f es la frecuencia en GHz.

Conclusiones

- Los resultados son consistentes con dispositivos de medición estándar.
- Este enfoque es aplicable para análisis en FM, TDT, Banda L y 5G.

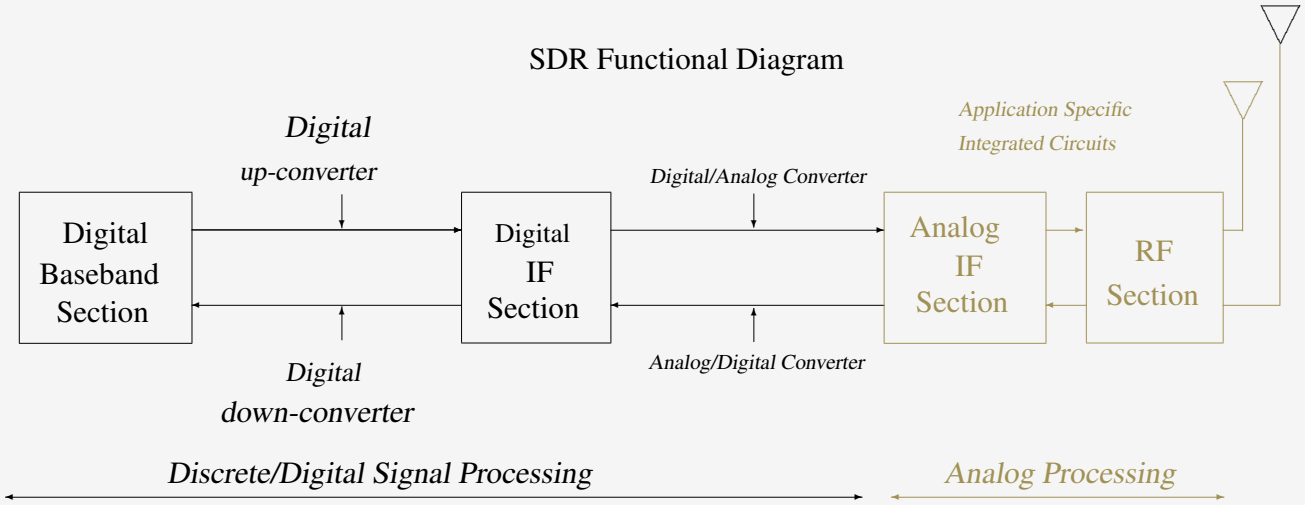
trellis and MER estimation + Signal Identification

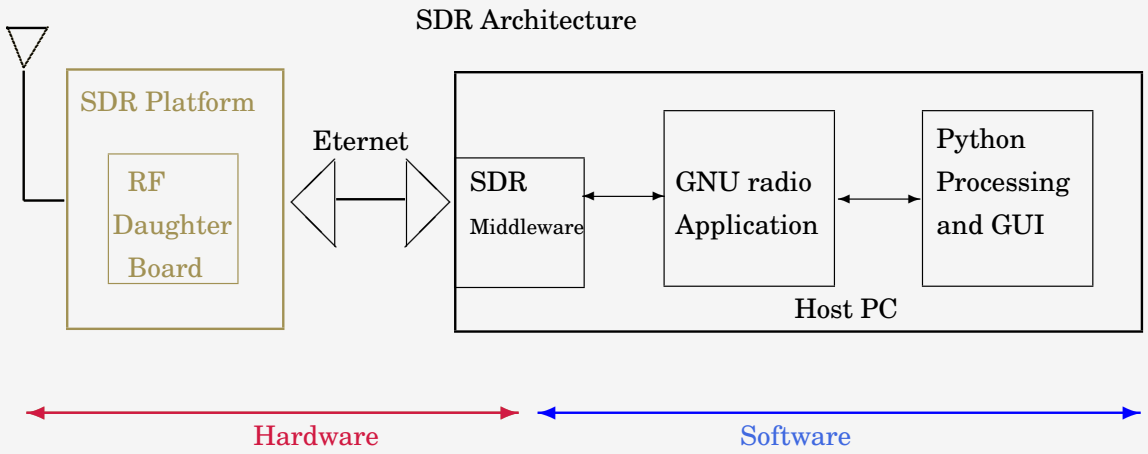
SDR-based Testbed Environment

Software-Defined Radio is a radio communication system in which any physical item or function can be performed or automated by software.

