NAME: - ROHIT SATISH PATIL | **M.Tech.** CMN
YEAR: - 1^{st} Year | **SEM: -** 2^{nd} Sem
SUBJECT: - CS-II Lab | **EXPERIMENT 1**

AIM: - Perform Amplitude Shift Keying (ASK) modulation and demodulation using GNU Radio.

SOFTWARE: - GNU radio.

THEORY: -

GNU Radio:

GNU Radio is an open-source software development toolkit that includes a comprehensive collection of signal processing blocks for the creation and implementation of software-defined radios (SDRs) and other signal processing applications. It can be integrated with external radio frequency (RF) gear to develop real-world SDR applications or used on its own in a simulation environment. GNU Radio is widely used in academic, research, and commercial settings to aid wireless communication studies and real-world radio system development. The framework facilitates the development of software-defined radio applications using modular components known as flowgraphs, which depict interconnected signal processing blocks that determine data flow.

GNU Radio Companion (GRC):

GNU Radio Companion (GRC) is a graphical user interface (GUI) for creating GNU Radio-based apps. It acts as a front-end for the underlying GNU Radio libraries, allowing users to design signal processing workflows visually. GRC was originally developed by Josh Blum during his time at Johns Hopkins University (2006-2007), and was later made available as free software during the October 2009 Hackfest. Since 3.2.0, it has been officially included in the GNU Radio distribution, making it easier to construct SDR applications using a visual programming technique.

Amplitude Shift Keying:

In digital communication, various modulation techniques are employed to transmit data or messages to a receiver over a communication channel. One such technique is Amplitude Shift Keying (ASK), a modulation scheme in which the amplitude of a carrier signal is varied to encode and transmit information. ASK is widely utilized in real-world applications, including radio and television broadcasting, as well as digital data transmission systems.

ASK Modulation:

Amplitude Shift Keying (ASK) is a digital modulation technique in which digital information is transmitted by varying the amplitude of a carrier signal. In ASK, a high-amplitude carrier signal represents a binary '1', while a low-amplitude carrier signal corresponds to a binary '0'. This technique involves the superimposition of a high-frequency sinusoidal carrier signal with a digital message signal. The binary message, composed of '1's and '0's, modulates the carrier signal's amplitude, generating a modulated waveform that is transmitted over the communication channel.

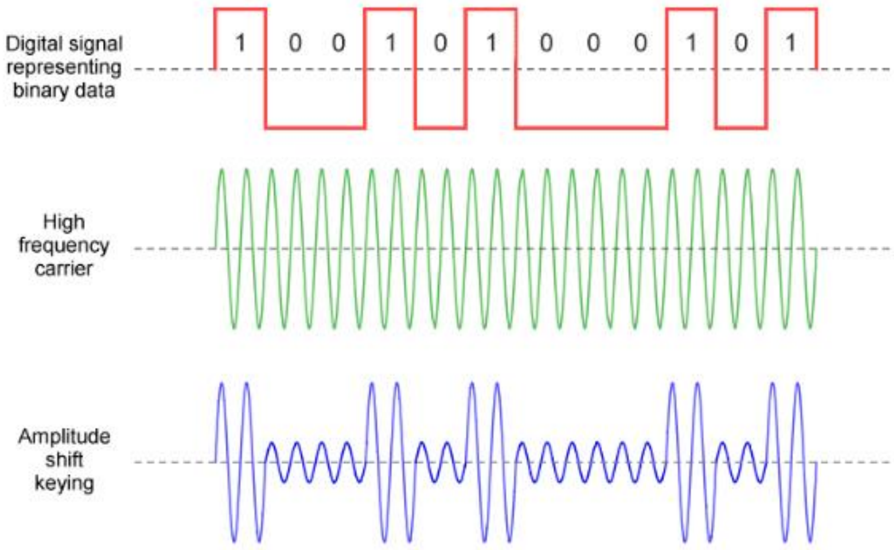


Figure 1.1. ASK waveforms

ASK Demodulation:

Demodulation in Amplitude Shift Keying (ASK) is the process of recovering the original digital message signal from the modulated carrier wave by reversing the modulation applied during transmission. ASK demodulation can be achieved using different techniques, primarily envelope detection and coherent detection.
Types of ASK Demodulation
ASK demodulation is broadly classified into two main techniques:

- Envelope Detection
- Coherent Detection

Envelope detection

Envelope detection is a basic method for demodulating Amplitude Shift Keying (ASK) signals that involves extracting the signal's envelope. Receiving the modulated signal, rectifying it using a diode, removing the carrier with a low-pass filter, and recovering the original digital data using threshold detection are the steps involved.

Coherent detection

Coherent detection is an ASK demodulation technique that maintains phase synchrony with the carrier wave. It entails combining the received signal with a local oscillator, filtering to remove the data signal, establishing an amplitude threshold, and comparing it to decode the transmitted binary data, therefore recovering the original message signal.

ASK Applications

Amplitude shift keying applications are mentioned below.

They are:

- Low-frequency RF applications
- Home automation devices
- Industrial networks devices
- Wireless base stations
- Tire pressuring monitoring systems

BLOCKS of GNU RADIO used:

1. Options Block: Defines global settings for the flowgraph.
   Parameters:
   Title: The name of the flowgraph (not specified here).
   Output Language: Python.
   Generate Options: QT GUI for visualizing signals.

2. Variable Block
   Variable Name: samp_rate.
   Value: 200k.
   Purpose: Sets the sample rate for all the blocks in the flowgraph, ensuring consistent signal processing.

3. Signal Sources
   Three signal sources are used for different purposes:
   Carrier Signal:
   Waveform: Cosine.
   Frequency: 10 kHz.
   Amplitude: 1.
   Purpose: Acts as the carrier signal for ASK modulation.
   Modulating Signal:
   Waveform: Cosine.
   Frequency: 500 Hz.
   Amplitude: 1.
   Purpose: Represents the digital data (modulation signal) to modulate the carrier.
   Demodulation Signal:
   Waveform: Cosine.
   Frequency: 10 kHz (same as the carrier).
   Amplitude: 1.
   Purpose: Used to demodulate the received ASK signal.

4. Throttle Blocks: Limits the processing speed of the flowgraph to the specified samp_rate (200k) to avoid excessive CPU usage.

5. Multiply Block (Modulation): Multiplies the carrier signal with the modulating signal to perform ASK modulation.
   Input 1: Carrier signal.
   Input 2: Modulating signal.

6. Float to Complex Block: Converts the real-valued ASK signal to a complex signal for visualization and further processing.

7. QT GUI Frequency Sink: Displays the frequency spectrum of the signals.
   Parameters:
   FFT Size: 1024.
   Center Frequency: 10 kHz (for modulated/demodulated signals).
   Bandwidth: 200 kHz (matching samp_rate).

8. QT GUI Constellation Sink: Displays the constellation diagram of the complex signal after modulation or demodulation.

9. QT GUI Time Sink: Visualizes the time-domain waveform of the signals.

PROCEDURE:

1. Click on the GNU Radio icon to launch GNU Radio companion.
2. Go to file option then select new→ QT GUI. And save the file.
3. Click on Variable block and set the sample rate 200k (default value 32k).
4. In order to design ASK modulator we need various block such as source, multiply, QT GUI Frequency sink and QT GUI Time sink.
5. We need two signal source blocks, one for message signal and another for the carrier signal.
6. Location of signal source block is
   Core →waveform generator→ signal source
   Double click on signal source block (carrier signal block), select output type: float, waveform: cosine, frequency: 16000 (16 kHz), Amplitude: 1, offset: 0 for the carrier signal.
7. Double click on the second signal source block (message signal block), select output type: float, waveform: square, frequency: 500 (500 Hz), Amplitude: 1, offset: 0 for the message signal.
8. We need one throttle block. Location of throttle is
   Core → mise → throttle
9. We need one multiply block. Location of multiply is
   Core → math operator → multiply
   Set the output type: float
10. We need QT GUI Frequency sink block to see the output frequency response. Location of QT GUI Frequency sink is
    Core → instrumentation → QT→ QT GUI Frequency sink
    Set the output type: float
11. We need one QT GUI Time sink block to see the output time domain response. Location of QT GUI Time sink is
    Core → instrumentation → QT→ QT GUI Time sink
    Set the output type: float
12. Take care to set the input/output type to float in all the blocks.
13. Now connect the different blocks as shown in Figure 2.4.
14. Click on run option.

BLOCK DIAGRAM: -

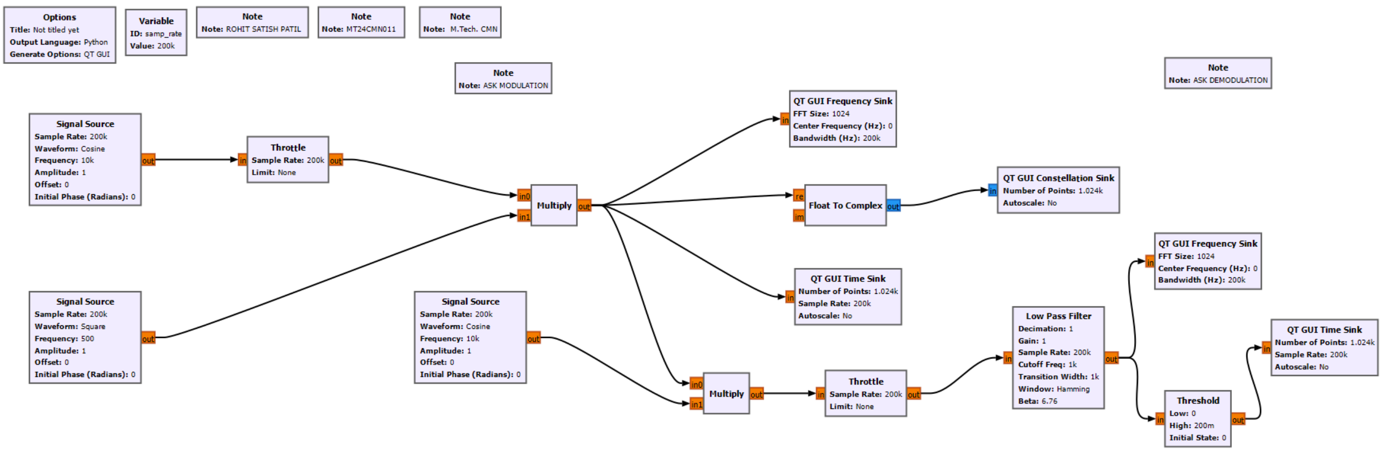


Figure 1.2. Block Diagram (GNU)

