**ITIS-6010-002-Wireless Security**

**HackRF Meets Drone**

**Team Members:**

Ruchira Pokhriyal, Trevon Williams.  
  
**Project Overview:**

Inspired by Mossmann’s HackRF and Lego car project, our term project: **HackRF Meets Drone** attempts to utilized a custom quadcopter with a generic firmware and receiver programming files, and using the **HackRF**, capture certain control signals from the provided controller. We then attempted to replay the captured signal to the quadcopter and tried to control its movements with the replayed signal.

**How is this related to Wireless-Security:**

🡪The core implementation of this project revolves around a basic attack technique in wireless networks called: ***Replay attack.*** By using HackRF and GNU Radio together, we could capture and reply signals to the drone to control the its movements, which could be beneficial in a lot of ways.

🡪Wireless Operation and remote control of a device such as a drone, allows the operator to move more freely, enabling him/her to gain the best viewpoint of the work being performed and saves man-power.

🡪 Since the operator does not have to be in direct contact of the drone, they can position themselves anywhere safe in an undesirable environment, while the drone is taking all the risks.

🡪The idea of remote controlling a quadcopter is actively being used in wireless technological advancements such as: Crop Spraying, Construction Site Monitoring, Security Surveillance, Search and Rescue and even Photography.

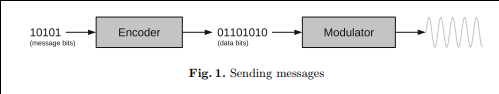
**Findings:**

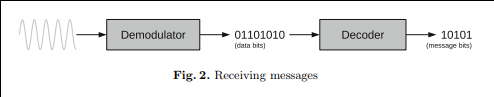
🡪We began by doing research on the libraries used by the controller (Taranis Q X7) and the quadcopter (FrSky). The FrSky protocol is proprietary so we had to search in forums to understand how others were progressing with reverse engineering the protocols.  
***🡪FrSky Protocol*** exchanges a certificate between the copter and controller to ensure that there is authenticated communication between the two. This exchange is broadcasted on 5GHz spectrum. After binding, the copter and controller communicate on 2.4GHz spectrum and the copter receives 5GHz for video streaming where applicable.

🡪The OpenTX Libraries were open source and were also incredibly extensive. We found out some information about binding and post binding communication from there.

🡪We also found out that the quadcopter utilizes a popular method called **“Frequency Hopping Spread Spectrum (FHSS)”**, which allows data transmission to be more resistant to interference.

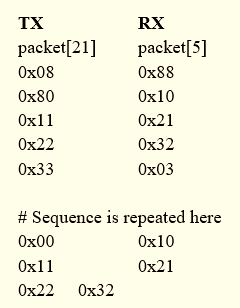
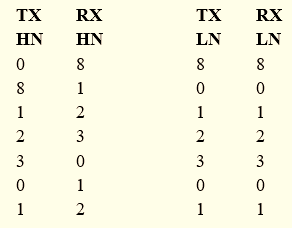
We reference the methods for sending and receiving messages from a diagrammatic representation shown below:





**Handshake:**The handshake between the drone and the quadcopter is handled within the OpenTX implementations. The handshake is a handshake between the tx and rx packets, which each have their own unique headers. The Tx is 21 bytes and the Rx is 5 bytes. The handshake contains packet sequence numbers and 2 control bits [though we are unsure what they are for]. The quadcopter and controller each begin their own transmission sequence with 0x88. The data transferred in the captured handshake is included below.

