# GNU Radio Software Defined Radio University Project-Based Learning Using the Lime Mini SDR 2.0, Raspberry Pi 5.0

Application Focus: Multi-Hop Mesh Networks using the Cluster Duck Protocol

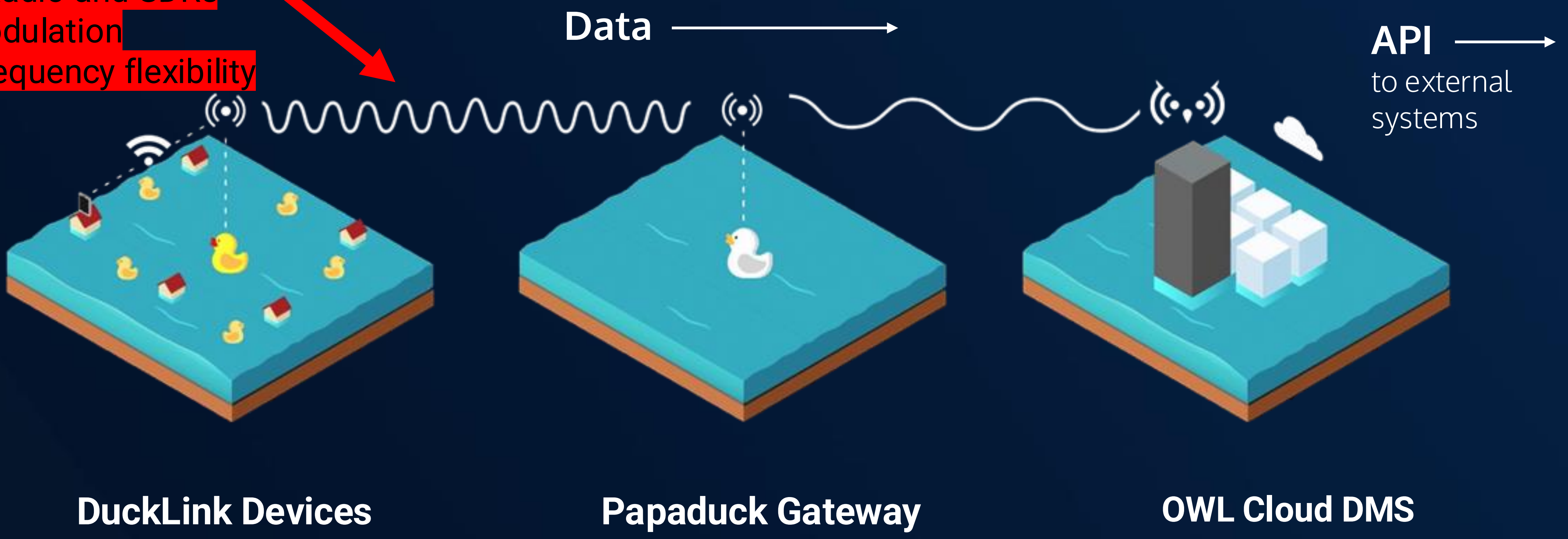
By Liam McCarthy, Benjamin Duval, Steve Dunton, Dennis Derickson

# Outline:

1. *GNU Radio Application Goal* – Physical Layer Flexibility for a Multi-Hop Mesh Network
2. EE504 Software Defined Radio Course – Spring 2024: Getting up to Speed with GNU Radio
3. Our Summer 2024 Efforts to incorporate GNU Radio with the Cluster Duck Protocol
4. Summary and Next Steps.

# **GNU Radio Application Goal** – Open-Source Multi-Hop Mesh Network using the Cluster Duck Protocol

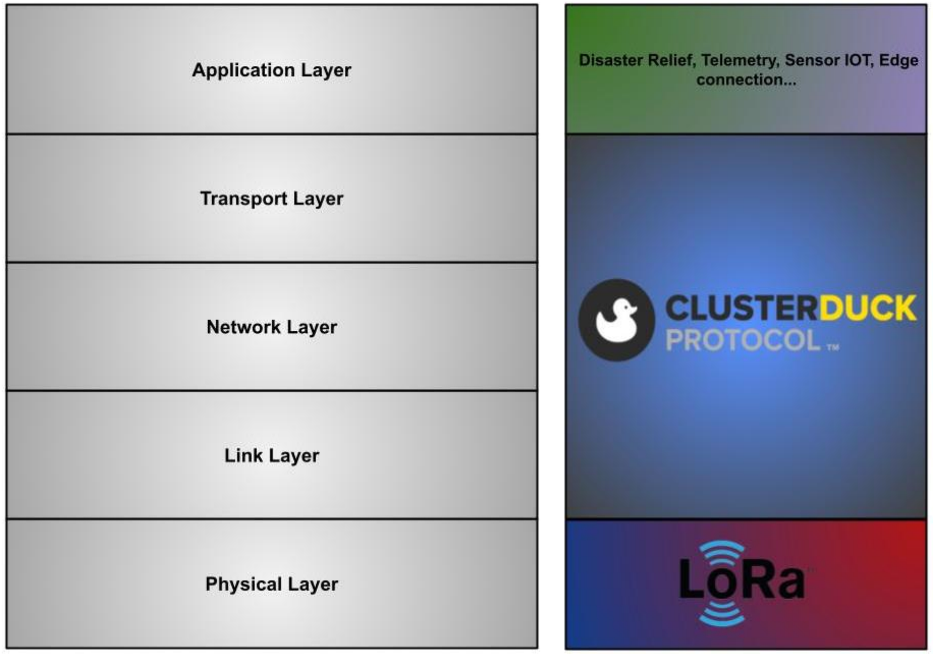
USING LoRa NOW
BUT WANT TO INCLUDE
GNU Radio and SDRs
for modulation
and frequency flexibility

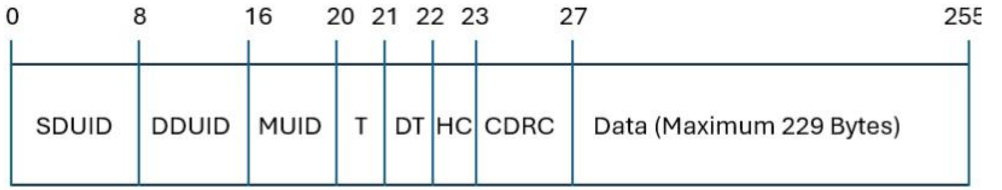



Source: https://clusterduckprotocol.org/

# OWL's Ducks – The Old and the New

- Started Cluster-Duck-Protocol to run "Duck" LoRa Mesh Network Radios https://clusterduckprotocol.org
- A single CDP Packet fits into the LoRa payload
- Open source CDP Software, DIY radios: https://github.com/ClusterDuck-Protocol/ClusterDuck-Protocol
- Sponsored a Senior Project which turned into Summer Research using LimeSDR + Raspberry Pi




| Parameter | Number of Bytes | Data Type | Description |
|---|---|---|---|
| SDUID | 08 | Byte Array | Source Device Unique ID |
| DDUID | 08 | Byte Array | Destination Device Unique ID |
| MUID | 04 | Byte Array | Message Unique ID |
| T | 01 | Byte Value | Topic |
| DT | 01 | Byte Value | Duck Type; 0=Duck Link 1= MamaDuck 2= PapaDuck |
| HC | 01 | Byte Value | Hop Count (The number of times that the packet was relayed) |
| DCRC | 04 | Byte Value | Data Section Cyclical Redundancy Code |
| Data | 229 | Byte Array | Data Payload (could be sensor data or any type of text) |

# Open-Source Hardware and Software for the Cluster Duck Protocol Network and Our Vision for the "QuAD Pro"

## QuAD R1 vs. QuAD RC1 vs. QuAD RC2 vs. QuAD-PRO





***Rev 1 Based on current electronics solution, custom packaging and software.***
**-GPS**
**-LORA**
**-WiFi**
**-Cluster Duck Protocol (CDP)**
-Existing Development
- Ongoing Feature additions

***QUAD RC1 Based on RP Boards***
**-GPS**
**-LORA**
**-WiFi**
**-Cluster Duck Protocol (CDP)**
-In Progress
-First Hardware early spring quarter 24
- Rev 2 planned for Fall Quarter

***QUAD RC2 Based on TI Board***
**-GPS**
**-LORA**
**-WiFi (on processor)**
**-Cluster Duck Protocol (CDP)**
**-hardware security features**
-In Progress
-First Hardware late spring quarter 24
-Rev 2 planned for fall quarter