RESULT: -

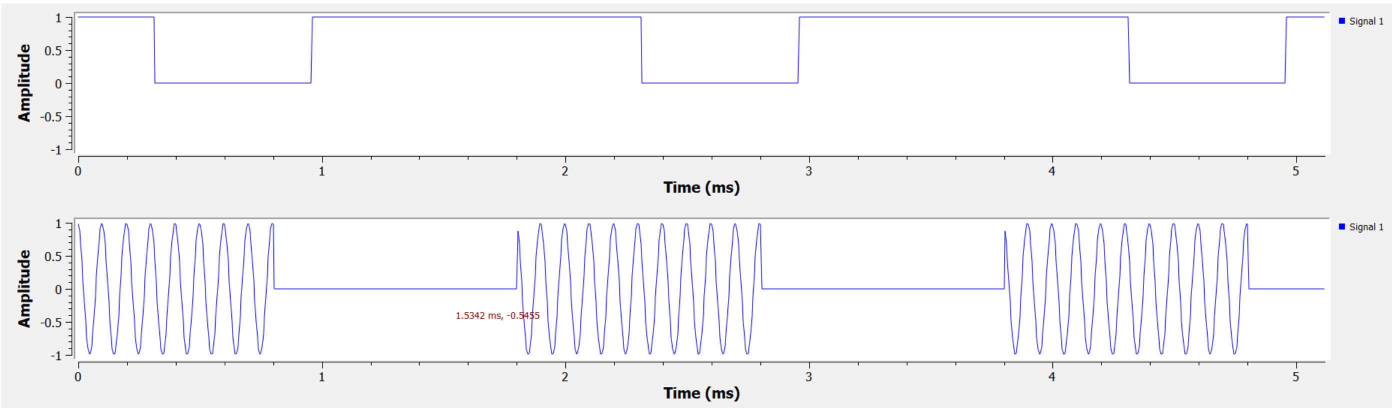


Figure 1.3. (a) Demodulated waveform (b) Modulated waveform

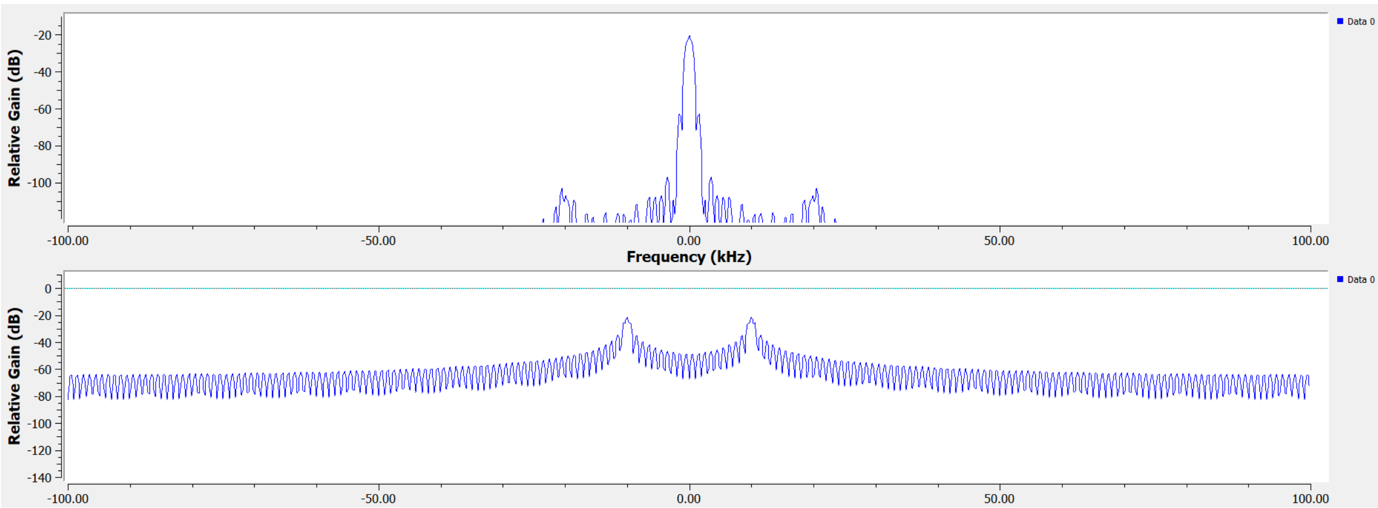


Figure 1.4. (a)Demodulated waveform (b)Modulated waveform

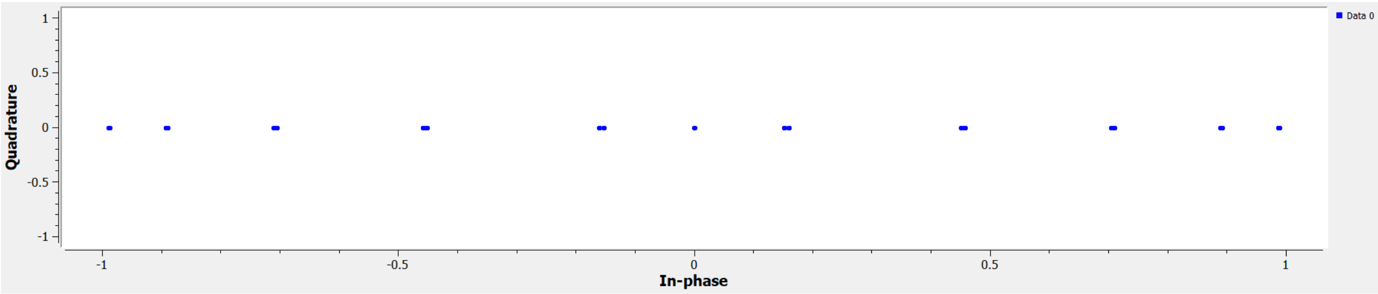


Figure 1.5. Constellation diagram

CONCLUSION: -

The ASK modulation and demodulation experiment with GNU Radio constructed and studied Amplitude Shift Keying (ASK), a fundamental digital modulation method. The experiment proved binary data transmission by altering the amplitude of the carrier signal and retrieving it using demodulation. GNU Radio was used to view ASK waveforms, spectra, and constellation diagrams, which provided insights into signal behaviour during transmission and reception. This analysis aided in a better understanding of ASK modulation principles, such as signal representation and decoding procedures, highlighting its importance in digital communication systems. The experiment demonstrated the practical use of GNU Radio for simulating and analysing digital modulation techniques.

NAME: - ROHIT SATISH PATIL **M.Tech.** CMN
YEAR: - 1st Year **SEM: -** 2nd Sem
SUBJECT: - CS-II Lab **EXPERIMENT 2**

AIM: - To perform QAM using GNU radio.

SOFTWARE: - GNU radio.

THEORY: -

Quadrature amplitude modulation (QAM):

Quadrature Amplitude Modulation (QAM) is a popular modulation technique in modern telecommunications, which includes both digital and analog modulation systems. It enables the transmission of two distinct analog message signals or digital bit streams by modulating the amplitudes of two orthogonal carrier waves with Amplitude Shift Keying (ASK) for digital signals and Amplitude Modulation (AM) for analog signals. These carriers, which have the same frequency, are phase-shifted by 90° to ensure orthogonality. The transmitted signal is the sum of these two modulated carriers, which allows for coherent separation and demodulation at the receiver.

QAM is distinguished by its adherence to the narrowband assumption, which states that the modulated waveforms have a substantially smaller bandwidth than the carrier frequency. Phase Modulation (PM) and Phase Shift Keying (PSK) are specific examples of QAM in which the amplitude is constant but the phase varies. Similarly, FM and FSK are expansions of phase modulation.

QAM is widely used in digital communication systems, including the IEEE 802.11 Wi-Fi standards, optical fiber networks, and cable television. Its spectral efficiency can be improved by increasing the constellation size, which is limited primarily by channel noise and linearity. Optical implementations, such as QAM16 and QAM64, use interferometers to simulate QAM transmission.

The constellation diagram is an important tool in QAM, with symbols commonly organized in a square grid, but other configurations, such as hexagonal or triangular layouts, exist. Because digital data is binary, the number of constellation points usually correlates to powers of two (for example, 16-QAM, 64-QAM, 256-QAM). Non-square constellations, such as Cross-QAM, can improve efficiency, but their additional complexity restricts their practical application.

Higher-order QAM increases data transfer per symbol; nevertheless, keeping constant mean energy necessitates closer spacing between constellation points, making the signal more susceptible to noise and raising bit error rate (BER). To achieve successful high-order QAM transmission, the signal-to-noise ratio (SNR) must be enhanced by increasing signal strength, reducing noise, or doing both. QAM is chosen over PSK at high data rates because it is more efficient in optimizing the Euclidean distance between symbols.

QAM's most common applications are digital cable television and cable modems. In the United States, the ANSI/SCTE 07 2013 standard requires 64-QAM and 256-QAM for digital cable, while in the United Kingdom, 64-QAM is used for Freeview and 256-QAM is used for Freeview HD broadcasts.

BLOCKS of GNU RADIO used:

1. 1. Options Block: Sets general properties for the flowgraph. Parameters: Title: The name of the flowgraph(QAM-16). Output Language: Python. Generate Options: Specifies the GUI type (QT GUI in this case).

2. Variable Blocks: Define key parameters used in the flowgraph: samp_rate: Sets the sample rate (1M samples per second). mod_order: Determines the number of bits per symbol.

3. Random Source: Generates random bytes (representing digital data to be transmitted). Parameters: Minimum/Maximum: Range of random values (0 to 255). Num Samples: Number of samples (1k). Repeat: Ensures data repeats for continuous transmission.