**TX-RX Telemetry Handshake:**

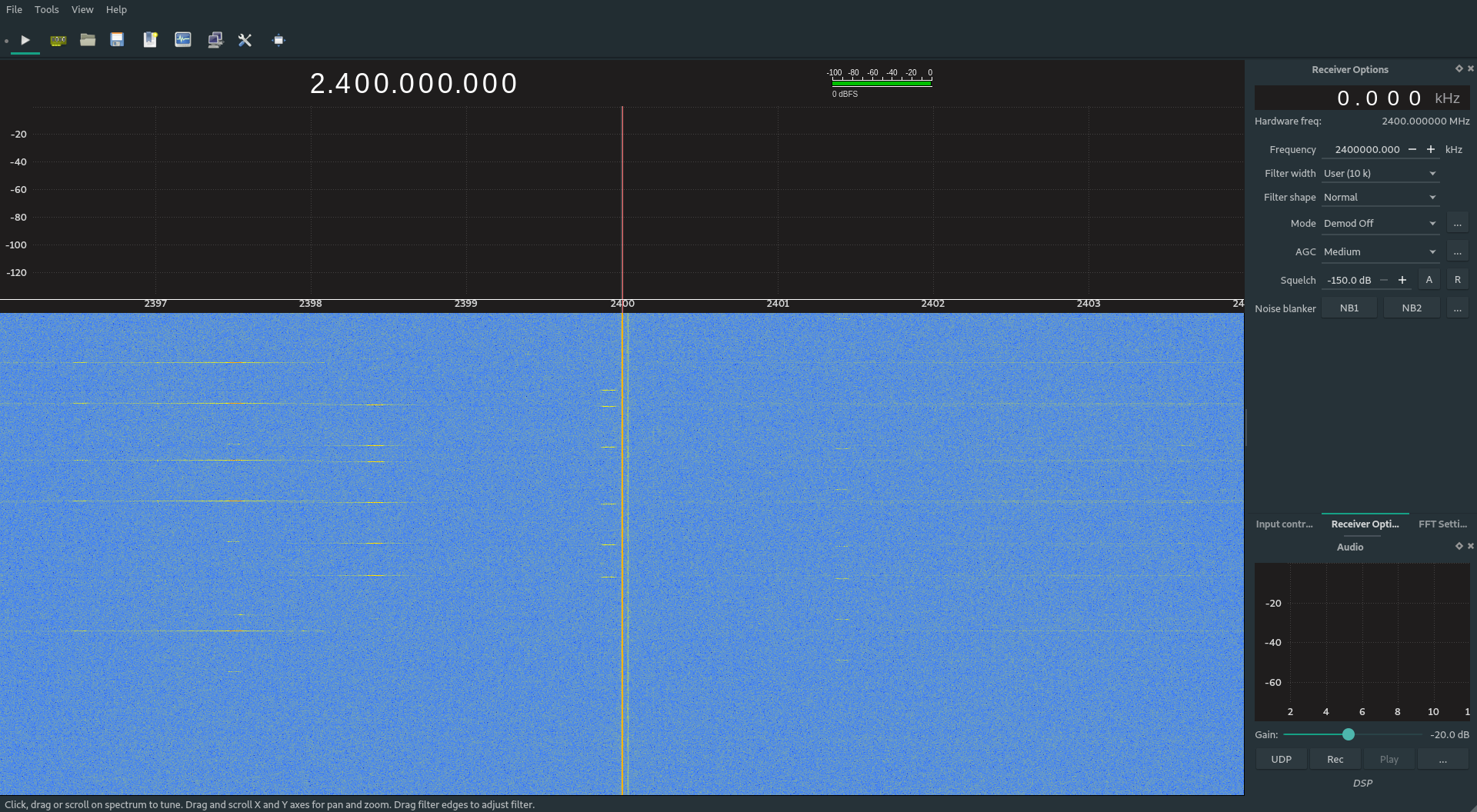


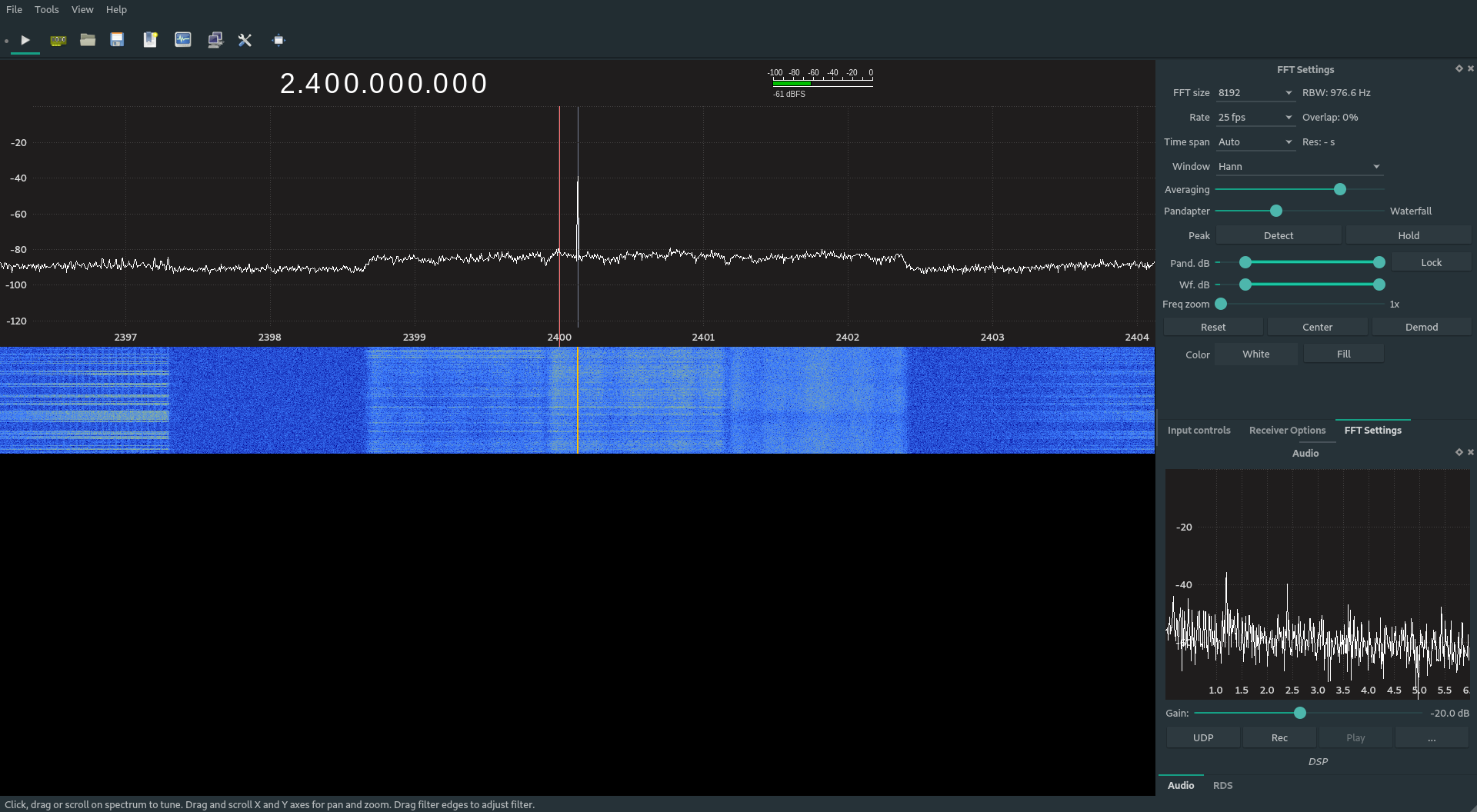
**Security Analysis:**

Vendors are commonly known for relying on the secrecy of their designs to provide some type of level for security for drone protocols. The ***FrSky*** ***protocol*** is no different in this aspect and this has led us to conduct a lot of research on the drone protocol security mechanisms and handshake implementations. One weakness of the drone technology that we found was how the drone telemetry data was broadcasted back to the controller. With enough knowledge on how to reverse engineer the FrSky libraries, one can purchase an ***arduino uno*** and a receiver to be able to capture and piece together received telemetry data.

We later found that, the FrSky protocol used with our drone, implements the D8 communication protocol, which handles the “frequency hopping”, “spectrum spreading” and the binding sequence where a key is shared. This communication protocol utilizes the 2.4GHz range to communicate and you can see some of the signals captured in our Working with HackRF sections. Within our research, we also found that the D8 protocol utilizes packet-based transmission and this was also confirmed in our HackRF findings (through assumptions).