Thus, SDR replaces traditional hardware components through software algorithms, enhancing flexibility and adaptability in radio systems, and enabling the development of CR prototypes and solutions.

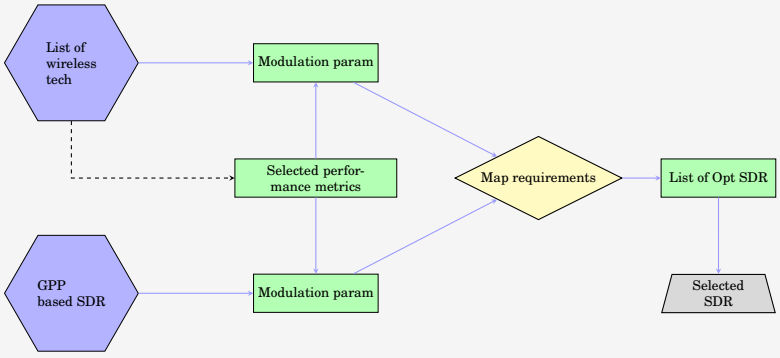
The testbed allows for the evaluation of communication protocols to be implemented in real-world scenarios ●.





SDR-based platform

Hardware: The hardware of an SDR transceiver usually contains components (processing devices) such as a general purpose processor (GPP), digital signal processing (DSP), and Field Programmable Gate Arrays (FPGA). A SDR device employs reconfigurable hardware that may be programmed over-the-air by software to function under different wireless standards.



Selection of SDR Platforms for Wireless Technologies. [•]

Peripherals: Auxiliary devices for Transmission/Reception of radio signals through reconfigurable hardware components (real-time processing devices, middleware, or Open-source electronic prototyping platforms)

– *Antenna Reception System*, determining the signal's propagation characteristics and coverage area.

– *RF Components* such as to filter out unwanted frequencies, and amplify signals to a desired power level.

Software

– *Open source software*: GNU Radio provides signal-processing blocks that are interconnected to form a flow graph representing the implemented transceiver in software

– *Third-Party Software* Compatible with HackRF Baseband Data Processing to extract meaningful information through relevant processing based on the application's requirements.

– *Control and Monitoring Interfaces* through GUI software, allowing to adjust and optimize TX/RX parameters in varying conditions. Post-processing outputs to make available the processed data for use by the application or sending to another system (display, storage unit) for further analysis or action.

SDR-based Spectrum Sensing Platform

Spectrum Sensing Prototype

Data Acquisition & Peripherals

- *SDR Hardware*:
  ✓ SDR Dongle: `HackRF One`

	USRP B210	**LimeSDR**	`HackRF One`	**ADALM Pluto**	**RTL-SDR**
RF Range	70MHz-6GHz	100kHz-3.8GHz	1MHz-6GHz	325MHz-3.8/6GHz	500kHz-1.766GHz
Bandwidth	61.44 MHz	61.44 MHz	20 MHz	20 MHz	2.4 MHz
ADC	12 bits	12 bits	8 bits[1]	12 bits	8 bits
MIMO	2×2	2×2	none[2]	none	none
FPGA	Xilinx Spartan-6	Altera Cyclone IV	none[3]	Xilinx Zynq Z-7010	none

Available SDR platforms

1. Poor representation format
2. Straightforward diversity reception excluded
3. Poor processing capability

✓✗ PC's sound card (ADC/DAC + PC processing) is free and easier to use for RF experimentation.

✗ Maia SDR that is a portable solution using Web-based interface accessed from a smartphone, PC or other device. •

- *SW for Supporting SDR*:
  * Drivers and Libraries to interface with the `HackRF One` SDR:
    ✓✗ GNU Radio[4]
    ✓ Proprietary drivers developed in C from `HackRF One` SD: libHackRf
    ✓ Python. Collab • Deepnote •
  * Programming Environment developed for the Signal Processing and Recognition Group:
    ✓ python-chaski. An advanced distributed communication framework designed to streamline data exchange between nodes over TCP/IP networks. •.
    ✓ python-dunderlab.foundation. A comprehensive Python framework designed to support modular and scalable applications, providing essential tools for data processing, communications, and system management. •.