***QUAD Pro Project***
**-GPS**
**-LORA**
**-WiFi (on processor)**
**-Cluster Duck Protocol (CDP)**
**-improved processing power**
**-video capable**
**-frequency and modulation agile**
**-edge network machine learning**
-In Definition and prototyping stage
-First Hardware theorized late Spring Quarter 2024

*THIS IS THE FOCUS OF OUR EFFORT FOR GNU RADIO*

Source: Dr Dennis Derickson

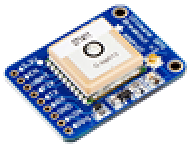
# Provided Hardware used for Senior Project, EE504 and Summer Research

EE504

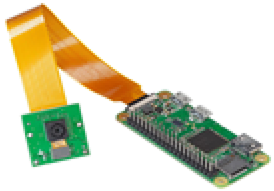
Lime Mini 2.0 SDR

Raspberry Pi 5 kit

Summer Research, QuAD Pro



GPS Module + Antenna



Raspberry Pi Camera


SX1262 LoRa Module

Monitor + Keyboard Setup

Test equipment: Tiny SA, Analog Discovery II. Test Coax cables and attenuators

Old generation Ducks (SX1276)

Additional Testing Hardware

# Software used, provided or found

GNU Radio Companion: https://www.gnuradio.org

DragonOS Raspberry Pi Image:
https://sourceforge.net/projects/dragonos-pi64/

LoRa Out of Tree Module: https://github.com/tapparelj/gr-lora_sdr

LimeSuite: https://github.com/myriadrf/LimeSuite/
LimeSuiteNG: https://github.com/myriadrf/LimeSuiteNG/

GNU Radio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

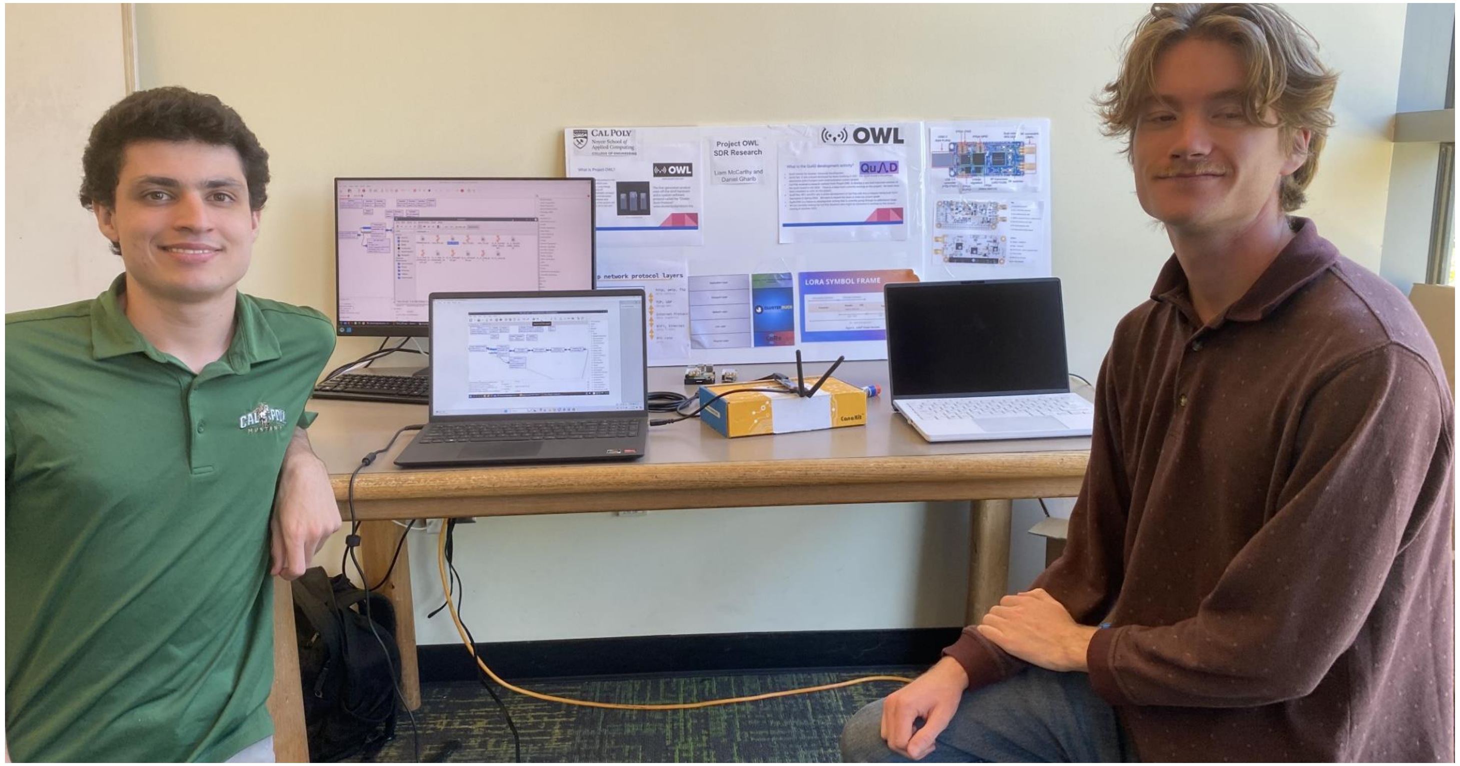

Lime microsystems

# LimeSDR Mini 2.0 Pros/Cons

| Pros | Cons |
|---|---|
| Wide frequency range | Limited output power |
| Highly configurable in GNU Radio without modifying FPGA | Requires good foundation of SDR understanding. |
| Small, Easy to transport and use in field. | Requires a lot of USB power under some situations. |
| Semi-affordable | Complex for beginners. Not easy to get started with as of 2024 |
| Active software support, "next gen" LimeSuite | Current software currently does not work for LimeSDR Mini 2.0, no built-in GR 3.10 OOT Modules |
| SoapySDR Support for GR3.10 | Not ideal for a dedicated solution to one modulation |

# Our Introduction to GNU Radio with the LimeSDR Mini 2.0

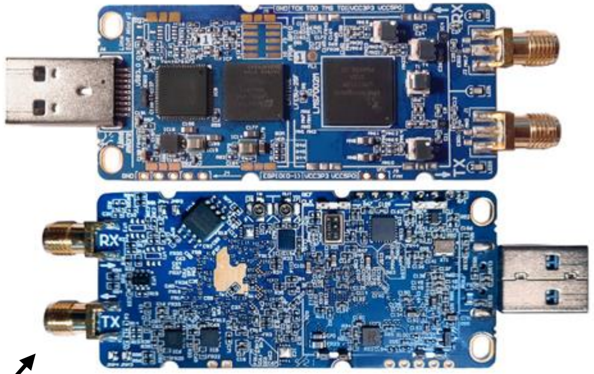
- Senior Project: **Exploration of Software Defined Radios in Mesh Networks**




- Tasked with exploring using Software Defined Radios on a theoretical Raspberry Pi Computer-based radio and pathfinding uses of the LimeSDR 2.0

Question: How do I learn to use my Lime Mini 2.0 and GNU Radio *quickly*

Answer 1: Go to the wiki tutorials!
Answer 2: Also volunteer to TA and develop GNU Radio Lessons for EE504: Software Defined Radio Laboratory




Getting Lime Mini 2.0 Working on my Pi:

- First Step
- Took way longer than expected

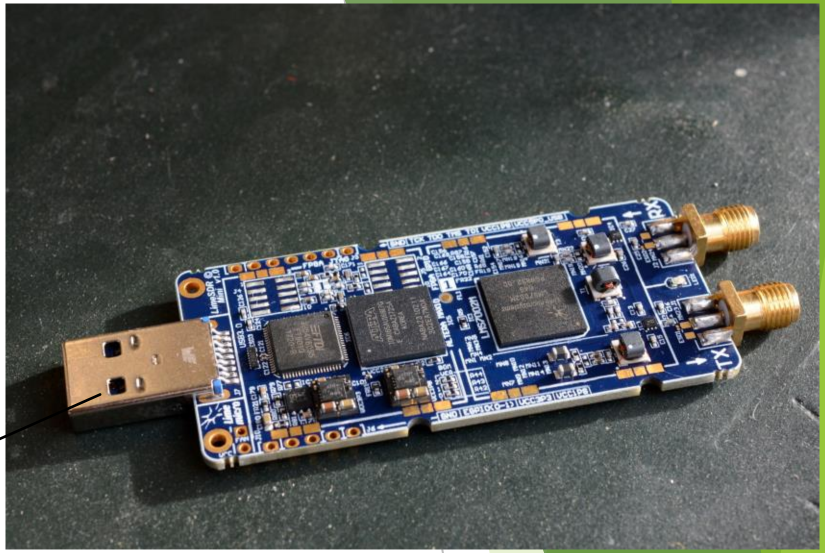

Needed DragonOS to be successful and to be able to quickly get students onboarded