**Working with HackRF & Technical Difficulties Faced**:

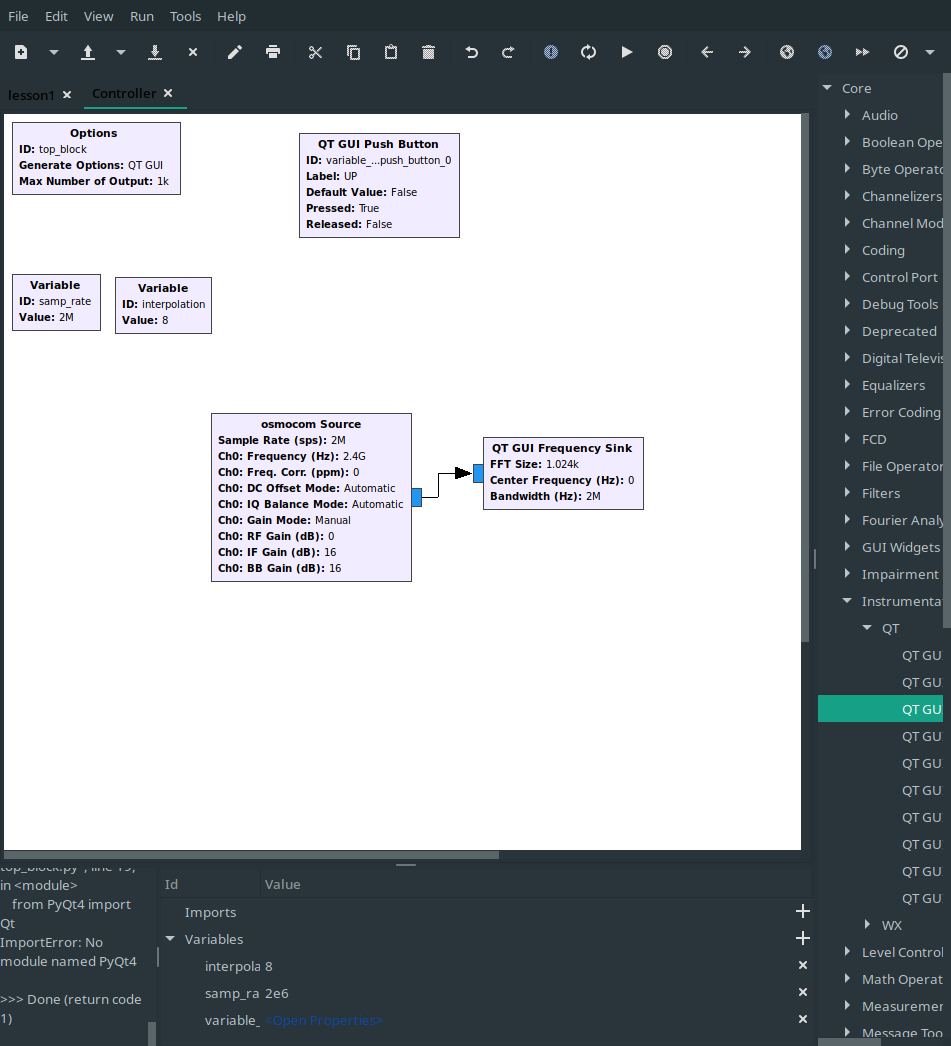
**Software Setup**:We have set up the HackRF using Ubuntu OS. We included the most recent release packages such as ***libhackrf*** and ***gr-osmosdr,****,* and use them to install those packages in addition to GNU Radio.