[4]GNU Radio is a free & open-source software development toolkit that provides signal processing blocks to implement SDR designs

GNU Radio (Python Script)) of installing libraries for `HackRF One`•.

```
pip install pyhackrf
```

```
# Spectrum Sensing Script

import numpy as np     # numerical operations
import matplotlib.pyplot as plt      # visualization
from pyhackrf import HackRF

def main():
  # Initialize HackRF
  hackrf = HackRF()          # Initialize the SDR device.

  try:
    # Configure HackRF
    hackrf.sample_rate = 20e6  # Set the sample rate to 2.4 MHz
    hackrf.center_freq = 100e6 # Set the center frequency to 100 MHz
    hackrf.gain = 40 # Gain level can be adjusted as needed
    #hackrf.gain = 'auto' # Automatically adjust the gain.

    # Read samples
    print("Reading samples from HackRF...")
    samples = hackrf.read_samples(1024*1024)
    # Read 1M samples

  finally:
    # Ensure the HackRF is properly closed
    hackrf.close()

  if __name__ == "__main__":
    main()
```

ARxA *Antenna Reception Array*

* *Single Antenna Reception.* Direct connection to `HackRF One` having a single RF input/output

  ANT1 Omni-directional antenna

  ✓ Omni-directional (adjustable, telescopic) reception antenna ANT500, covering frequencies from 75 MHz to 1 GHz

  ✓ High Gain Omni-Directional FM Reception Antenna •

  ✗ Yagi Antenna, Discone Antenna, Log-Periodic Antenna, vertical ground plane antenna

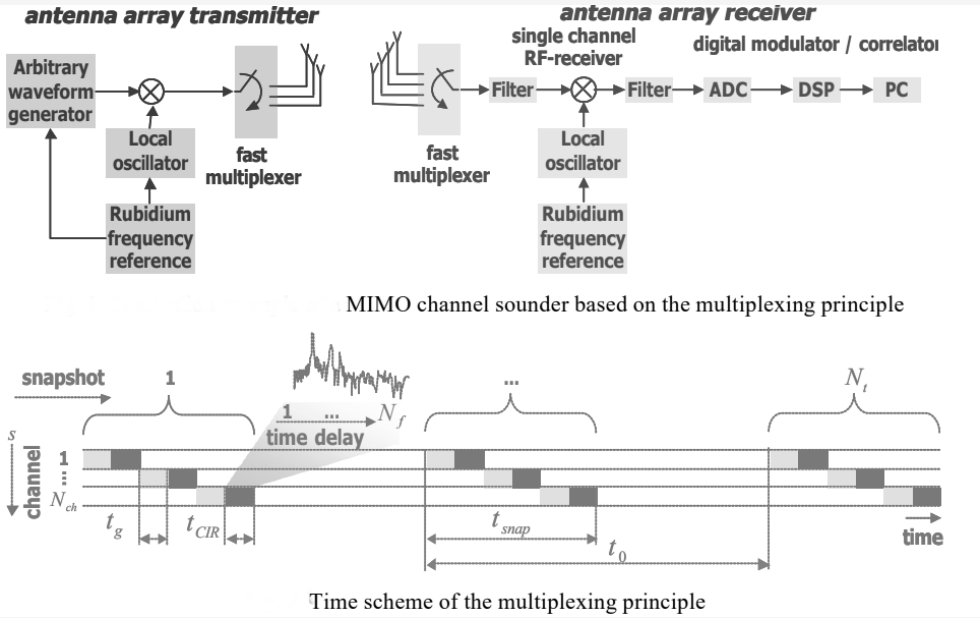
  ANT2 Directional Antenna

  ✓ Taoglas TG.66.A113 A compact GPS antenna intended for navigation and geolocation applications, covering frequencies from 600 MHz to 6 GHz

* ✗ *Two separate* `HackRF One` *devices + Synchronization.* Each SDR is connected to a separate antenna, involving external clock sources or custom synchronization circuits (e.g., for MIMO or coherent signal processing), as `HackRF One` does not natively support phase synchronization across multiple units.

* ✓ *Dual antenna time-multiplexed SDR system.* Two antennas are used in conjunction with a time multiplexing technique to share resources such as a single SDR receiver or transmitter. Synchronization is also required by Software Integration. Time-division multiplexing is used to alternate between the two antennas, allowing each to transmit or receive signals in different time slots, without the need for two separate SDR channels.

  GNU Radio (Python Script) •.