# EE504 - Software Defined Radio Lab

- Students started with exposure to ADALM-PLUTO and MATLAB from another TA, more in line with lecture topics
- Transitioned to learning GNU Radio in lab ~Week 6, giving 5 weeks to learn GNU Radio and produce a unique final project

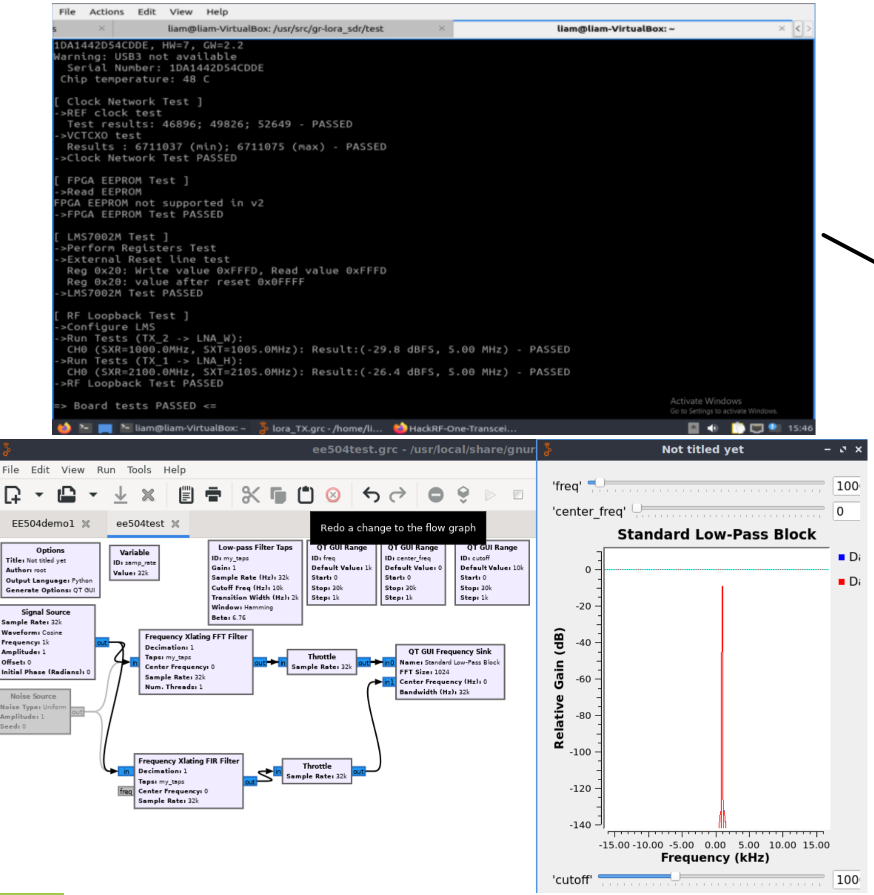
GNURadio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