4. Packed to Unpacked: Converts bytes into bits for modulation. Parameters: Bits per Chunk: Sets the number of bits per symbol (4 for QPSK). Endianness: Specifies the bit order (MSB first).

5. Chunks to Symbols: Maps input bits to complex symbols based on the selected modulation scheme.

6. Noise Source: Adds Gaussian noise to the modulated signal to simulate channel interference. Parameters: Noise Type: Gaussian. Amplitude: Controls noise strength (currently 0, i.e., no noise).

7. Signal Source: Generates a carrier signal (cosine waveform) for modulation. Parameters: Sample Rate: Matches the samp_rate (1M). Waveform: Cosine wave. Frequency: Carrier frequency (0 Hz in this setup). Amplitude: Signal amplitude (1).

8. Multiply: Modulates the carrier with the digital signal.

9. Add: Combines the modulated signal with noise.

10. Throttle: Limits the processing speed to the specified sample rate (1M) to avoid excessive CPU usage.

11. QT GUI Constellation Sink: Displays the received signal's constellation diagram. Parameters: Number of Points: Total displayed points (1,024). Autoscale: Disabled (fixed scale).

12. QT GUI Time Sink: Visualizes the time-domain waveform of the signal. Parameters: Number of Points: 1,024 samples. Sample Rate: Matches the samp_rate.

PROCEDURE:

1. Click on the GNU Radio icon to launch GNU Radio companion.

2. Go to file option then select new→ QT GUI. And save the file.
3. Click on Variable block and set the sample rate 1M (default value 32k).
4. In order to design QAM we need various block such as random source, signal source, packed to unpacked, chunks to symbols, noise source, multiply, add, throttle, QT GUI Time sink and QT GUI Constellation sink.
5. Location of random source block is: Core →waveform generator→ random source Minimum=0, Maximum=255, Num of samples=1k
6. For packed to unpacked: Core → byte operator→ packed to unpacked
7. For Chunks to symbols: Core → Symbol coding→ Chunks to symbols
8. Location of noise source block is: Core →waveform generator→ noise source
9. We need one add block. Location of add is: Core → math operator→ add
10. We need one throttle block. Location of throttle is: Core → mise → throttle
11. We need one multiply block. Location of multiply is: Core → math operator → multiply Set the output type: float
12. We need QT GUI Constellation sink block to see the constellation. Location of QT GUI Constellation sink is: Core → instrumentation → QT→ QT GUI Constellation sink Set the output type: float
13. We need one QT GUI Time sink block to see the output time domain response. Location of QT GUI Time sink is: Core → instrumentation → QT→ QT GUI Time sink Set the output type: float
14. Take care to set the input/output type to float in all the blocks.
15. Now connect the different blocks as shown in Figure 2.1.
16. Click on run option.

BLOCK DIAGRAM: -

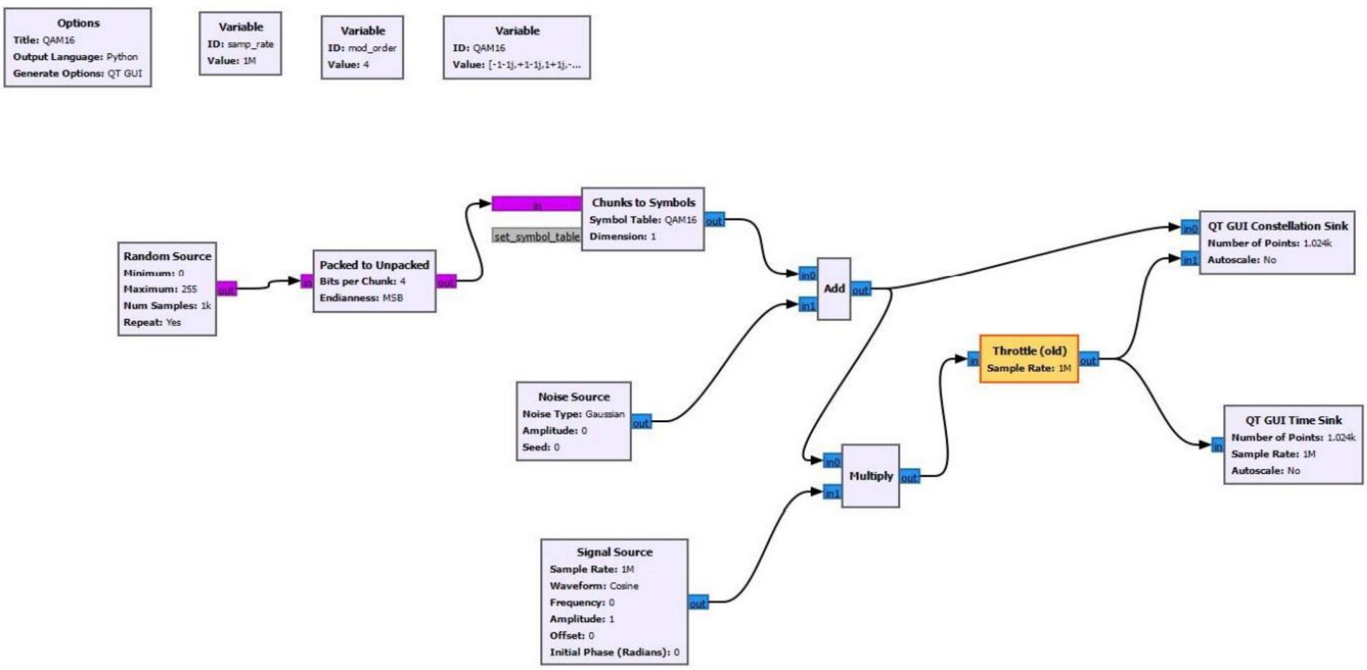


Figure 2.1. Block Diagram (GNU)

RESULT: -

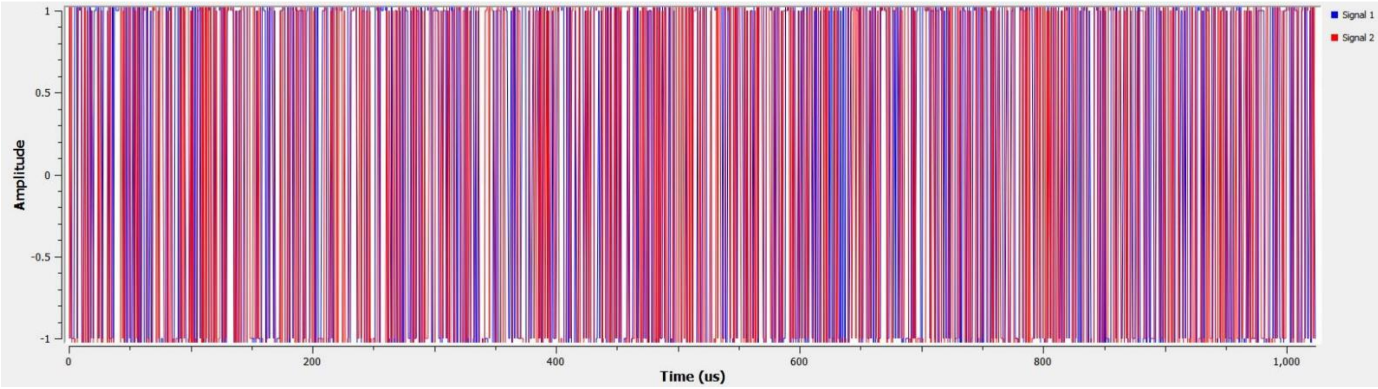


Figure 2.2. Time Domain

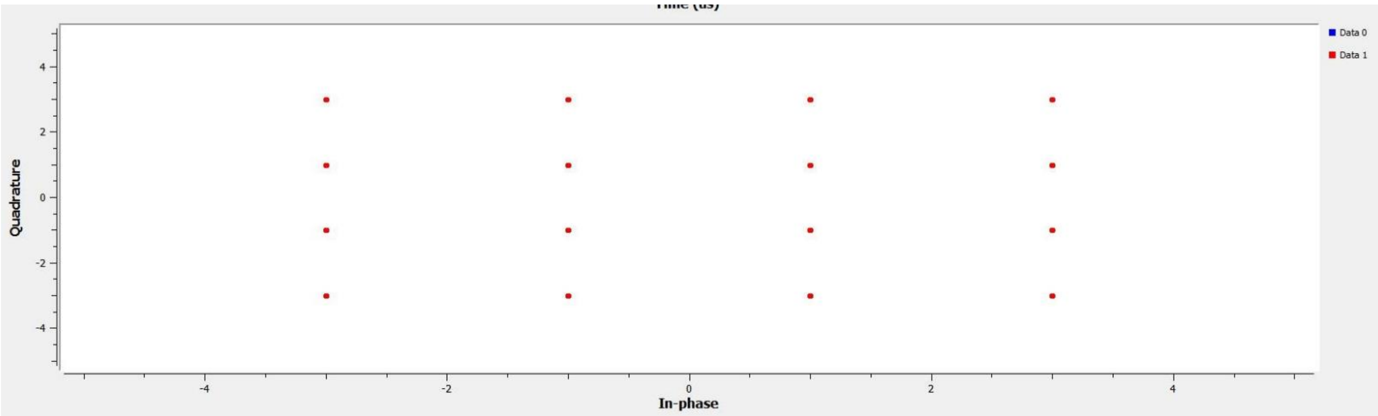


Figure 2.3. Constellation diagram

CONCLUSION: -

We tested and studied Quadrature Amplitude Modulation (QAM) using GNU Radio, a sophisticated software-defined radio (SDR) tool. The approach illustrated digital modulation concepts and provided insights into practical communication systems.

NAME: - ROHIT SATISH PATIL **M.Tech.** CMN
YEAR: - 1st Year **SEM: -** 2nd Sem
SUBJECT: - CS-II Lab **EXPERIMENT 3**

AIM: - To perform Audio/Music transmit and receive on USRP.

SOFTWARE: - GNU radio.

THEORY: -

Audio transmission using GNU Radio and Universal Software Radio Peripheral (USRP) involves modulating an audio signal and transmitting it over a wireless channel. This setup is commonly used for experimentation in Software-Defined Radio (SDR) applications, including FM transmission and reception.

