Cross-Platform Comms Manual

Making USRP (Alice) talk to Pluto (Bob)

Communications Lab

November 29, 2025

1 The Big Picture

We are building a security system where Alice (USRP B210) sends a secret message using two antennas, and Bob (Adalm Pluto) tries to receive it using one antenna.

The Main Challenge

The USRP and the Pluto are like two people who speak different dialects and hence different way of handling things. Just to list :

- **USRP**Our Alice has to send and receive data and also do channel processing.
- **Pluto** Bob has to just receive and decode the data.

To make them understand each other and make good use of the hardware, we must force them to speak at exactly the same speed (3 MHz).

2 Step 1: The Hardware Setup

We use "Objects" in MATLAB to control the hardware. Here is how we configured them in `TX_dual_pluto.m` and `Hardware_RX_Bob.m`.

Alice: The Transmitter (USRP B210)

Alice is the "Master" of the transmission. She sends two signals at once.

- **Device:** USRP B210
- **Goal:** Send separate streams on Antenna 1 and Antenna 2.
- **The Trick:** We use **Interpolation** to slow down the internal clock to a speed the Pluto can handle.

Key Settings:

1. `MasterClockRate = 30 MHz` (Internal Heartbeat)
2. `InterpolationFactor = 10` (Slow down by 10x)
3. **Resulting Speed:** 30 MHz/10 = **3** MHz
4. `ChannelMapping = [1 2]` (Use both antennas)

Bob: The Receiver (Adalm Pluto)

Bob is the listener. He uses a cheaper radio, so we have to tell him exactly what to listen for.

- **Device:** Adalm Pluto (PlutoSDR)
- **Goal:** Listen to Alice and separate her two signals.
- **The Trick:** We force the Pluto's "Baseband Rate" to match Alice exactly.

Key Settings:

1. `BasebandSampleRate = 3 MHz` (Must match Alice!)
2. `CenterFrequency = 800 MHz` (Must match Alice)
3. `Gain = 60 dB` (High gain because Pluto is less sensitive)

3 Step 2: How Bob Finds the Message

The air is full of noise. Bob needs a way to know "Hey, a message is starting now!"

The Packet Detection

Alice sends a special pattern called a Preamble at the start of every message. This acts like a message to capture the data.

Schmidl-Cox Algorithm (Simple Explanation)

1. Alice sends a short pattern: `[A, A, A, A]`.
2. Bob constantly compares the signal he just received with the signal he received a split second ago.
3. If **Current Signal** looks exactly like **Past Signal**, Bob knows it's the repeating preamble.
4. **Math:** We calculate a score $M(n)$. When $M(n) > 0.6$, we trigger the recording.

4 Step 3: Separating the Two Antennas

This is the most critical part of Physical Layer Security. Bob has only **one ear** (antenna), but Alice is speaking with **two mouths** (Tx1 and Tx2). How does he know who said what?

The "Pilot" Solution

We use **Orthogonal Pilots**.Please refer to the codes in the same folder for more detail.

gray!20 **Frequency Key**	**Tx1 Action**	**Tx2 Action**
Key #16	**Signal1**	Silent(Null)
Key #18	Silent	**Signal2**
Key #24	**Signal1**	Silent
Key #26	Silent	**Signal2**

1. Bob listens to Key #16. He hears a sound. Since he knows Tx2 is silent there, **that sound must describe Tx1's channel.**
2. Bob listens to Key #18. He hears a sound. Since he knows Tx1 is silent there, **that sound must describe Tx2's channel.**

5 Summary: The Code Logic

Here is the flow of `OFDM_RX_Bob.m` in plain English:

The Receiver Logic Chain

1. **Wait** for the trigger signal ($M_n > 0.6$).
2. **Cut** the signal into a "Frame" (480 samples).
3. **Correct** the frequency (Pluto and USRP clocks are slightly different, so we rotate the signal back).
4. **Extract** the Pilot Keys (indices 16, 18, 24, etc.).
5. **Calculate** Channel 1 from the Tx1 keys.
6. **Calculate** Channel 2 from the Tx2 keys.
7. **Decode** the secret message using these channel estimates.