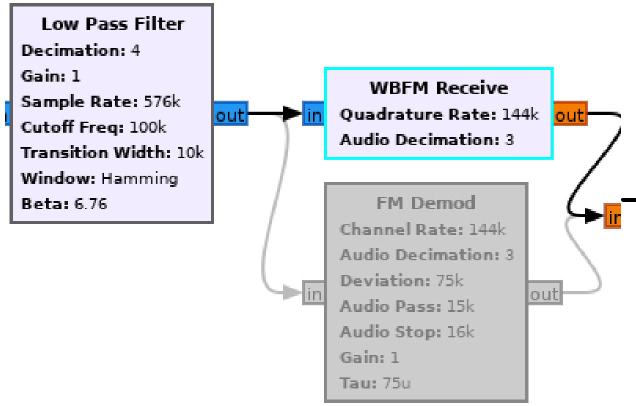
MATLAB®

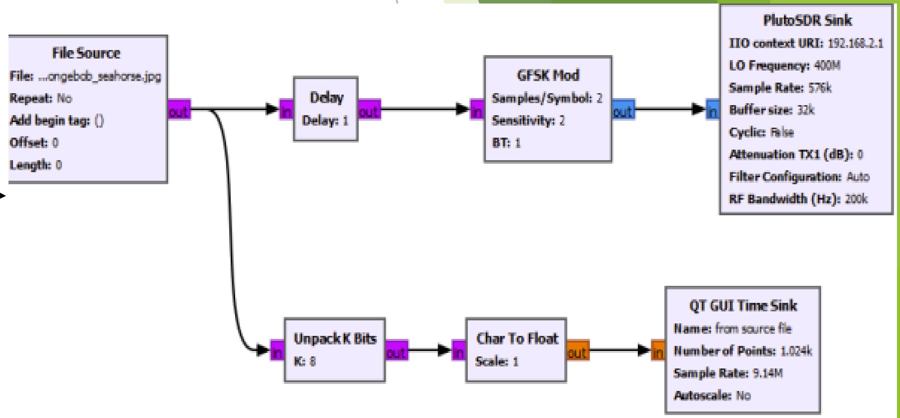


# Approach to Introducing GNU Radio to EE504 Lab

Get everyone's hardware working.
LimeSDR needs to pass its self-test



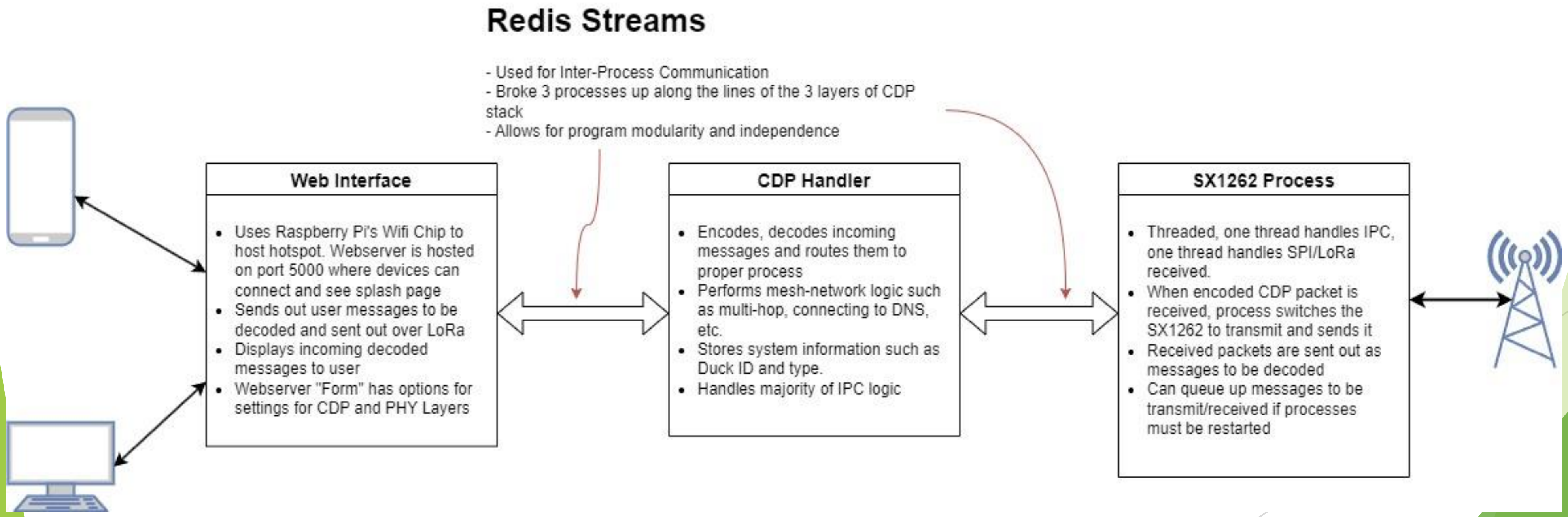

Have students work up to simple Modulators, Demodulators

Assist students with more advanced topics and their final projects

Start with very basic simulated flowgraphs to introduce concepts and blocks

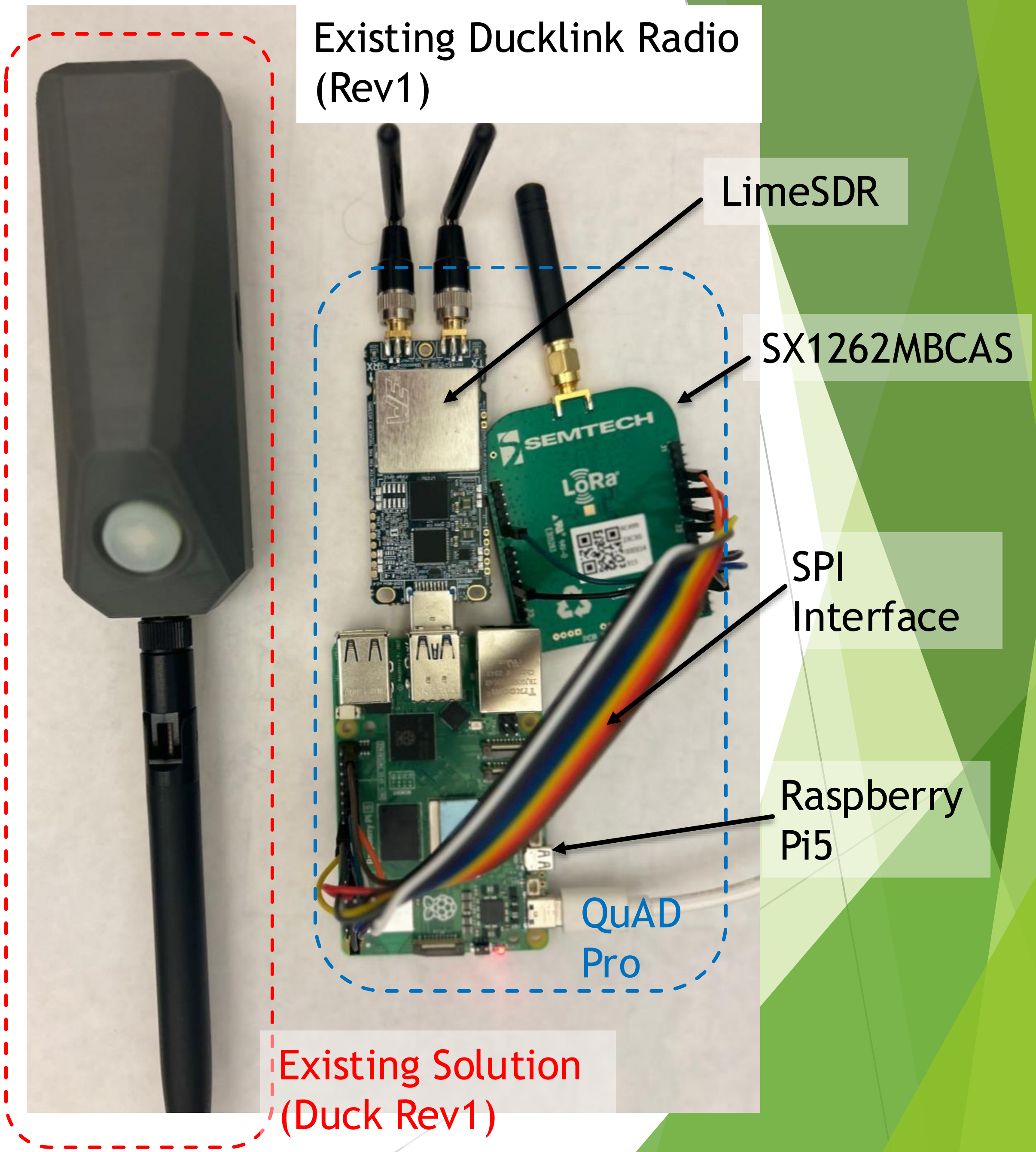
# The QuAD Pro Prototype

- 3 separate processes run at the same time, each loosely representing one part of the "CDP Stack"
- Uses Redis, a simple IPC, to communicate between processes using message streams. Messages are queued up for tasks to complete to handle multiple messages
- Using separate processes with a defined IPC format allows us to in the future replace the "PHY Process" with a proces that can select and run flowgraphs



# GNU Radio for QuAD Pro Summer Research

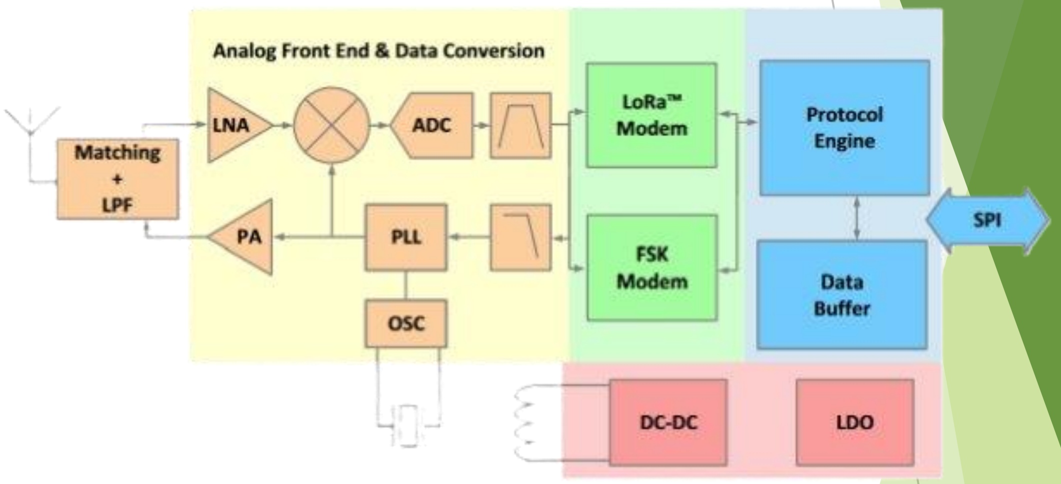
- Protoype "QuAD Pro" Software developed and in debugging phase: (QuAD pro github)
  - SX1262 Driver, Hotspot + Web-server and CDP Packet code all written by students
  - Efforts in the 10-week program were to make a basic system others can debug, improve, and eventually use with SDR applications
- Preliminary "Robustness" and "Reliability" tests comparing SDR to old and new Semtech SX1276, SX1262 Transceivers.
- Improvement of LoRa and FSK Flowgraphs to minimize weaknesses and open up potential interfacing to other processes



# SX1262 LoRa and FSK Transceiver

- Dual modem transceiver (LoRa/FSK).
- Frequency range: **150 - 960 MHz**.
- Data rate: FSK **300kbps**, LoRa **62.5kbps**
- Max RF Output Power **22 dBm**
- Low power consumption

- **Objective:** Compare the reliability and performance of SX1262 and LimeSDR Mini 2.0 using FSK/LoRa modulation across configurable parameters.




SX1262 Block Diagram
*Source: Semtech Corporation, SX1262 Datasheet, [Datasheet]*

# LoRa and FSK Through GNU Radio - Can SDR compete with the SX1262


- **Why Compare FSK & LoRa on SDR vs. SX1262?**
  - **Application:** Both have been proposed to be used on QuAD Pro
  - **Flexibility:** Can SDR quickly change LoRA/FSK parameters like the SX1262?
  - **Dedicated Performance**: SX1262 is hardware-optimized, how does SDR stack up?
  - **Real-World vs. Emulated Results**: Practical insights into differences in reliability and use cases
- **Expectations:**
  - **SDR**: Adaptable, great for testing and prototyping but more likely to be susceptible to noise or demodulate incorrectly.
  - **SX1262**: Tuned for efficiency and real-world applications, much more power efficient than SDR. Likely to take slightly longer to set up even with given code than SDR

# SDR Flowgraph vs New Duck Transceiver vs Old Duck Transceiver


- **Bit Error Rate and Packet Drop frequency are our two main points to compare**
- **Each hardware set is going to be slightly different**

**SX1276: (LoRa only)**

- Onboard Rev1 Duck, meaning code is run on a T-Beam board
- Outdated board doesn't work with newest ESP Drivers -> RadioLib Doesn't work -> FSK not possible. Used other LoRa library