Audio reception using GNU Radio and USRP involves capturing an FM-modulated signal, demodulating it, and playing back the extracted audio. The UHD: USRP Source block receives the transmitted signal at a specified frequency. Automatic Gain Control (AGC) optimizes signal strength before passing it to the WBFM Receive block, which demodulates the audio. The signal is then converted from complex to float for processing. A QT GUI Sink visualizes the spectrum, and the Audio Sink block plays the recovered audio in real-time. This setup enables wireless SDR-based audio reception effectively.

BLOCKS of GNU RADIO used:

Transmitter Side Blocks:

1. WAV File Source – Loads the audio file.
2. Multiply Const – Adjusts the signal amplitude.
3. WBFM Transmit – Modulates the audio using wideband FM.
4. UHD: USRP Sink – Transmits the modulated signal wirelessly.
5. QT GUI Frequency Sink – Displays the transmitted signal spectrum.

Receiver Side Blocks:

1. UHD: USRP Source – Captures the transmitted signal.
2. AGC (Automatic Gain Control) – Adjusts the signal gain dynamically.
3. WBFM Receive – Demodulates the FM signal.
4. Float to Complex & Complex to Float – Converts signal formats for processing.
5. QT GUI Sink – Visualizes the received signal.
6. Audio Sink – Outputs the demodulated audio.

PROCEDURE:

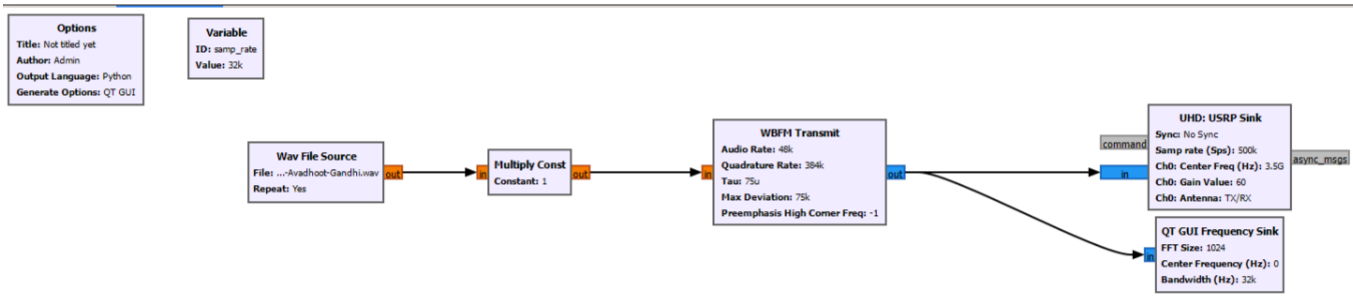
Transmitter Setup:

1. Open GNU Radio Companion (GRC) and create a new flowgraph.
2. Add a WAV File Source block and select the desired audio file.
3. Use a Multiply Const block to scale the signal amplitude appropriately.
4. Connect the output to a WBFM Transmit block and set the required parameters (audio rate, quadrature rate, max deviation, etc.).
5. Link the modulated signal to a UHD: USRP Sink block and set the transmission frequency, sampling rate, and antenna settings.
6. Add a QT GUI Frequency Sink to monitor the transmitted signal.
7. Run the flowgraph to start transmitting the audio signal.

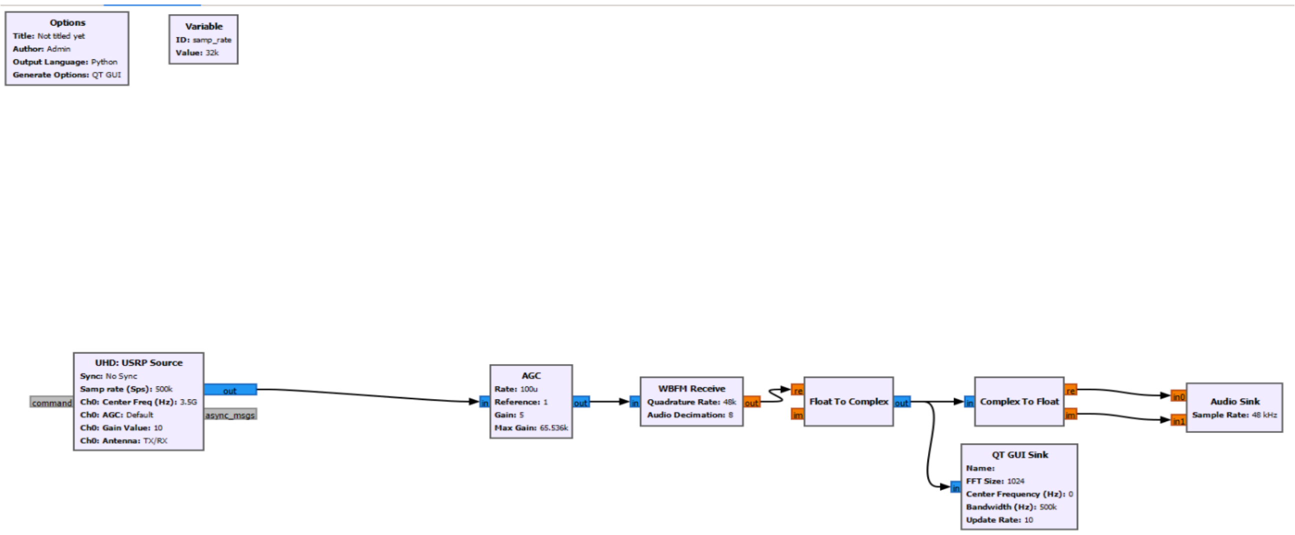
Receiver Setup:

1. Open GNU Radio Companion (GRC) and create a new flowgraph for reception.
2. Add a UHD: USRP Source block and set the frequency, sample rate, and antenna settings to match the transmitter.
3. Insert an AGC (Automatic Gain Control) block to adjust the received signal strength.
4. Use a WBFM Receive block to demodulate the FM signal.
5. Convert the signal using Float to Complex and Complex to Float blocks.
6. Add a QT GUI Sink to visualize the received spectrum.
7. Use an Audio Sink block with an appropriate sample rate (e.g., 48 kHz) to play the demodulated audio.
8. Run the flowgraph to start receiving and listening to the transmitted audio.

BLOCK DIAGRAM: -



(a) Transmitter



(b) Receiver

Figure 3.1. Block Diagram (GNU)

RESULT: -

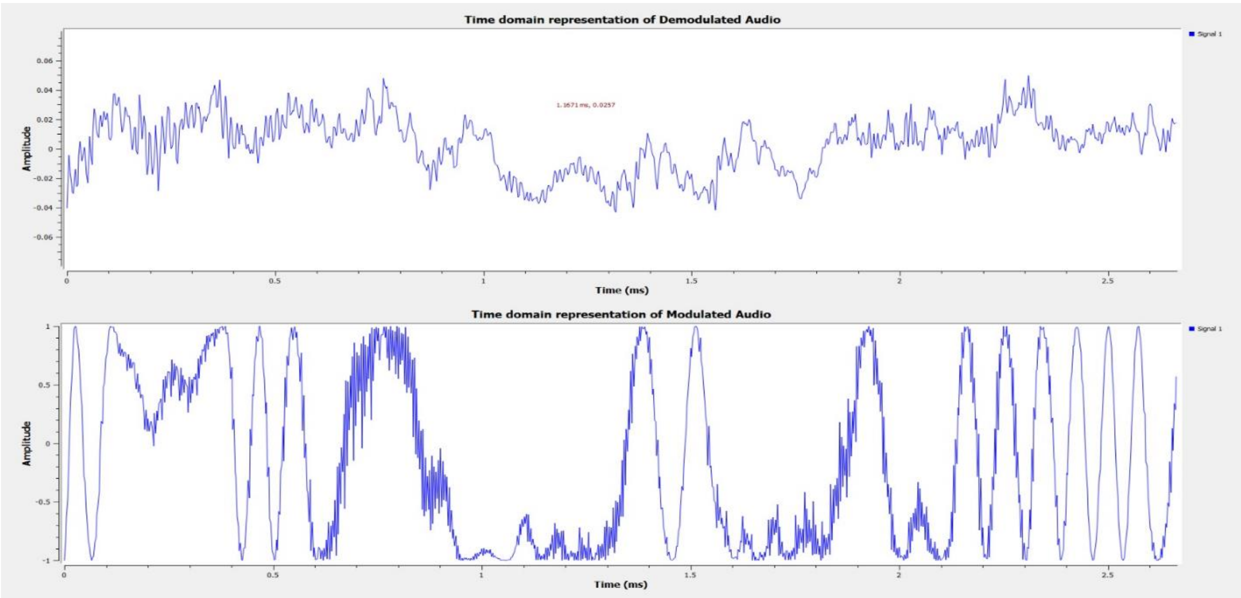


Figure 3.2.