Antenna Multiplex System•

Considerations:
Time Accuracy: Depending on your application, more precise control over time multiplexing maybe needed. Timers or hardware interrupts are used to ensure accurate time control.
Antenna Latency Switching: Some SDR devices have some delays when switching antennas. The switching time for hardware must be tested to ensure it meets MUX requirements.

DASw ✓ *Dual Antenna Switch* •. An external RF switch controlled by an Raspberry Pi GPIO on the `HackRF One` is used to toggle between two antennas connected to the RF switch. `HackRF One` has four GPIO pins (labeled GPIO4, GPIO5, GPIO6, and GPIO7) used to control external hardware like an RF switch.

GNU Radio (Python Script) •.

Consideration
Depending on each application, the switching time may be a concern. Some RF switches might have delays or require debounce logic, especially for fast switching applications.

✓ Enhanced Reception

** Two omni-antennas for Diversity Reception or Scanning Different Bands

** Two Directional Signal Analysis (one directional antenna (e.g., Yagi) towards a specific signal source, while the other (e.g., omnidirectional) covers a broader area)

** FM DX to receive distant or rare FM stations, including the use of UAVs.

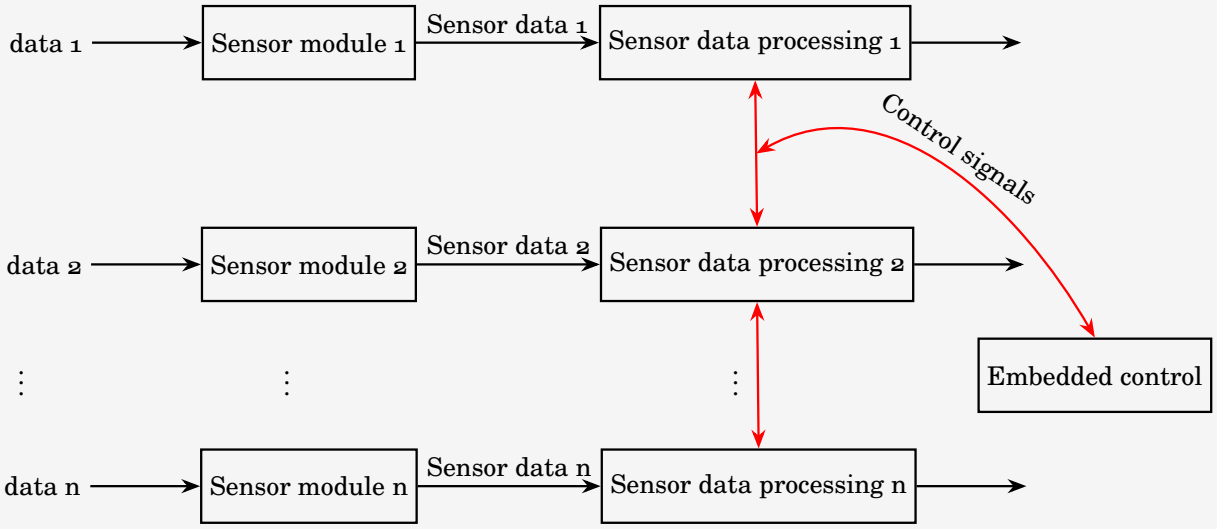
- ✓ *Antenna Rotator* to monitor stations from different directions, allowing to remotely adjust the antenna's direction. Controlling an antenna rotator involves interfacing with external hardware (e.g., using an Raspberry Pi), or some specific serial interface to rotate the antenna to a desired azimuth and/or elevation.

Considerations:
Precision Control of the rotator to accurately respond to azimuth commands. Some rotators may have slight delays, so you there is a need for experimenting with time.sleep() to give the rotator enough time to move.

Signal Strength Integration by processing the `HackRF One`'s received samples after each rotation to decide if further adjustments are needed.

GNU Radio Python Script to Control Antenna Rotator •.