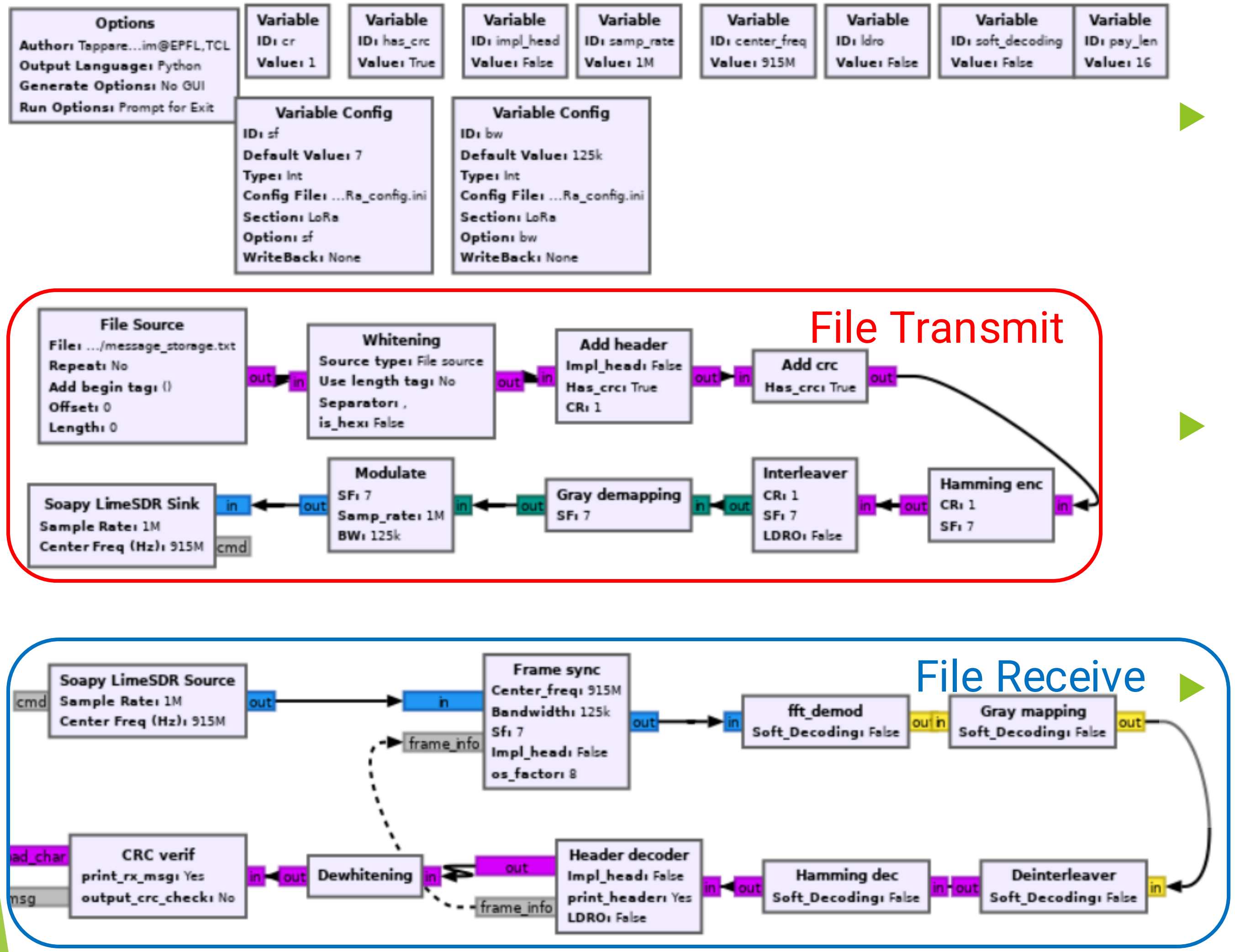
**SX1262:**

- Uses GPIO pins of Raspberry Pi 5, WiringPi to control
- Also uses RadioLib with custom HAL code. Can compare FSK

**SDR:**

- USB Port of Raspberry Pi 5, requires drivers on DragonOS image (or proper version of LimeSuite)
- Flowgraphs can differ from true PHY, ability to add functionality not possible with Semtech Transceiver

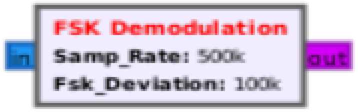
# LoRa on GNU Radio – gr-lora_sdr




- All true LoRa blocks were built by tapparelj: https://github.com/tapparelj/gr-lora_sdr. Probably wasn't going to be able to do this myself with <2 months of GNU Radio experience
- Originally built flowgraphs for Senior Project to encode+decode CDP packets with custom blocks. Limited success due to packet corruption and perhaps SX1276 timing
- Added blocks for more advanced file read/write options. Added config file to be used for semi-automated testing.

# FSK Modulator/Demodulator

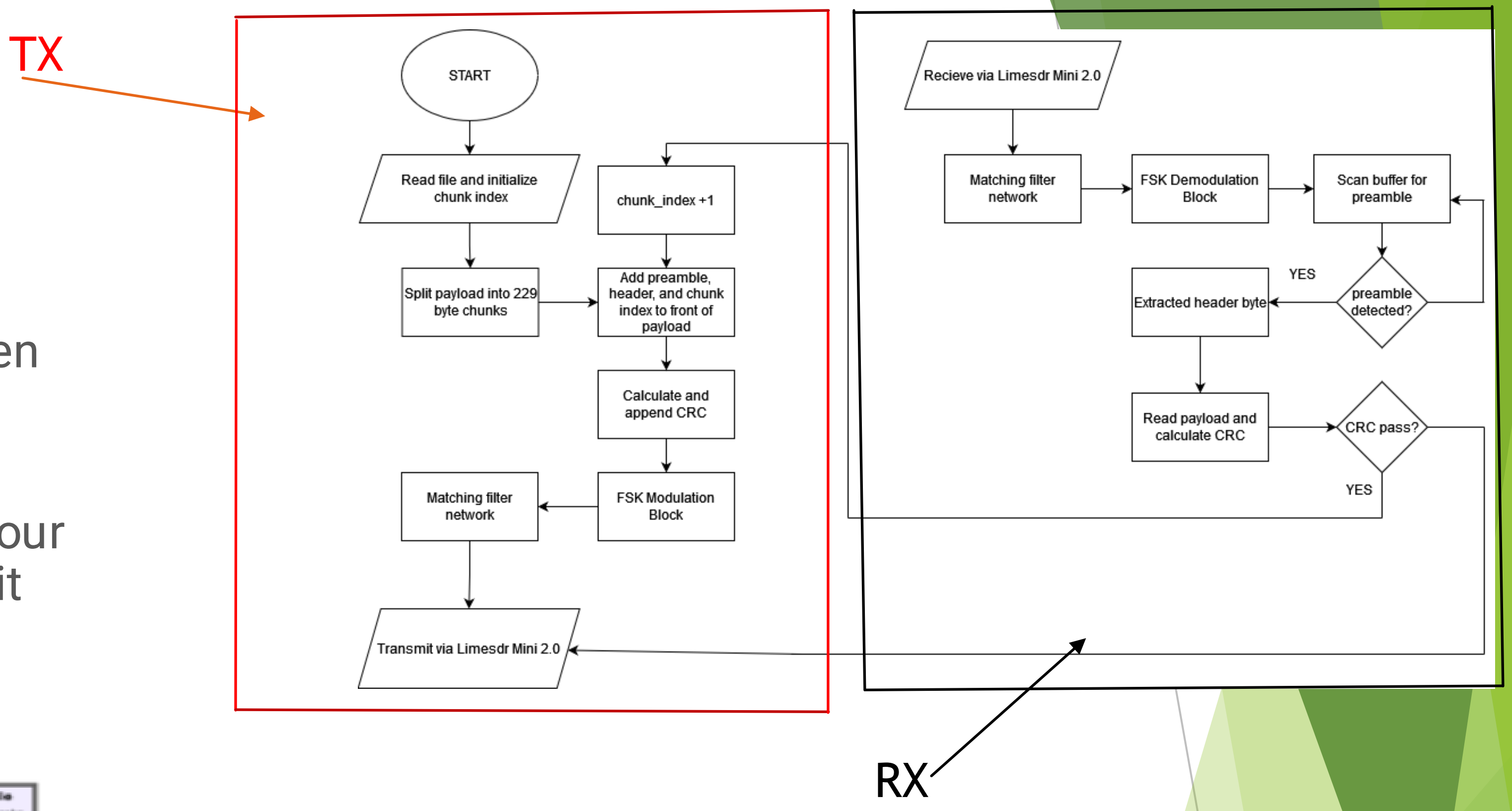

```python
    self.samp_rate = samp_rate
    self.fsk_deviation = fsk_deviation
    self.center_freq = 0
    self.phase_inc0 = 2.0 * np.pi * (self.center_freq - self.fsk_deviation) / self.samp_rate
    self.phase_inc1 = 2.0 * np.pi * (self.center_freq + self.fsk_deviation) / self.samp_rate
    self.phase = 0

def work(self, input_items, output_items):
    out = output_items[0]
    in0 = input_items[0]

    # Process the input data
    if len(in0) > 0:  # Check for data
        for i in range(len(in0)):
            if in0[i] == 0:
                self.phase += self.phase_inc0
            else:
                self.phase += self.phase_inc1
            out[i] = np.exp(1j * self.phase)
            if self.phase > 2.0 * np.pi:
                self.phase -= 2.0 * np.pi
    else:
        print("Received empty input data")  # no data

    return len(out)
```