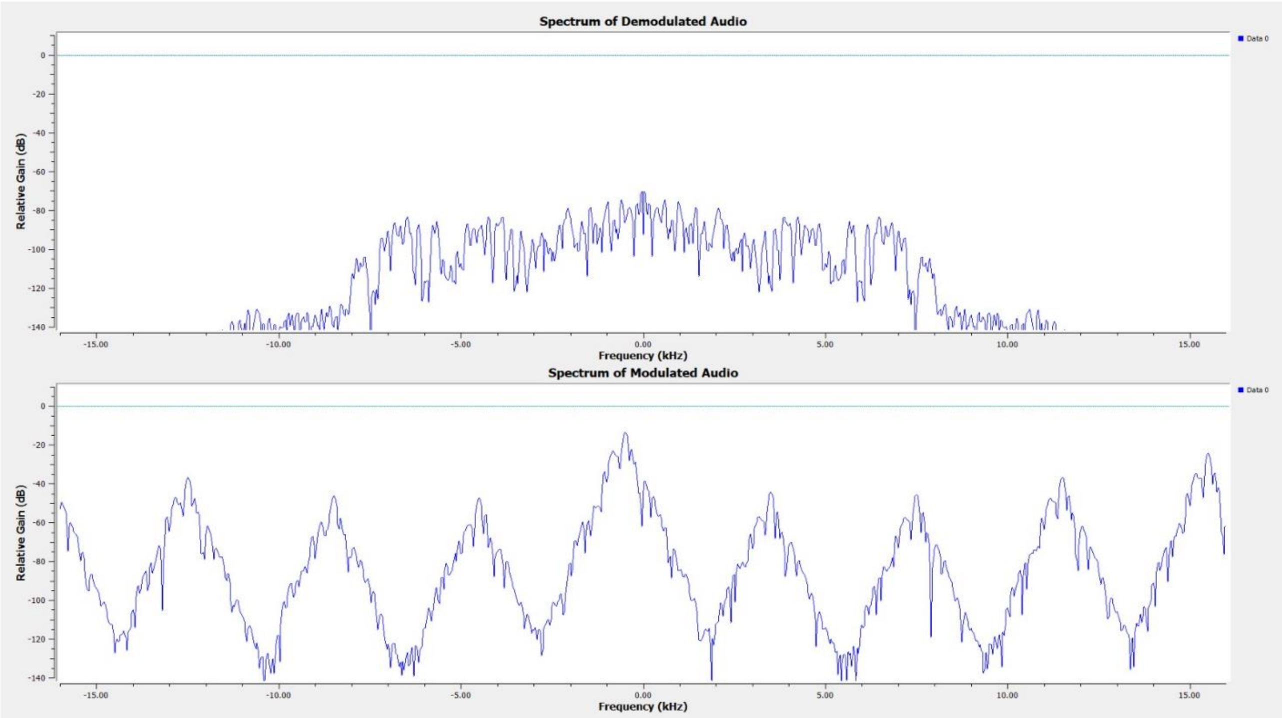


Figure 3.3.

CONCLUSION: -

Using GNU Radio and USRP gear, this experiment effectively illustrates the transmission and receiving of audio. Wideband FM (WBFM) is used on the transmitter end to modulate an audio signal, and it is demodulated at the receiver end. QT GUI sinks are used to visually inspect the signal, guaranteeing correct transmission and reception. While the conversion blocks guarantee correct data format processing, the usage of AGC aids in signal strength optimization. Key ideas in signal processing and software-defined radio applications are reinforced by the final output at the receiver, which attests to the efficacy of SDR-based audio communication.

NAME: - ROHIT SATISH PATIL **M.Tech.** CMN
YEAR: - 1st Year **SEM: -** 2nd Sem
SUBJECT: - CS-II Lab **EXPERIMENT 4**

AIM: - To design and implement a simplex (one-way) real-time audio transmission and reception system using GNU Radio and a USRP (Universal Software Radio Peripheral).

SOFTWARE: - GNU Radio (with the Companion GUI)

UHD drivers (for USRP)

Audio drivers (compatible with your operating system)

THEORY: -

Frequency Modulation (FM)

Frequency Modulation (FM) is a technique in which the instantaneous frequency of the carrier wave is varied in proportion to the amplitude of the input signal (in this case, audio). Wideband FM (WBFM) is typically used for high-fidelity audio transmissions (e.g., commercial broadcast FM radio).

Simplex Communication

Simplex communication is one-way communication, where data flows in only one direction from the transmitter to the receiver. In this lab, we will transmit audio from a microphone (or a stored audio source) and receive it on another USRP without providing a return link.

GNU Radio Flow Graphs

GNU Radio allows us to build signal processing chains (flow graphs) visually in the GNU Radio Companion (GRC). Each block performs a specific function, such as source (audio or USRP), modulation (WBFM), filtering, and sink (audio or USRP).

USRP Hardware

The USRP (Universal Software Radio Peripheral) is a hardware device used for implementing software-defined radio. It provides an interface for transmitting and receiving signals in a wide range of radio frequencies, controlled via the UHD (USRP Hardware Driver).

BLOCKS of GNU RADIO used:

Below is a list of key GNU Radio blocks typically used in the transmitter and receiver flow graphs.

Transmitter Side

1. **Audio Source**
   - Captures audio from the system microphone or a file.
   - Sample Rate (commonly 48 kHz or 44.1 kHz, but can be set to 32 kHz, 48 kHz, or 96 kHz depending on your audio device).
2. **Multiply Const.**
   - Adjusts the amplitude of the audio signal.
3. **WBFM Transmit**
   - Implements wideband FM modulation.
   - Parameters: Audio Rate (e.g., 48 kHz), Quadrature Rate (often 200 kHz or 192 kHz), Tau (de-emphasis or pre-emphasis time constant), Max Deviation (peak frequency deviation).
4. **Complex to Float**
   - Splits the complex baseband signal into real (I) or imaginary (Q) components if needed for visualization.
5. **QT GUI Time Sink / QT GUI Frequency Sink**
   - Allows time-domain and frequency-domain visualization of the signal.
6. **UHD: USRP Sink**
   - Sends the modulated signal to the USRP hardware for over-the-air transmission.
   - Key parameters:
     - **Sample Rate**: Must match the Quadrature Rate or a rate that is integer-related to the Quadrature Rate.
     - **Centre Frequency (RF centre frequency)**: Choose an appropriate transmit frequency. For us 3.5 GHz.
     - **Gain**: Adjusts the transmit power.
     - **Antenna**: Choose the antenna port (e.g., TX/RX).

Receiver Side

1. **UHD: USRP Source**
   - Receives the RF signal from the USRP hardware.
   - Key parameters:
     - **Sample Rate**: Matches the transmitter's RF bandwidth or is set appropriately to capture the FM signal.
     - **Centre Frequency**: Must match the transmitter frequency. For us 3.5 GHz.
     - **Gain**: Adjusts the front-end LNA gain.
2. **Decimating FIR Filter / Low-pass Filter**
   - Used to decimate and filter the incoming signal to a suitable rate before demodulation.
3. **AGC (Automatic Gain Control)**
   - Adjusts signal amplitude automatically.
4. **WBFM Receive**
   - Demodulates the received wideband FM signal.
   - Parameters: Quadrature Rate, Audio Decimation, De-emphasis or Pre-emphasis settings.
5. **Multiply Const.**
   - Adjusts the amplitude of the demodulated audio if needed.
6. **Complex to Float**
   - If the demodulated output is complex, this block can separate it into real/imag components.
7. **Audio Sink**
   - Outputs the demodulated audio to the system speakers.
8. **QT GUI Frequency Sink / QT GUI Time Sink**
   - Visualizes the received signal in frequency/time domain for debugging.

PROCEDURE:

Transmitter Side Graph

1. Open GNU Radio Companion and create a new flow graph.
2. Add an Audio Source block:
   - Set the Sample Rate (e.g., 48 kHz) to match your sound card.
3. Insert a Multiply Const. block to control audio amplitude.

4. Add a WBFM Transmit block:
   - Set the Audio Rate to 48 kHz.
   - Set the Quadrature Rate (e.g., 192 kHz).
   - Set Deviation to an appropriate value (e.g., 75 kHz for typical broadcast).
   - Set Pre-emphasis if desired (e.g., Tau = 75e-6).
5. Add QT GUI Time Sink / Frequency Sink blocks to visualize the signal.
6. Add a UHD: USRP Sink block:
   - Set Sample Rate = 192 kHz (or whatever matches your WBFM Transmit quadrature rate).
   - Set Centre Frequency to a valid transmit frequency within your hardware's capabilities and licensing regulations (e.g., in an ISM band or with a proper license).
   - Adjust Gain as needed.
   - Select the appropriate Antenna port (e.g., TX/RX).
7. Connect the blocks as follows:
   - Audio Source → Multiply Const. → WBFM Transmit → (optional visualization blocks) → UHD: USRP Sink.
8. Save the flow graph and run it. The transmitter is now sending out a WBFM signal.