GPSSync ✓ Multi Sensor GPS-based Synchronization •



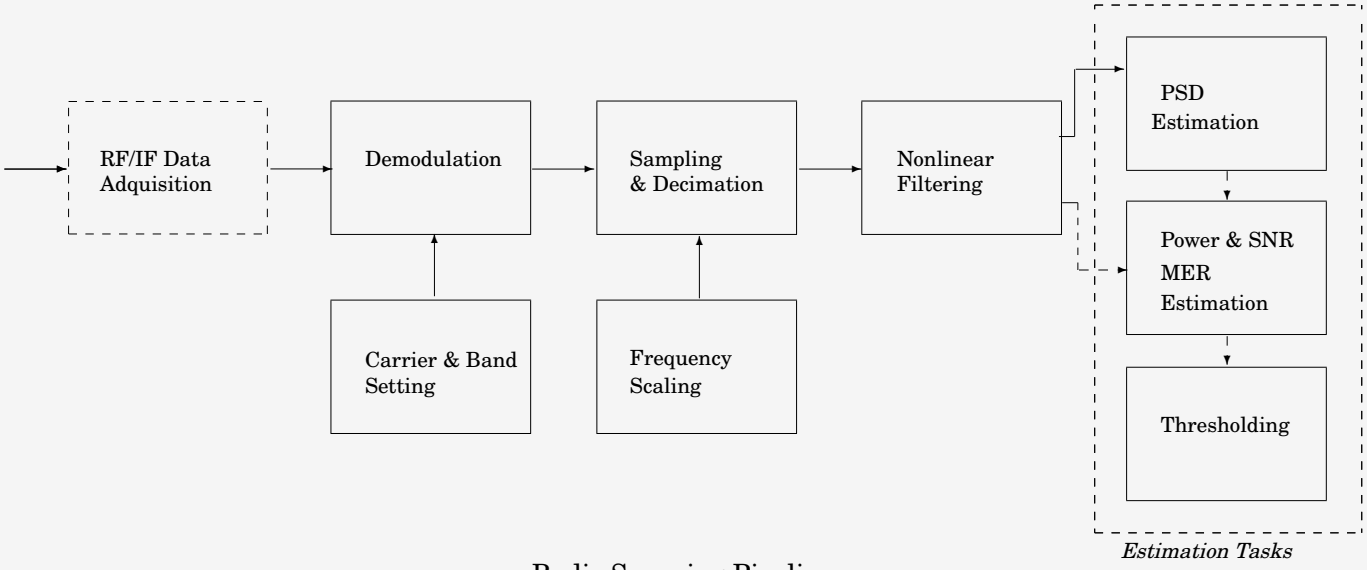
Global Synch Scheme

GNU Radio for GPS-based Synchronization (Python Script) •.

	Bandwidth	**DASw**	**ARxA**	**GPSSync**
VHF 1	88MHz - 108MHz	No	ANT 1	Yes
VHF 2	137MHz - 144MHz	No	ANT 1	Yes
VHF 3	148MHz - 174MHz	No	ANT 1	Yes
UHF 1	400MHz - 470MHz	No	ANT 1	Yes
UHF 2	470MHz - 512MHz	No	ANT 1	Yes
UHF 3	1GHz - 2GHz	No	ANT 2	Yes
SHF 1	2555 MHz - 2680 MHz	No	ANT 2	Yes
SHF 2	3300 MHz - 3620 MHz	No	ANT 2	Yes