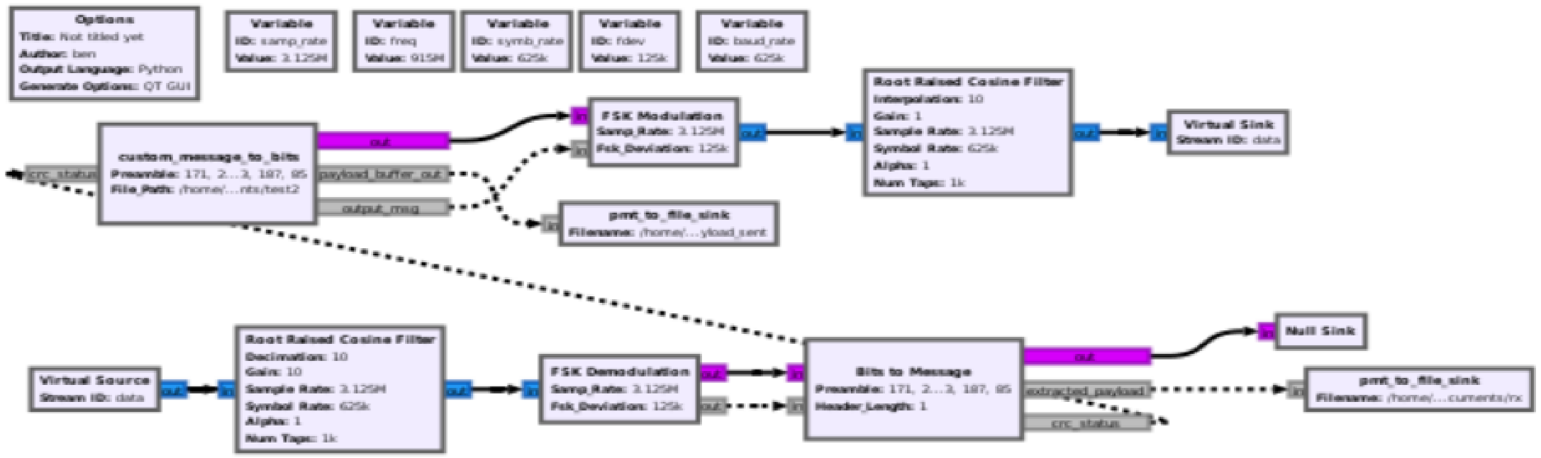



```python
class fsk_demod(gr.sync_block):
    def __init__(self, samp_rate=1e6, fsk_deviation=500e3):
        gr.sync_block.__init__(
            self,
            name='FSK Demodulation',
            in_sig=[np.complex64],
            out_sig=[np.int8]
        )
        self.samp_rate = samp_rate
        self.fsk_deviation = fsk_deviation

    def work(self, input_items, output_items):
        in0 = input_items[0]
        out = output_items[0]

        for i in range(1, len(in0)):
            phase_diff = np.angle(in0[i] * np.conj(in0[i-1]))
            out[i] = 1 if phase_diff > 0 else 0
        return len(output_items[0])
```

# FSK on Gnu Radio

- Implemented a packet based system establishing a link between 2 LimeSDR mini-2.0s using FSK modulation.
- A CRC is calculated and used as our basic check if a packet is transmit and received correctly