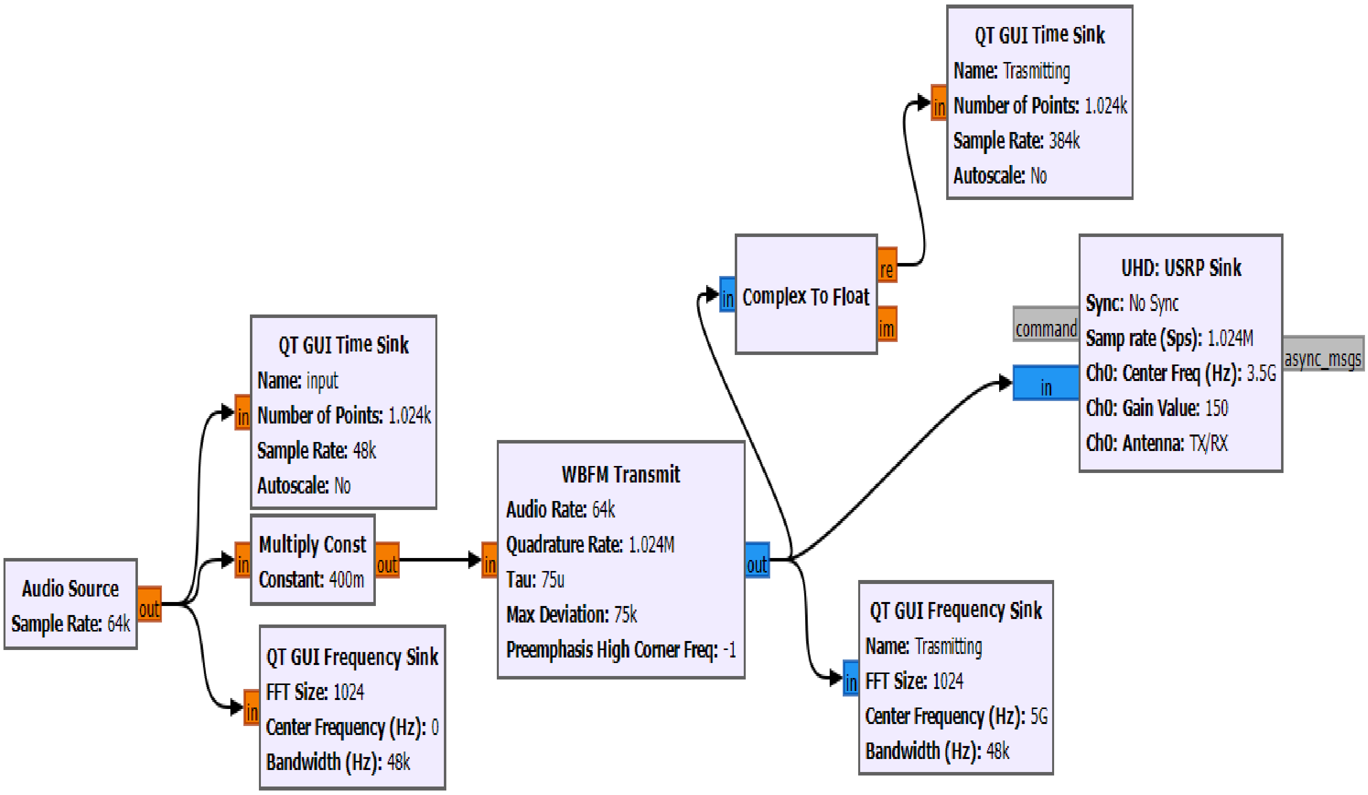
Receiver Side Graph

1. Create another flow graph in GNU Radio Companion.
2. Add a UHD: USRP Source block:
   - Set Centre Frequency to the same frequency used by the transmitter.
   - Set Gain as needed.
   - Set Sample Rate to capture the FM signal bandwidth (e.g., 192 kHz).
3. Add a Decimating FIR Filter to reduce the rate if necessary.
4. Insert an AGC block to stabilize signal amplitude.
5. Add a WBFM Receive block:
   - Set Quadrature Rate to match the USRP Source sample rate or the rate after decimation.
   - Set Audio Decimation so that the final audio rate is suitable (e.g., 48 kHz).
   - Adjust de-emphasis settings if required.
6. Add a Multiply Const. block to set the audio volume level.
7. Add an Audio Sink block:

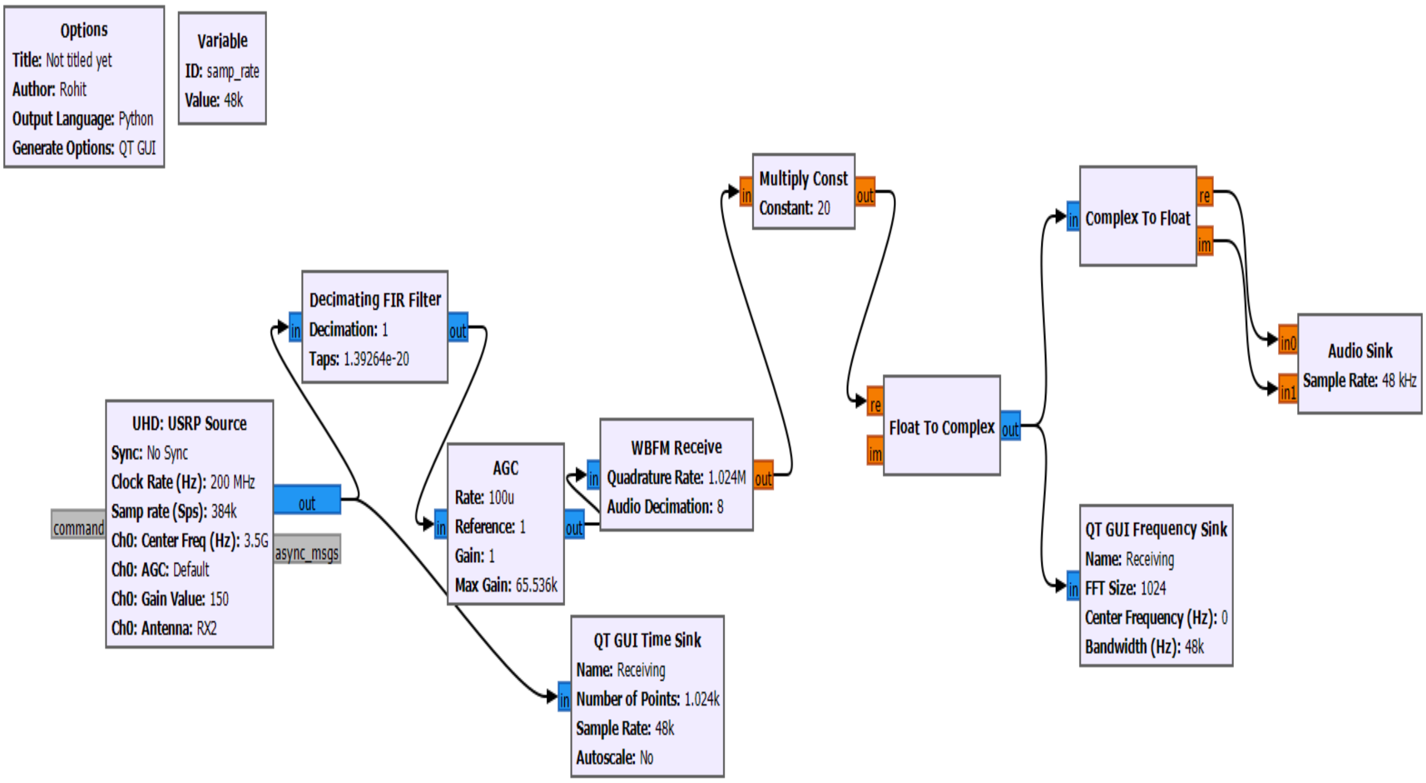
  - Ensure the Audio Sink sample rate matches the WBFM Receive output rate (e.g., 48 kHz).
8. Add QT GUI Time Sink / Frequency Sink to visualize the received audio or intermediate signals.
9. Connect the blocks as follows:
  - UHD: USRP Source → (optional Filter / AGC) → WBFM Receive → Multiply Const. → Audio Sink.
10. Save the flow graph and run it. You should hear the transmitted audio on the receiver side.