Tabla de Notation, Bandwidth, Dual Antenna Switch, ArxA y MGPSync

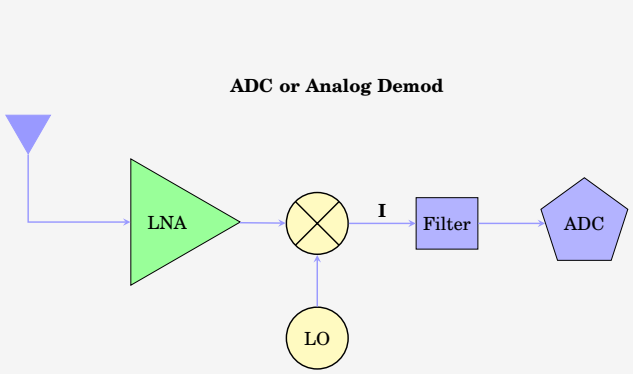
Signal Processing & Radio Scanning



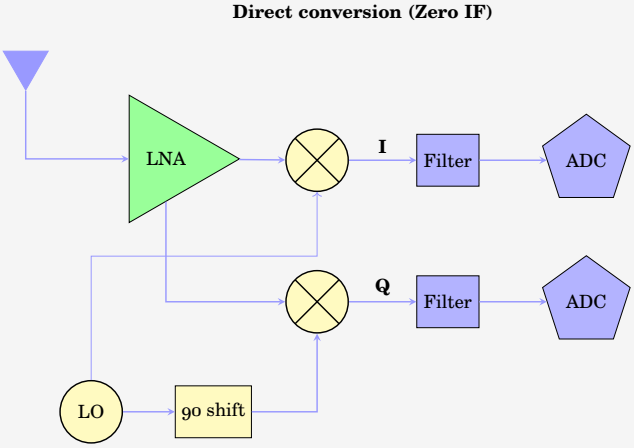
Radio Scanning Pipeline

Demod ✓ *Demodulation*

Basic reception structures to convert RF signals into baseband signals for further processing:



Homodyne receiver



Heterodyne receiver

By default, `HackRF One` SDR uses heterodyne architecture in the front-end.

HomoRx *Homodyne receiver* converts directly the RF signal to baseband without any intermediate frequency stage (Zero-IF), which is more suited for SDR devices focused on DSP. It provides high sensitivity and better image rejection due to the use of FI stages, but is more complex. However, imperfect in-phase (I) and quadrature (Q) signals can cause IQ imbalance, leading to signal degradation.

HeteRx *Heterodyne receiver* is the process of converting an incoming high-frequency (f_{RF}) signal to an intermediate frequency (f_{IF}) before it is down-converted to baseband f_{BB}. The mixer output in a heterodyne receiver is $f_{IF} = | f_{RF} - f_{BB} |$

Samp *Sampling*

OverSam ✓ Oversampled Nyquist Sampling [Oversampling]. The sample rate must be at least twice the highest frequency component of the signal to accurately reconstruct the signal without aliasing.

UnderSam ✓ Undersampled Nyquist Sampling [Undersampling]. Bandpass sampling is a technique where one samples a bandpass-filtered signal at a sample rate below its Nyquist rate. The sample rate is few times under Nyquist rate because of hardware restrictions.

GNU Radio. Python Script for Heterodyne receiver ●.

```
# Block 1:    Signal Source captures the signal.
# Block 2:    Filtering applies a low-pass filter to isolate the IF signal.
# Block 3:    Data Storage the filtered IF data.

  class if_data_acquisition(gr.top_block):
  def __init__(self):
  gr.top_block.__init__(self, "IF Data Acquisition")

  # Parameters
  center_freq = 100e6    # center radio frequency at 100MHz
  samp_rate = 2e6      # sampling rate at 2Mps

  # Block 1
  self.src = osmosdr.source(args="numchan=1")
  self.src.set_sample_rate(samp_rate)
  self.src.set_center_freq(center_freq)
  self.src.set_gain(30)

  # Block 2
  # Low-pass filter to isolate the IF signal
  self.lp_filter = filter.fir_filter_ccf(1,
  filter.firdes.low_pass(1, samp_rate, 200e3, 10e3, filter.firdes.WIN_HAMMING, 6.76))

  # Block 3
  # File sink to store the IF data
  self.file_sink = blocks.file_sink(gr.sizeof_gr_complex*1, "if_data.bin")

  # Connections
  self.connect(self.src, self.lp_filter, self.file_sink)
```

RedSamp *Reduced Rate Sampling*:

Decim ✓ Decimation. In wideband RF systems, the initial sample rate may be very high (e.g., several MHz) to capture a wide range of frequencies. Decimation is used to reduce the sample rate for more manageable processing while preserving the signal of interest.

GNU Radio. Python Script for decimation ●.

Interp ✓ Interpolation can be used to increase the sample rate if necessary for specific processing stages.

GNU Radio. Python Script for Wavelet Interpolation ●.

ModType Information Signal Format

Analog Analog Signal

Digital Digital Signal

Notation	Scheme	Mod Type	Samp	RedSamp
VHF 1	Homodyne	Analog	OverSam	Decimation
VHF 2	Homodyne	Analog	OverSam	Decimation
VHF 3	Homodyne	Analog	OverSam	Decimation
UHF 1	Heterodyne	Digital	OverSam	Decimation
UHF 2	Heterodyne	Digital	OverSam	Decimation
UHF 3	Heterodyne	Digital	OverSam	Decimation
SHF 1	Heterodyne	Digital	OverSam	Decimation
SHF 2	Heterodyne	Digital	UndersSam	Decimation

Tabla de Notation, Scheme, Mod Type, Samp y RedSamp

- *Flow Control of RF Scanning*

  * *Carrier & Bandwidth Setting*
  Adjusting Input Settings: falta enumerar todos los parametros

  ModBandwidth Bandwidth to capture the modulated signal.