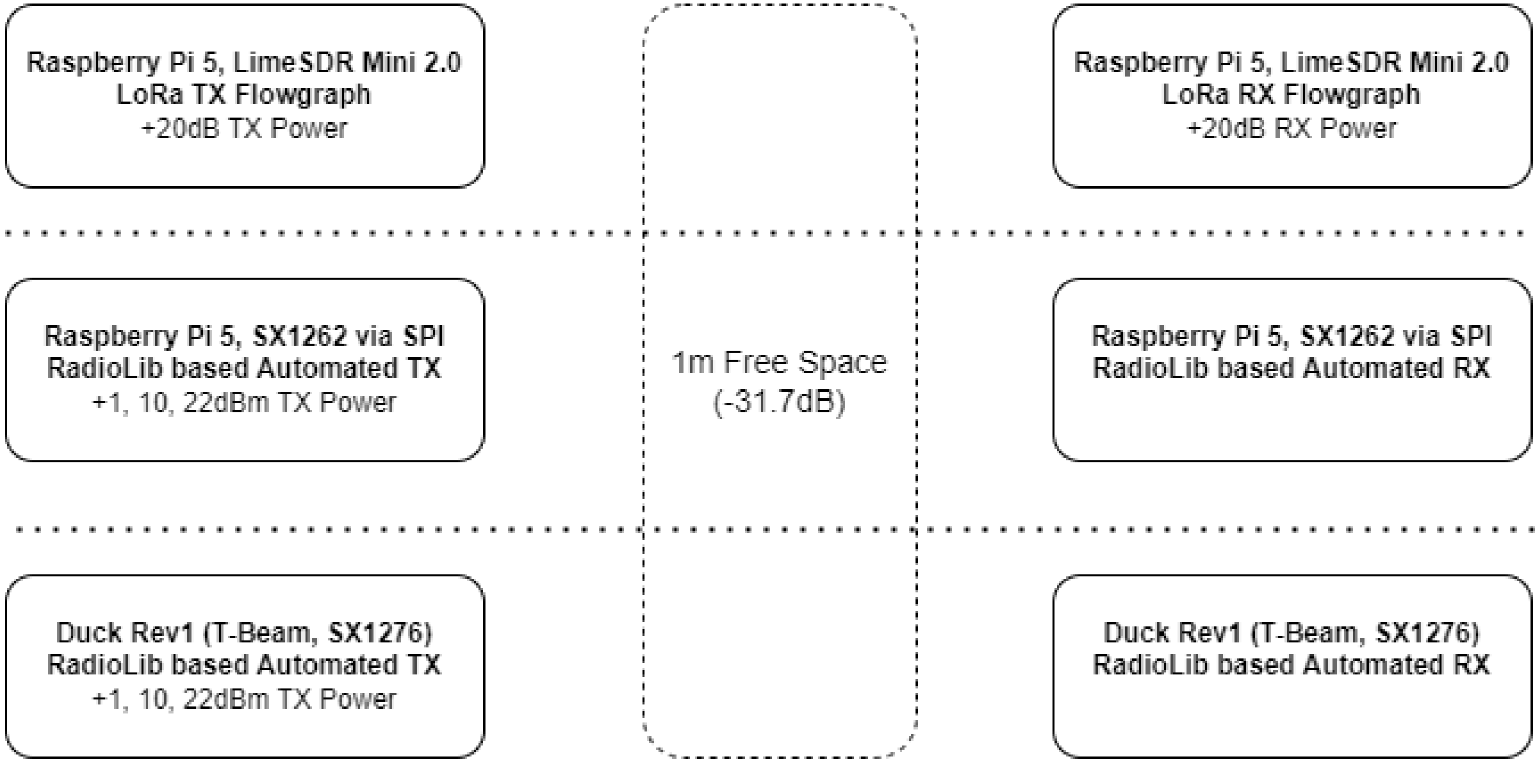


# Testing setup LoRa

**Parameters Tested**

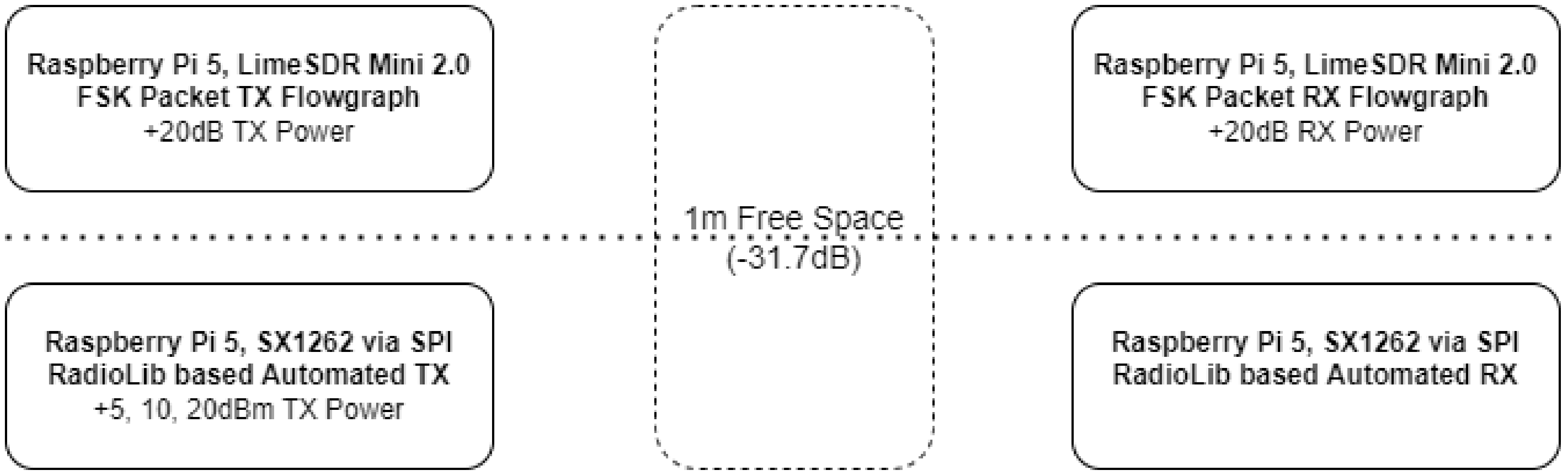
SF: 7, 9, 11
BW: 62.5, 125, 250, 500kHz
CR: 5, 8



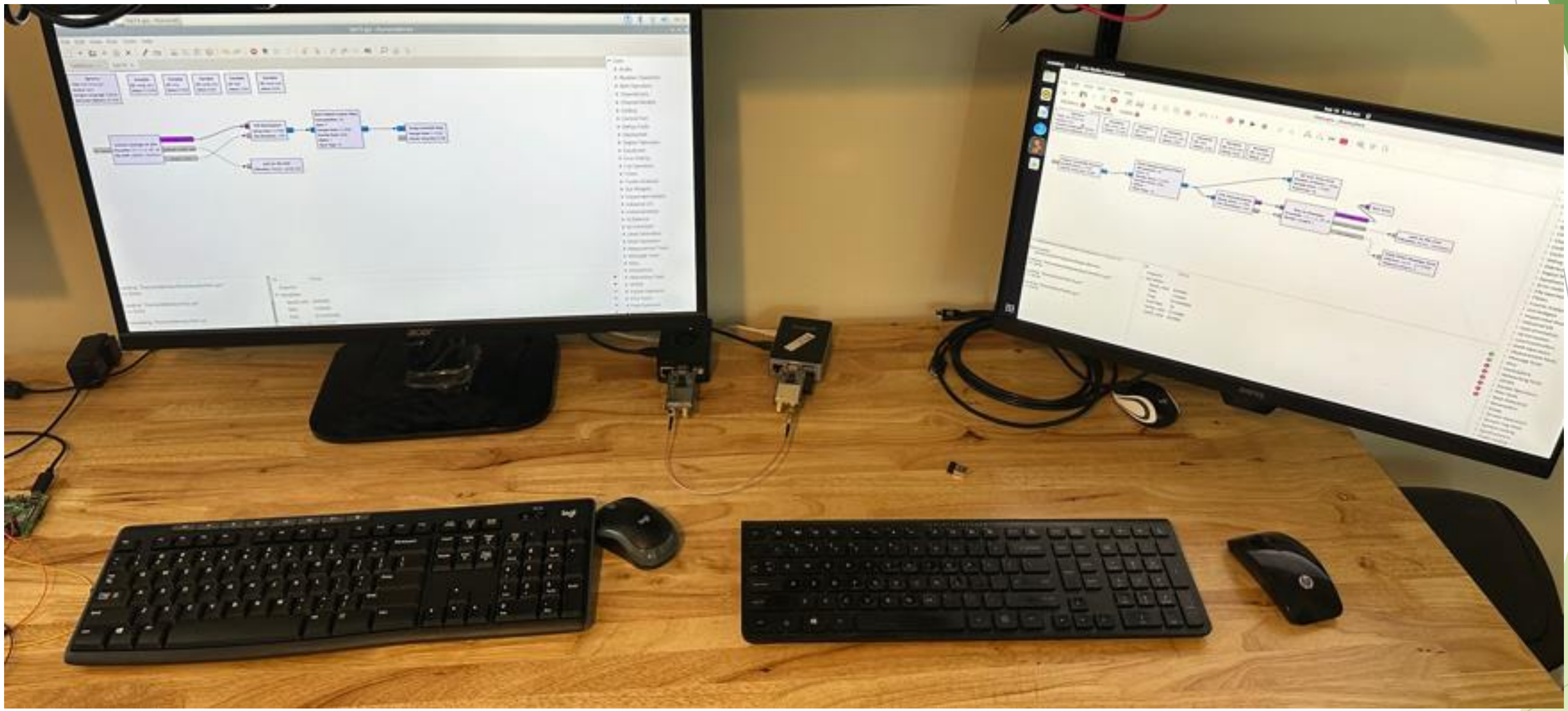
# Testing setup FSK

**Parameters Tested**

BitRate: 4.8, 9.6, 19.2, 76.8, 153.6kbps
Freq. Deviation: 10, 25, 50, 75, 100kHz



# Testing setup



# LoRa Performance Comparisons

**Notes:**

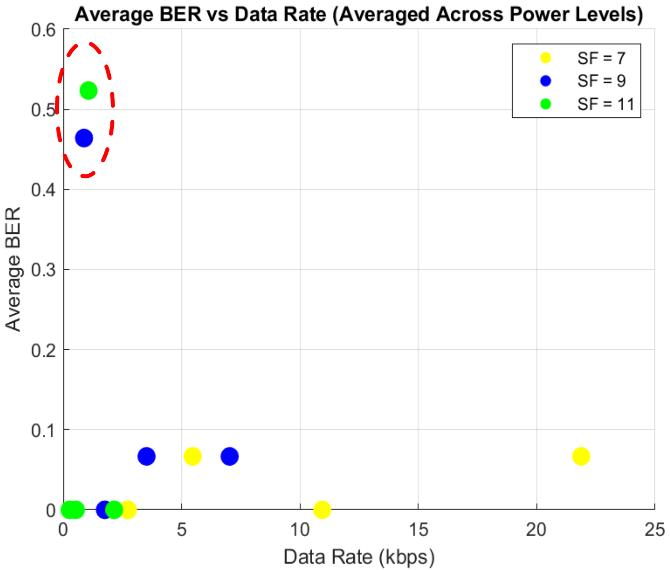
- Experienced abnormally high error rates/packet losses at extremely low data rates
- LoRa OOT module is restricted. "Requires too many taps" for extremely low data rate demodulation, I.E. these could not be compared with dedicated LoRa transceivers

**SX1276/SX1262:**

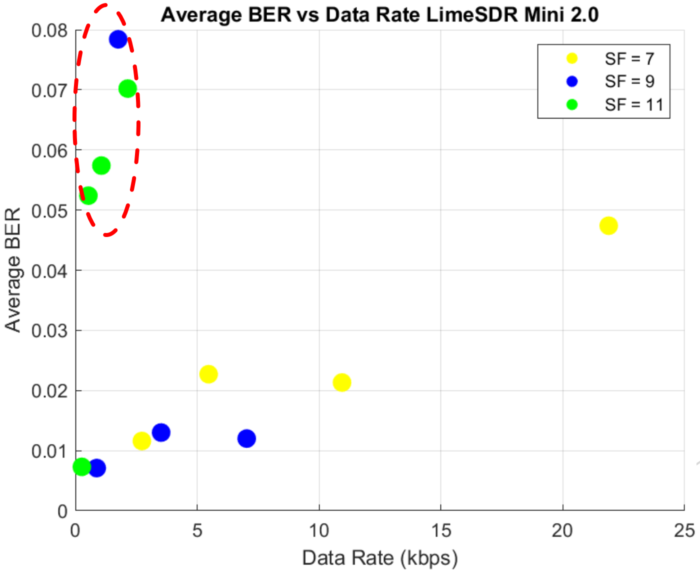
- Similar "low data rate problem" on both modules
- When packets were properly received, the SX1262 performed better in bit errors. More packets dropped led to the seemingly worse performance

**Lime Mini 2:**

- Consistent performance, much more susceptible to noise and gain settings
- Bit errors are to be expected unless settings are "perfect"




SX1262




Lime Mini 2

# FSK Performance Comparisons

**Performance at Lower Bit Rates (4.4 kbps-76kbps)**

•**SX1262:**

- **PER:** 0%
- **BER:** 0%
- **Reliability:** Excellent, error-free communication at low data rates.