 *↑* ***The first image above shows the interference***

**Viewing Transmission Signal GQRX**

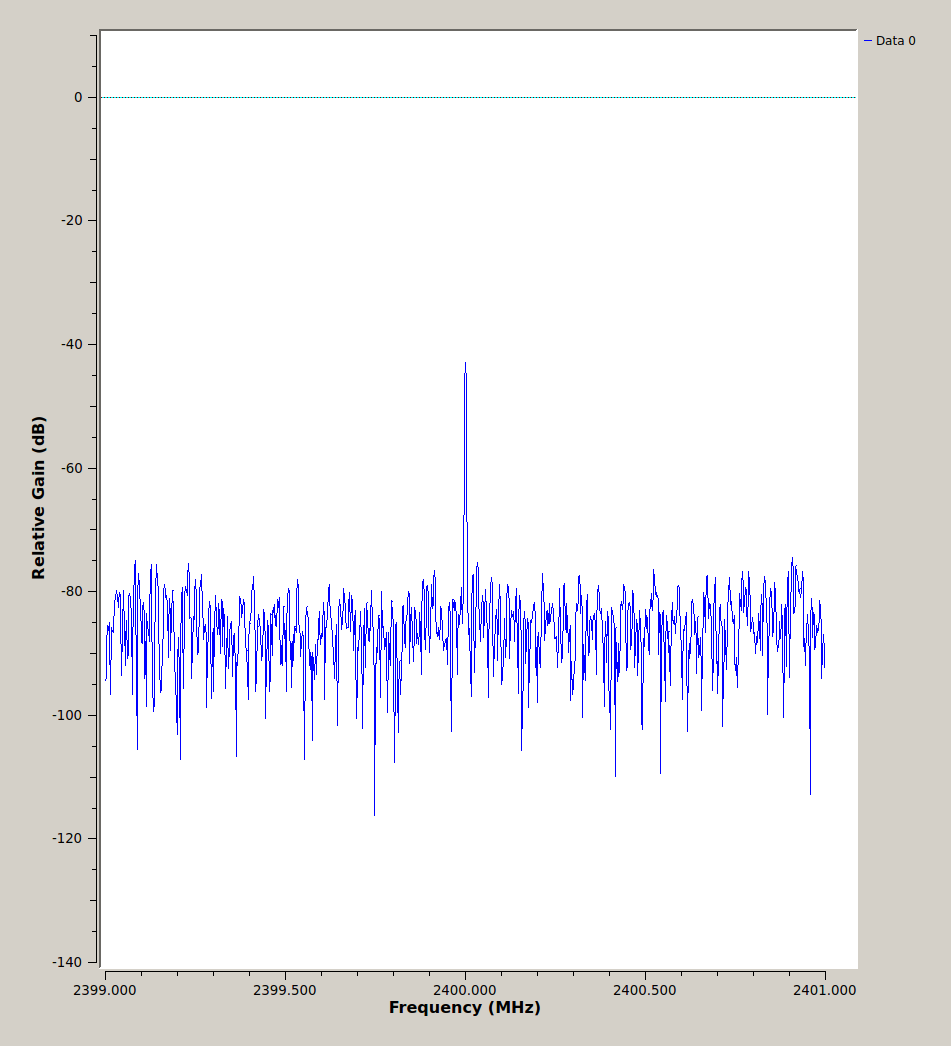
Here we are viewing the control signal being sent on the 2.4Ghz channel and other telemetry data being sent less powerful wave lengths. We are unsure which is the sequence of command grouping, but we can analyze the signal in GNU radio to get a better idea of how the controller and quadcopter are communicating.

We had some problems being able to decipher the frequency the controller was communicating with the quadcopter on, because of signal interference with other devices. We included a screenshot of the interference so that you can see the difference***.***