  InitSRate Initial Sample Rate

  AudioSRate Audio Sample Rate

  Gain Gain Control settings within the SDR software to balance signal strength with noise reduction.

  LPFil *Low-Pass Filter* for removing high-frequency noise and unwanted components.

```
from gnuradio import filter

# Low-pass filter to remove high-frequency components
lpf = filter.fir_filter_ccf(1, filter.firdes.low_pass(1, sample_rate, 100e3, 10e3
                                                      ))
```

  FreqScaling *Frequency scaling*:
  This procedures in SDR involves adjusting the sampling rate to match the frequency of interest for effectively capturing and processing signals, especially when dealing with different frequency bands.

  NonOver Non-overlapped channel scaling

  Overlap Overlapping by slicing window

  GNU Radio. Python Script for Frequency scaling •.

  * ✓ *Throttling*
  This block is added for controlling the flow of data to match the processing rate. Throttling is used to control the rate at which operations are performed, ensuring that the system does not get overwhelmed by too many requests in a short period.

  Specific aspects of throttling in SDR:

  .- *Sample Rate Control*: SDR systems often need to process signals at specific sample rates. A throttling block can ensure that samples are produced or consumed at a constant rate, matching the desired sample rate of the system.

  .- *Buffer Management*: In SDR, data is often processed in chunks or buffers. Throttling helps manage these buffers to ensure that they do not overflow (too much data too quickly) or underflow (too little data too slowly).

  .- *Synchronization*: Ensures that different components of the SDR system, which may operate at different rates, remain in sync. For instance, the rate at which data is received from an antenna must be synchronized with the rate at which it is processed and possibly transmitted.

  .- *Flow Graph Control*: In many SDR platforms like GNU Radio, flow graphs (which represent the data flow through various processing blocks) include throttling blocks to control the overall flow of data through the graph. This ensures that each block processes data at the correct rate.

  .- *CPU Load Management*: Throttling can also be used to manage CPU load. By controlling the rate of data processing, the system can avoid overloading the CPU, ensuring stable operation.

  GNU Radio. Python Script for Throttling •.

GNU Radio for FM demodulation ●.

GNU Radio for FM demodulation from a simulated source ●.

GNU Radio for Quadrature FM Demodulator ●. vH-perla IF

Signal Processing & Estimation Tasks

- *PSD Estimation*. Real-Time Implementation of Multiband Spectrum Sensing Using SDR •
  * ✓✓ Multi-taper Method of PSD. Estimation based on multiple orthogonal tapers to reduce spectral leakage and variance.

    GNU Radio for PSD – Multi-taper Method •.
  * ✓✓ Wavelet-based PSD estimation with denoising.

    GNU Radio for PSD – Wavelet •.
  * ✓ Welch's Method. Based on averaged histograms

    GNU Radio for PSD – Welch's Method •.
  * ✓ Neural Network-based Estimation

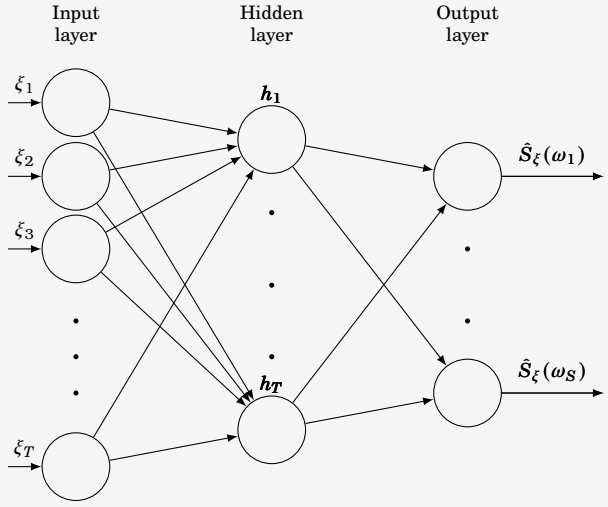
    GNU Radio for PSD – Neural Networks •.