•**GNU Radio LimeSDR Mini 2.0:**

- **PER:** 2.94%
- **BER:** 0.13%
- **Reliability:** Consistent performance, moderate error rate.

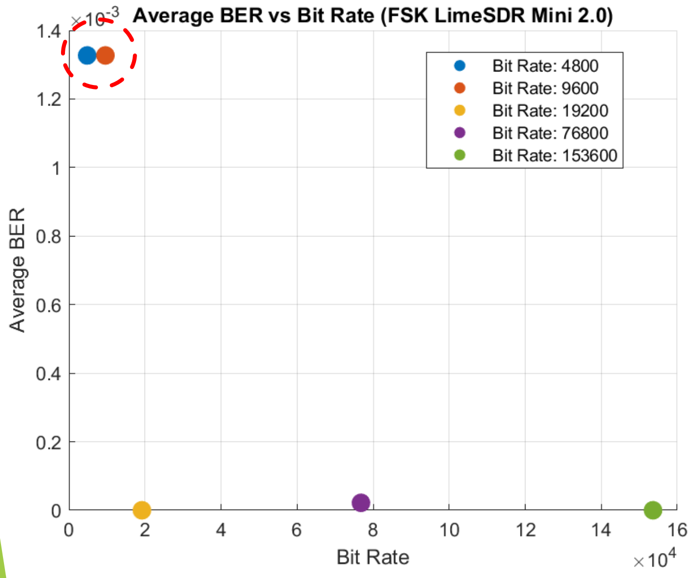
**Performance at Higher Bit Rates (153.6 kbps+)**

•**SX1262:**

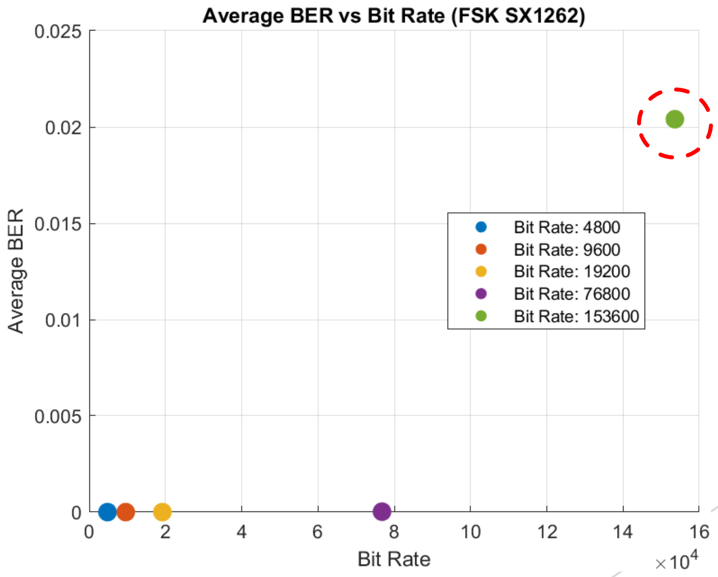
- **PER:** Increases up to 4% at 20 dBm at Fdev of 10kHz.
- **BER:** Increases up to 9.84%.
- **Challenge:** Significant errors, less reliable at high bit rates.

•**GNU Radio LimeSDR Mini 2.0:**

- **PER:** Consistent 2.94% across all configurations.
- **BER:** Typically, 0.00% to 0.13%.
- **Advantage:** Maintains steady performance even at higher bit rates.




SX1262




Lime Mini 2

# Summarizing Results

Code used for all tests and results can be found in "tests" folder on our GitHub: https://github.com/limccart7/GRCon-Project

**"Old" 1276:**

- Could not test FSK because of older hardware compatibility issues
- Not as frequency/parameter agile as newer chip

**SX1262:**

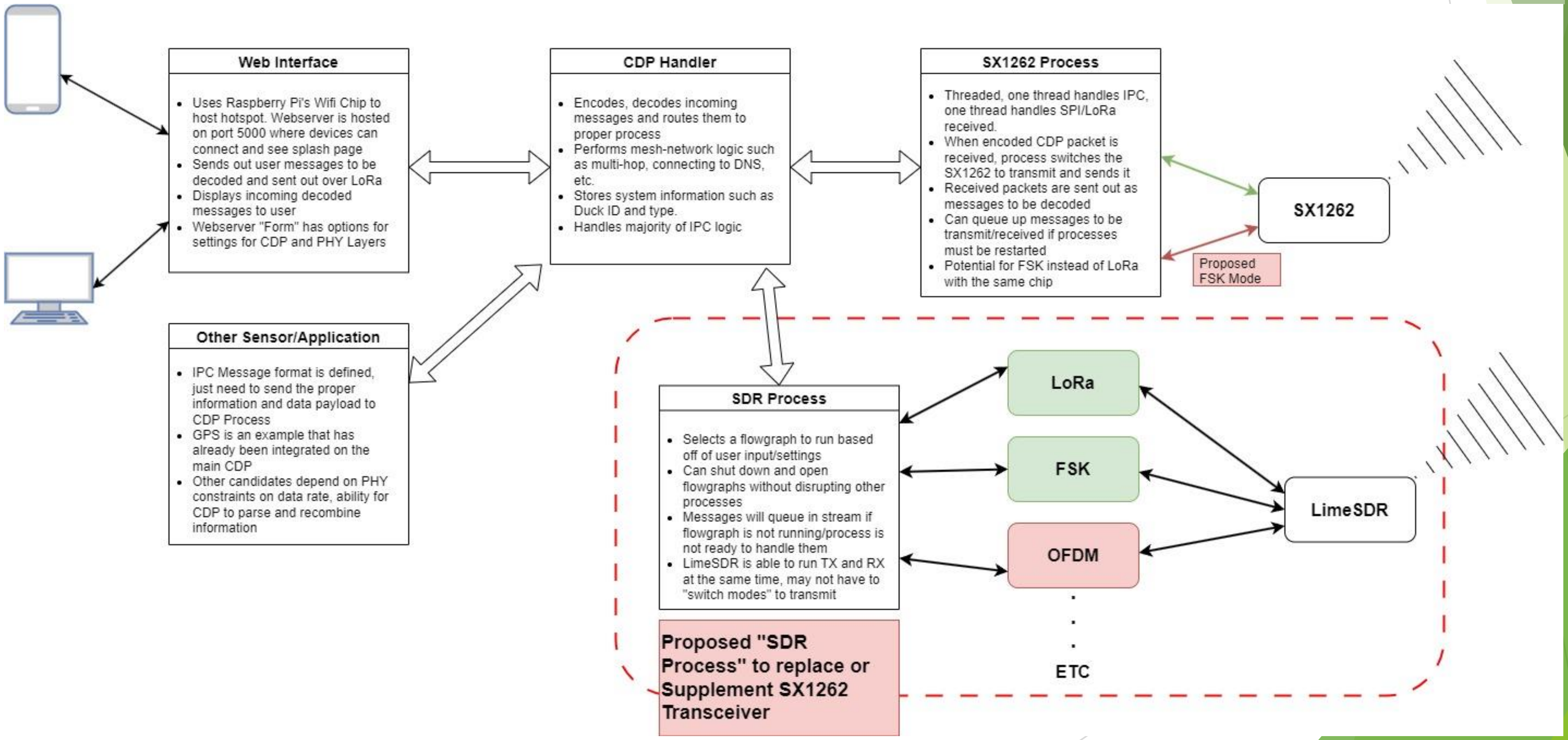
- Best performance in terms of BER and packets dropped
- Using Raspberry Pi GPIO pins made setup, interacting with the chip slightly more difficult

**Lime Mini 2:**

- Not ideal for "high end" situations, higher BER and much more likely to drop packets
- Flowgraphs give us the ability to retransmit on CRC check failure, other ways of mitigating shortcomings

# Future Work - A Frequency and Modulation Agile Transceiver for Raspberry Pi/QuAD Pro

- Use "QuAD Pro" Prototype's Inter-Process-Communication to integrate SDR Flowgraphs with ClusterDuck Protocol
- Develop a program that configures the parameters of flowgraphs and starts them. Eventual goal is to have a system that can quickly change its modulation and parameters based off user input



# Questions

# Links, Further Reading

- Our GitHub, again: https://github.com/limccart7/GRCon-Project
- Main QuAD Pro Prototype Summer Research: https://github.com/limccart7/QuAD-Pro-Prototype

References:

- https://clusterduckprotocol.org
- https://github.com/tapparelj/gr-lora_sdr
- https://cemaxecuter.com (DragonOS)