BLOCK DIAGRAM: -

Transmitter Overview

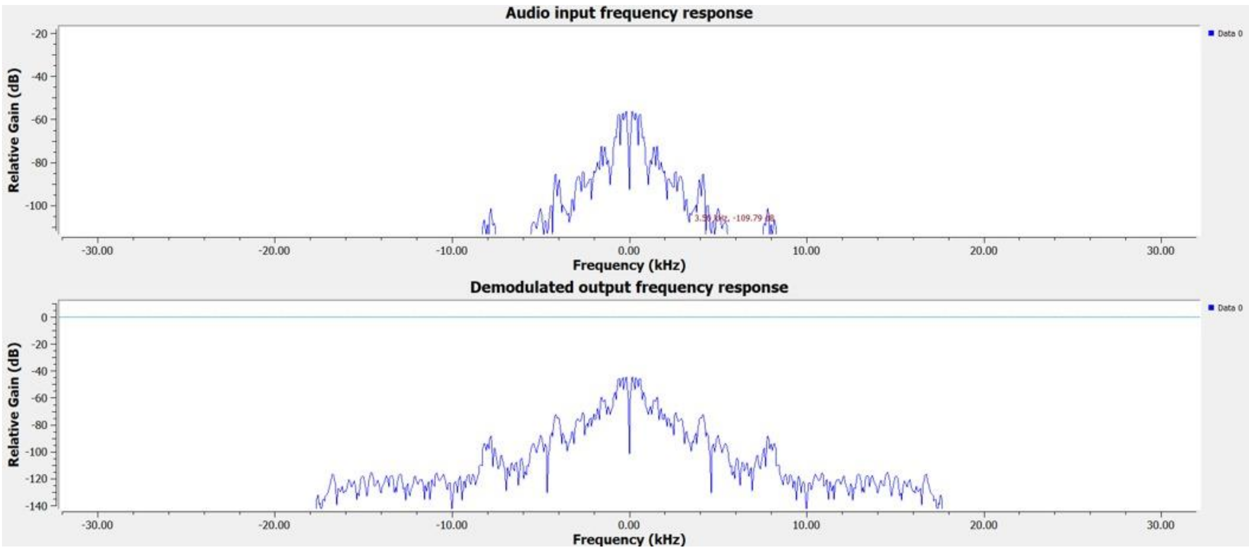


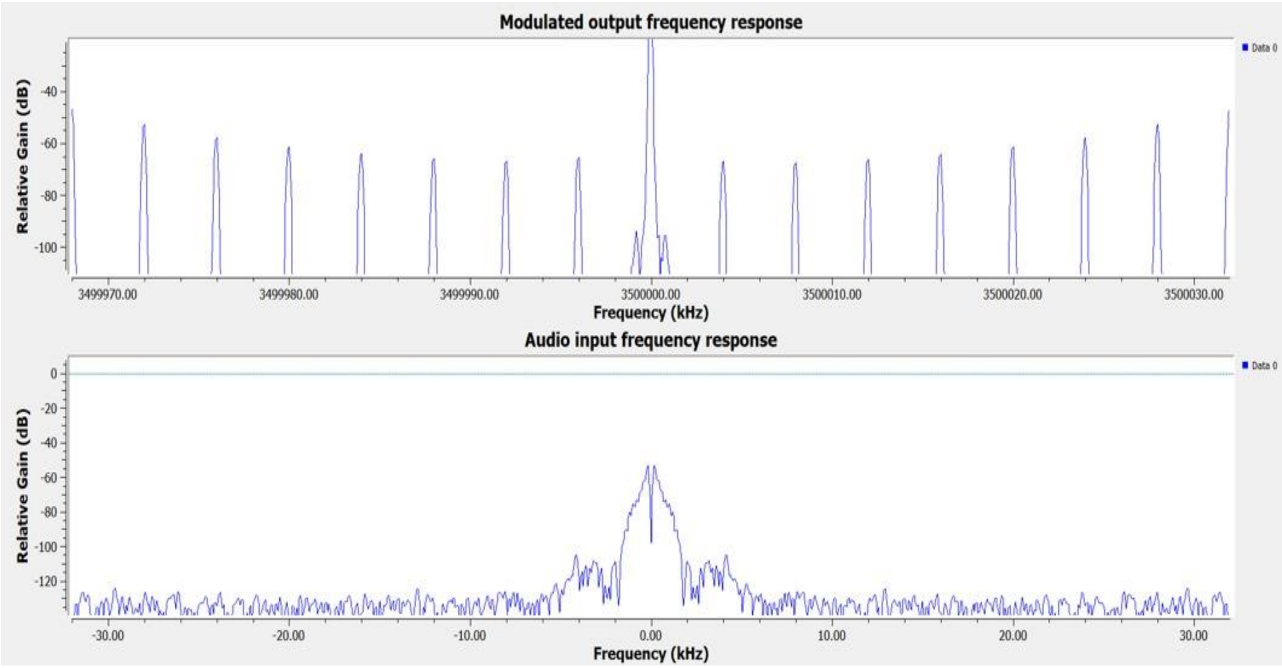
Receiver Overview

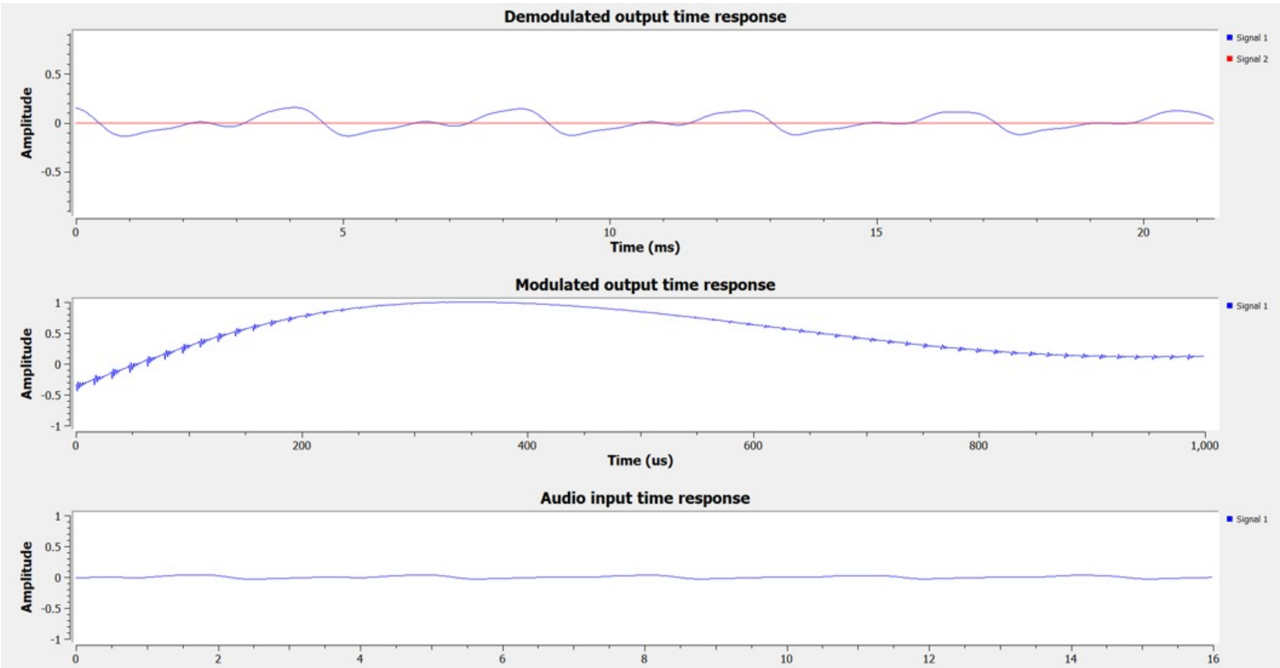


RESULT: -

After running the transmitter flow graph on one USRP and the receiver flow graph on another (or the same device in loopback mode, if supported), you should hear the transmitted audio on the receiver side. You can observe the real-time audio waveforms and spectra in the QT GUI sinks if included.







CONCLUSION: -

In this lab, we successfully demonstrated simplex real-time audio transmission using wideband FM modulation in GNU Radio. By carefully setting the transmit and receive parameters (centre frequency, sample rates, gains), we achieved reliable one-way real-time audio communication. This exercise showcases how GNU Radio and USRP hardware can be used to prototype and experiment with software-defined radio systems for real-time audio broadcasting.

FINAL UNDERSTANDINGS:

If you change the values of these parameters, different effects will occur in the real-time audio transmission and reception system using GNU Radio. Here's what happens if values are increased or decreased:

1) Audio Source - Sample Rate (64k)

- Increased: Higher fidelity but requires more processing power and bandwidth. If the receiver's sample rate is not adjusted accordingly, aliasing or distortion can occur.
- Decreased: Reduces audio quality, potentially causing loss of high-frequency details. May cause mismatch with the modulation and transmission process.

2) Multiply Const (400m)

- Increased: Amplifies the signal, potentially causing distortion or clipping if the values exceed the dynamic range.
- Decreased: Reduces signal strength, leading to poor transmission, lower signal-to-noise ratio (SNR), and difficulty in demodulation.

3) QT GUI Time Sink - Number of Points (1.024k)

- Increased: Smoother waveform visualization but requires more processing.
- Decreased: Lower resolution of waveform display, making fine details harder to observe.

4) QT GUI Frequency Sink

- FFT Size (1024)
  - Increased: Higher frequency resolution but more computation.
  - Decreased: Lower frequency resolution, making it harder to distinguish frequencies.
- Center Frequency (0 Hz)
  - Changed: Shifts the displayed spectrum.
- Bandwidth (64k)
  - Increased: Shows a broader frequency range but may capture unwanted noise.
  - Decreased: Focuses on a smaller frequency range, potentially missing parts of the signal.

5) WBFM Transmit

- Audio Rate (64k)
  - Increased: More data, requiring higher transmission bandwidth.
  - Decreased: Reduces audio quality.
- Quadrature Rate (1.024M)
  - Increased: Better FM signal resolution but higher computational load.
  - Decreased: Less accurate FM modulation.
- Tau (75u)
  - Changed: Alters pre-emphasis filter response, affecting high-frequency components.

- Max Deviation (75kHz)
  - Increased: Occupies more bandwidth, potentially interfering with adjacent signals.
  - Decreased: Reduces bandwidth, but also lowers audio fidelity.
- Preemphasis High Corner Freq (-1)
  - Changed: Affects the frequency response of the audio.

6) UHD: USRP Sink

- Sample Rate (64k)
  - Increased: Needs more bandwidth, increasing processing requirements.
  - Decreased: Reduces signal clarity and transmission effectiveness.
- Center Frequency (3.5 GHz)
  - Changed: Shifts the transmission to a different frequency.
- Gain (60 dB)
  - Increased: Can cause signal saturation and interference.
  - Decreased: Weakens the signal, possibly making it undetectable.
- Bandwidth (40M)
  - Increased: May allow more signal variations but requires higher system capabilities.
  - Decreased: Reduces the range of transmitted frequencies.

7) UHD: USRP Source

- Sample Rate (64k)
  - Increased: Requires higher bandwidth but provides better resolution.
  - Decreased: Reduces clarity, leading to aliasing.
- Gain (60 dB)
  - Increased: Can introduce noise.
  - Decreased: Weakens reception.
- Bandwidth (40M)
  - Increased: Captures a wider range of frequencies.
  - Decreased: Limits the frequency content received.

8) Decimating FIR Filter - Decimation Factor (1)

- Increased: Reduces sample rate, potentially causing information loss.
- Decreased: Keeps more data but increases processing load.

9) AGC (Automatic Gain Control)

- Rate (100u)
  - Increased: Faster adaptation, but more fluctuations.
  - Decreased: Slower response, causing delays in gain adjustments.
- Reference (1)
  - Changed: Alters target signal power.
- Max Gain (65.536k)
  - Increased: Can amplify noise.
  - Decreased: May not fully amplify weak signals.

10) WBFM Receive

- Quadrature Rate (1.024M)
  - Increased: Higher signal resolution, but more computational effort.
  - Decreased: Less accurate demodulation.
- Audio Decimation (16)
  - Increased: Reduces sample rate, possibly lowering audio quality.
  - Decreased: Keeps more data but requires more processing.

11) Audio Sink - Sample Rate (64k)

- Increased: Requires more processing.
- Decreased: May result in aliasing or loss of high-frequency details.

Every parameter changes affects audio clarity, transmission quality, and computational requirements. Increasing values generally improves resolution but demands more resources, while decreasing values reduces quality and may cause distortion.

NAME: - ROHIT SATISH PATIL **M.Tech.** CMN
YEAR: - 1st Year **SEM: -** 2nd Sem
SUBJECT: - CS-II Lab **EXPERIMENT 5**

AIM: -: To design, implement, and evaluate a simplex OFDM communication system using GNU Radio and USRP hardware, capable of transmitting a raw image file (byte stream) over-the-air and reconstructing the image at the receiver.

OBJECTIVES: -

1. Build and deploy an OFDM transmitter and receiver in GNU Radio Companion, interfaced with USRP hardware.
2. Stream a binary image file as OFDM packets over-the-air, loop it back, receive, and reconstruct the image.

SOFTWARE & Hardware: -

- GNU Radio (with the Companion GUI)
- UHD drivers (for USRP)
- USRP
- Laptop/PC

THEORY: -

OFDM Fundamentals

- **Subcarriers & Orthogonality:** Orthogonal Frequency Division Multiplexing (OFDM) effectively partitions the available communication channel into a series of parallel narrowband subcarriers, each spaced at a frequency of 1/T. This strategic design ensures that the subcarriers maintain orthogonality, which is crucial for preventing Inter-Carrier Interference (ICI), thereby enhancing overall transmission quality and efficiency.

- **IFFT/FFT Processing:** In the OFDM transmission process, the transmitter employs an Inverse Fast Fourier Transform (IFFT) to convert frequency-domain data into time-domain OFDM symbols. Simultaneously, at the receiver's end, a Fast Fourier Transform (FFT) is utilized for efficient demodulation of these symbols back into the frequency domain, ensuring accurate data recovery and system performance.