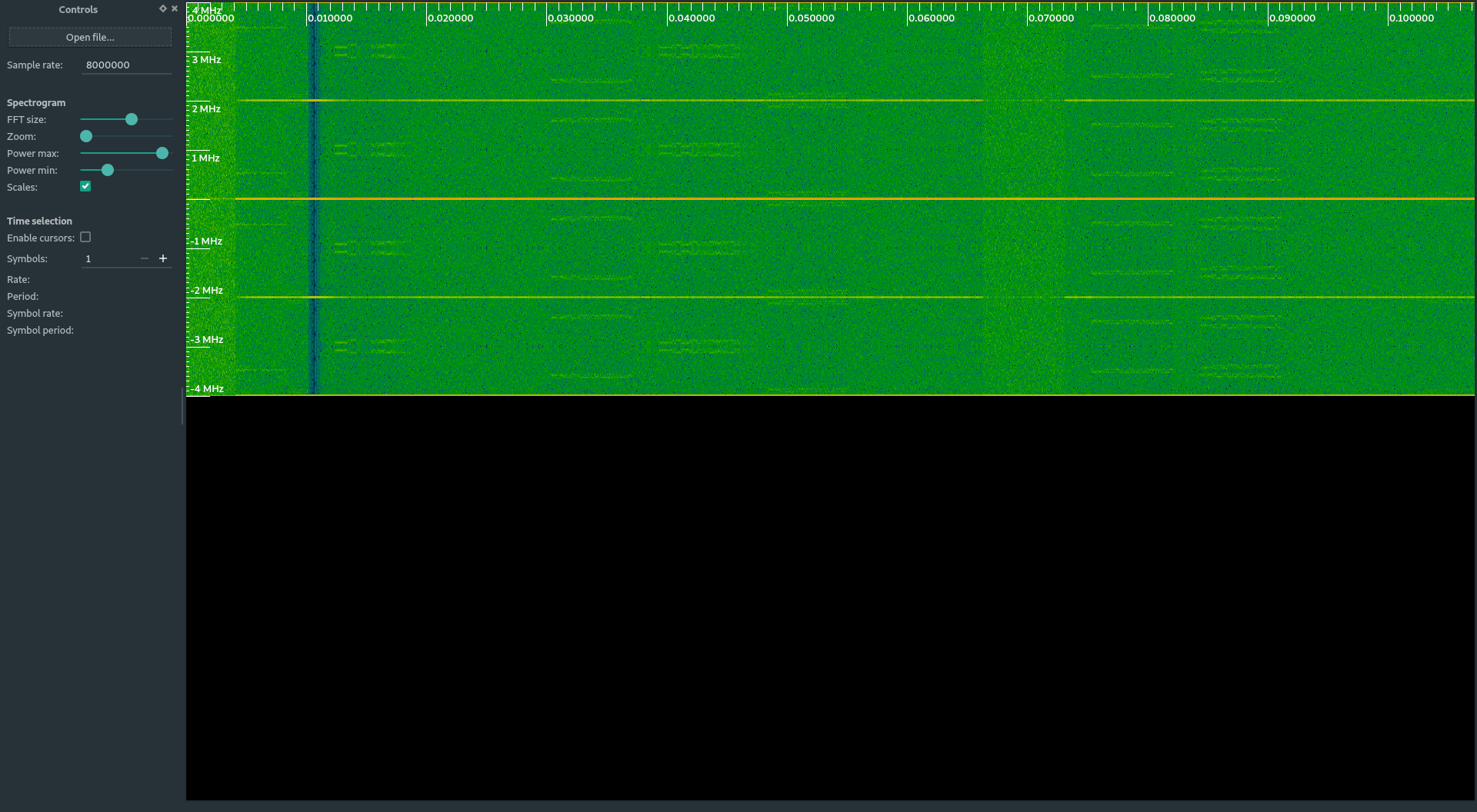
**Capturing the Signal**

After finding the frequency of the broadcasted signal, we then used GNU radio. For this, we used some of our HackRF knowledge along with several online guides to start building a GRC file to first display the broadcasting signals and then capture them.