$$x(t) \in \mathcal{L}_2(\mathbb{R}) \quad \text{nonrandom signal} \longrightarrow \text{FT:} \quad \mathcal{F}\{x(t)\} = X(\omega) \in \mathbb{C}$$

$$\text{FT} \longrightarrow \text{DFT} \longrightarrow \text{FFT} \quad \textit{Energy Spectral Density} \longrightarrow \text{Calculation}$$

$$\xi(t) \in \mathcal{L}_2(T) \quad \text{random signal} \longrightarrow S_\xi(\omega) = \mathbb{E}\{\mathcal{F}_T\{\xi\}\} \quad \textit{Power Spectral Density} \longrightarrow \text{Estimation}$$

$$\text{Statistical Estimation} \longrightarrow \text{Welsh's Method} \quad \text{Heuristic Estimation} \longrightarrow \text{MLP}$$



MLP architecture for PSD Estimation

- *Power Strength Estimation*
  - Source Block: '*osmosdr.source*' or equivalent.
  - Frequency Translation
  - Demodulation
  - Power Calculation

  GNU Radio for Power Strength Estimation •.

- *Signal-to-Noise ratio – S/N*

  Real-time computation GNU Radio for S/N •.

- *Energy Detection or Thresholding*

  Detección de canal: Valor reportado: presencia – 1 , ausencia – 0 por canal

  * Energy Detection by L_2-based estimation and Entropy-based detection

    GNU Radio for Thresholding •.

  * Energy detection-based spectrum sensing machine •
  * Maximizing Eigenvalue Using Machine Learning •
  * FlashFFTConv •
  * Energy detection under noise power •
  * Multiscale Wavelet Transform Extremum Detection With the Spectrum Energy Detection •
  * Exploring DL for Adaptive Energy Detection Threshold Determination: A Multistage Approach •
  * ✓ Deep Learning for Adaptive Energy Detection Threshold •

- *Signal Classification*falta
- *Wideband Signal Processing* falta

  Channelization, filter banks, and multichannel sensing

- *Two-antenna diversity*

  * By equal gain combining,

    .- Two SDR Source blocks for the two antennas.
    .- Two Complex to MagPhase blocks to extract magnitude and phase.
    .- Add, Multiply, and Complex Multiply blocks to align phases and combine signals.
    .- A Sink block (e.g., FFT Sink, QT GUI Sink) to visualize the combined signal.

    GNU Radio for Two-antenna diversity–equal gain combining •.

  * By maximal ratio combining.

    .- Setup SDR Receivers: Configure two SDR devices to receive signals simultaneously.
    .- Calculate Weights: Calculate the optimal weights based on the received signal strengths.
    .- Maximal Ratio Combining: Combine the signals using the calculated weights.

    GNU Radio for Two-antenna diversity–equal gain combining •.

*- First-level compression

$$
\begin{aligned}
X(N_{\max};\omega), &\quad \rightarrow \widehat{SNR}, \hat{\gamma} \\
X(N_{\max},\gamma,\widehat{SNR};\omega), &\quad \rightarrow \text{thresholding} \\
X_\eta(N'_{\max},\gamma;\omega) &\quad = \text{AR-model}\{X(N_{\max},\gamma,\widehat{SNR};\omega) - X(N_{\max};\omega)\} \\
X(N_{\text{opt}},\widehat{SNR};\omega) &\quad = X(N_{\max},\gamma,\widehat{SNR};\omega) + X_\eta(N_{\max},\gamma;\omega) \\
&\quad = \text{compress}\{X(N_{\max},\gamma,\widehat{SNR};\omega)\} + \text{compress}\{X_\eta(N_{\max},\gamma;\omega)\} \\
&\quad = [\overline{x}(\omega_n)\geq\gamma, \omega_n : n\in N_X^c] + [\alpha_k : k\in N_\eta^c] \\
&\quad \rightarrow N_X^c + N_\eta^c \ll N_{\max}
\end{aligned}
$$

*- Second-level compression

$$
\begin{aligned}
X_{\eta}(N_{\max}^{'}, \gamma; \omega) &= \text{imputation}\{X(N_{\max}, \gamma, \widehat{SNR}; \omega) - X(N_{\max}; \omega)\} \\
\Xi(N_{\text{opt}}; \omega) &= \text{downsampling}\{\Xi(N^{'}; \omega) : N^{'} \rightarrow N_{\text{opt}}\}
\end{aligned}
$$