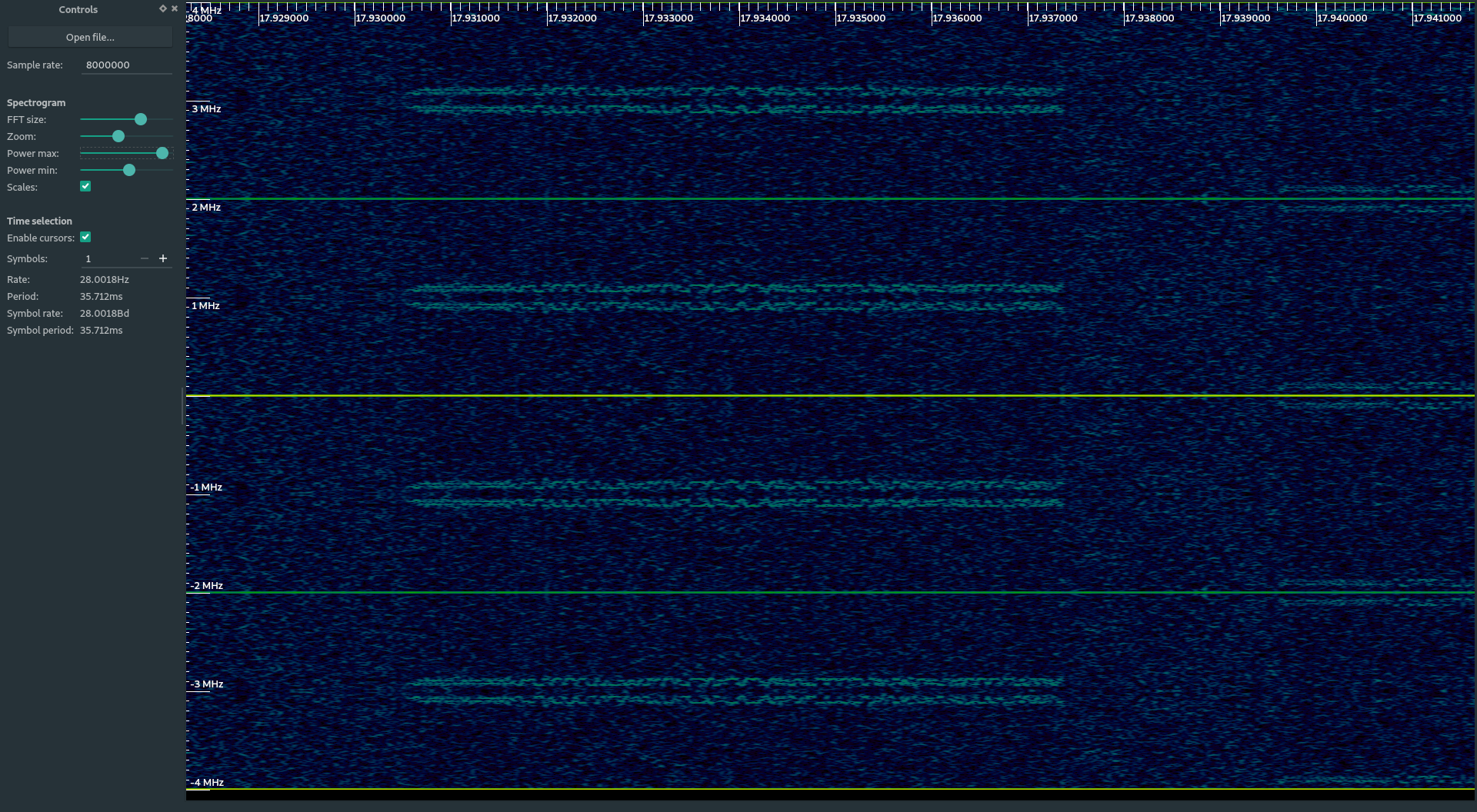
**Controlling Drone through HackRF**:  
  
We were unable to conduct a replay attack on the wireless drone due to unforeseen issues which arose from signal processing of the captured signal.   
We had this problem because of the amount of noise that was already being broadcasted on the 2.4GHz spectrum in the areas we were using hackRF to capture the signal.   
  
The image below, shows all of the interference we picked up when trying to capture and analyze the captured signal:



Using ***inspectrum*** (the program name) we could view the captured bytes of the quadcopter communication to further analyze some of the interference and telemetry data. In this program we can slice which blocks of the signals we wish to replay and include them in a new ***.grc*** file that will execute the replay attack.

****

In the image below, we can see the bytes of the handshake and controlling sequence:



**Future Work and Potential Impacts if Succeeded:**

🡪If we had more time to conduct this experiment, we would focus on reverse engineering the different control signals broadcasted on the different frequencies which is shown in one of the above images.  
🡪We would also look into using the extended knowledge obtained from this endeavor to see if we could spoof telemetry data being set to and from the drone see as we can capture these signals and they are in no way encrypted.

🡪Another thing which we would research on is how to recreate the communication sequences between the controller and the quadcopter. We learned early on that the controller was a more powerful software defined radio.   
🡪This means that with enough programming and signal knowledge, we would be able to create synthetic signals that can communicate with the drone to trick it into thinking that its communicating with the controller.

**Work Distribution:**

* **Software Setup:** Sneha Rangari, Shail Patel
* **Project Documentation:** Ruchira Pokhriyal, Sneha Rangari
* **Protocol Research and Reverse Engineering:** Trevon Williams
* **Capturing FM Radio Signals using HackRF and GNU Radio:** Shail Patel, Ruchira Pokhriyal
* **Video Making:** Introduction by Sneha, Demo by Trevon, Explanation of Initial Signal Capturing and Future Scope by Ruchira and Shail.

*Link to YouTube Video:*