- **Cyclic Prefix (CP):** The inclusion of a Cyclic Prefix (CP) involves adding a guard interval that replicates the last L_cp samples of each transmitted symbol. This additional segment is crucial for mitigating Inter-Symbol Interference (ISI) caused by multipath propagation effects in wireless channels. By doing so, it enhances the robustness of the communication link.

- **Pilot Carriers:** Within the OFDM framework, specific known symbols, referred to as pilot carriers, are strategically inserted at predetermined subcarriers. These pilots serve multiple purposes, including facilitating accurate channel estimation, correcting frequency offsets, and ensuring precise phase tracking during demodulation, ultimately allowing for improved performance and reliability in wireless communication systems.

Image File Streaming

- An image (e.g., JPEG, PNG) is loaded in Python and converted to raw RGB bytes.
- The byte stream is packetized into fixed-length frames (e.g., 60 bytes) via a tagged stream for OFDM framing.
- At the receiver, the raw byte stream is reassembled and converted back to image format using known width and height.

Performance Metrics

- Packet Error Rate (PER): Ratio of lost or corrupted packets to total transmitted packets.
- Bit Error Rate (BER): Ratio of incorrect bits to total bits, computed via bitwise comparison.
- Signal-to-Noise Ratio (SNR): Estimated from pilot symbols or measured noise floor.
- Frequency Offset: Residual CFO estimated and compensated by OFDM receiver.

BLOCKS of GNU RADIO used:

Below is a list of key GNU Radio blocks typically used in the transmitter and receiver flow graphs.

Transmitter Side

Block	Key Parameters	Purpose
File Source	File: img1.raw; Repeat: No	Streams raw image bytes
Stream to Tagged	Length Tag: 60; Key: packet_len	Packetizes byte stream into frames
OFDM Transmitter	FFT: 16; CP: 4; Header BPSK; Payload QPSK; Pilot carriers {-6,-5,5,6}; Occupied carriers {-3..-1,1..3,4}	IFFT, CP insertion, pilot insertion
Multiply Const.	Constant: 0.05	Scales baseband signal amplitude
UHD: USRP Sink	Address: TX Serial; Freq: 2.4 GHz; Rate: 1.024 MSps; Gain: 150	Sends signal to RF front end
QT GUI Time Sink	Points: 1024; Rate: 1.024 MSps	Time-domain waveform display
QT GUI Frequency Sink	FFT Size: 1024; Bandwidth: 1.024 MHz	Frequency-domain spectrum display

Receiver Side

Block	Key Parameters	Purpose
UHD: USRP Source	Address: RX Serial; Freq: 2.4 GHz; Rate: 1.024 MSps; Gain: 150	Captures RF signal
OFDM Receiver	FFT: 16; CP: 4; Header BPSK; Payload QPSK; Tag-Key: rx_len	FFT, CP removal, demodulation, sync
Tag Debug	Key Filter: rx_len; Display: Console	Monitors packet count and lengths
File Sink	File: received_image.raw; Unbuffered: No	Writes reconstructed byte stream to disk
QT GUI Range	ID: freq_offset; Range: ±300 kHz; Step: 10 kHz	Tunes residual frequency offset
QT GUI Time Sink	Points: 1024; Rate: 1.024 MSps	Post-demodulation waveform display

Pre-Lab Setup: -

1. Connect USRP Devices: Attach TX and RX to USB/Ethernet and verify with:
   uhd_find_devices

2. **Prepare Raw Image File:**

   Use any image processing software or tool to convert your desired image (e.g., `.jpg`, `.png`) into a `.raw` binary format.
   Ensure the original image dimensions (width × height) are noted for later reconstruction.

3. Directory Structure:

```
ofdm_image_lab/
├── img1.raw
```

├── ofdm_loopback.grc

└── received_image.raw

reconstruct_image.py

4. Flowgraph Configuration: Open MT24CMN011_Lab5_image_withUSRP.grc and set File Source/Sink paths and USRP serials.

PROCEDURE:

Image Pre-Processing

1. Convert an image of your choice to a .raw RGB file using any standard image conversion tool.
2. Ensure the byte format is RGB and take note of the width and height of the image for reconstruction.
3. Save the file as img1.raw in your working directory.

Transmitter Flowgraph

1. Open MT24CMN011_Lab5_image_withUSRP.grc.
2. Verify the File Source path points to img1.raw.
3. Confirm Stream to Tagged block uses tag key packet_len.
4. Set UHD: USRP Sink to your TX serial and antenna.
5. Check GUI sinks for sample rate and visualization settings.
6. Click **Run**. Observe the time and frequency plots—look for stable OFDM symbols and pilot carriers.

Receiver Flowgraph

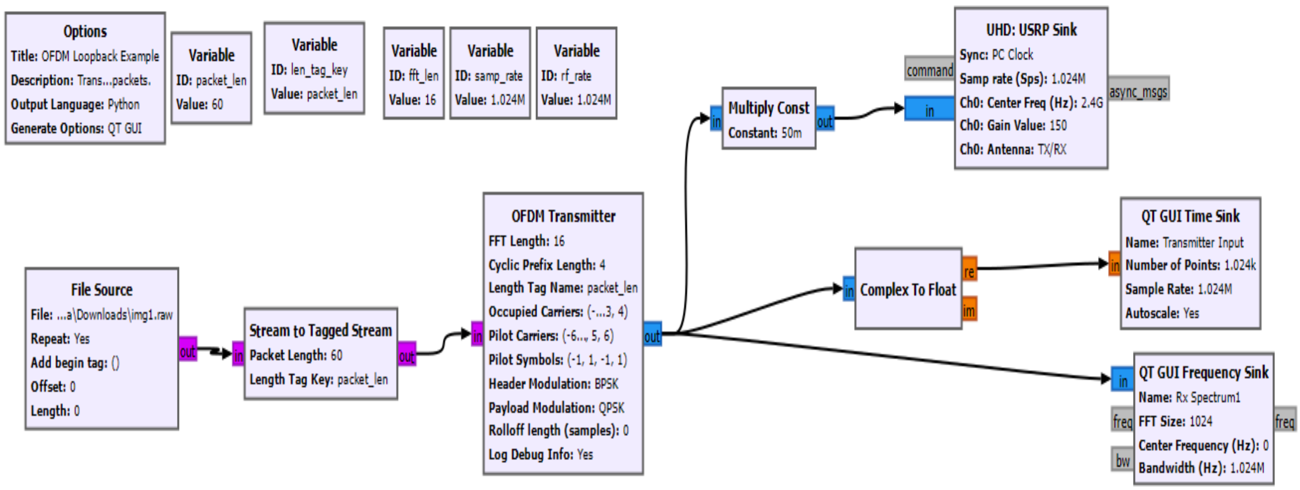
1. In a separate GRC instance, open the same flowgraph or use the included receiver-only graph.
2. Confirm UHD: USRP Source serial and center frequency match TX.
3. Adjust QT GUI Range slider to center the OFDM spectrum.
4. Click **Run**. Monitor Tag Debug for packet counts and Time Sink for waveform.

Post-Processing

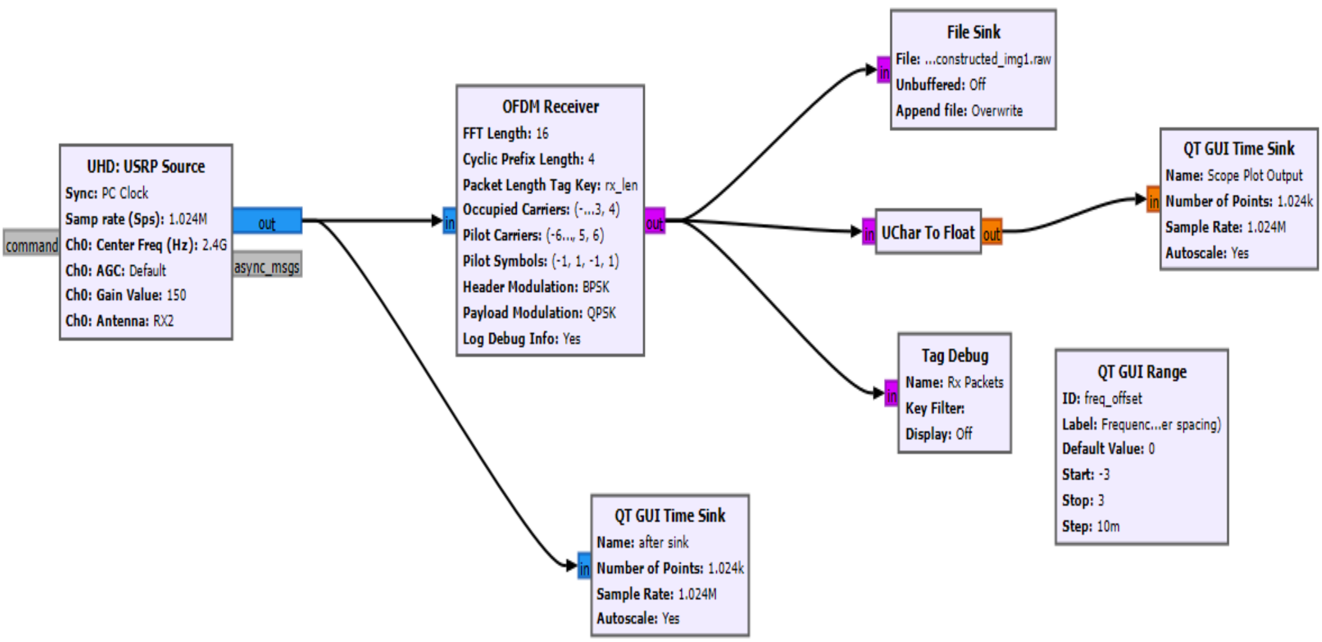
1. After finishing transmission, stop both flowgraphs.
2. Use any standard image tool or utility to convert received_image.raw back into an image using the recorded width × height.
3. Save and open the reconstructed image and visually compare with the original.

BLOCK DIAGRAM: -

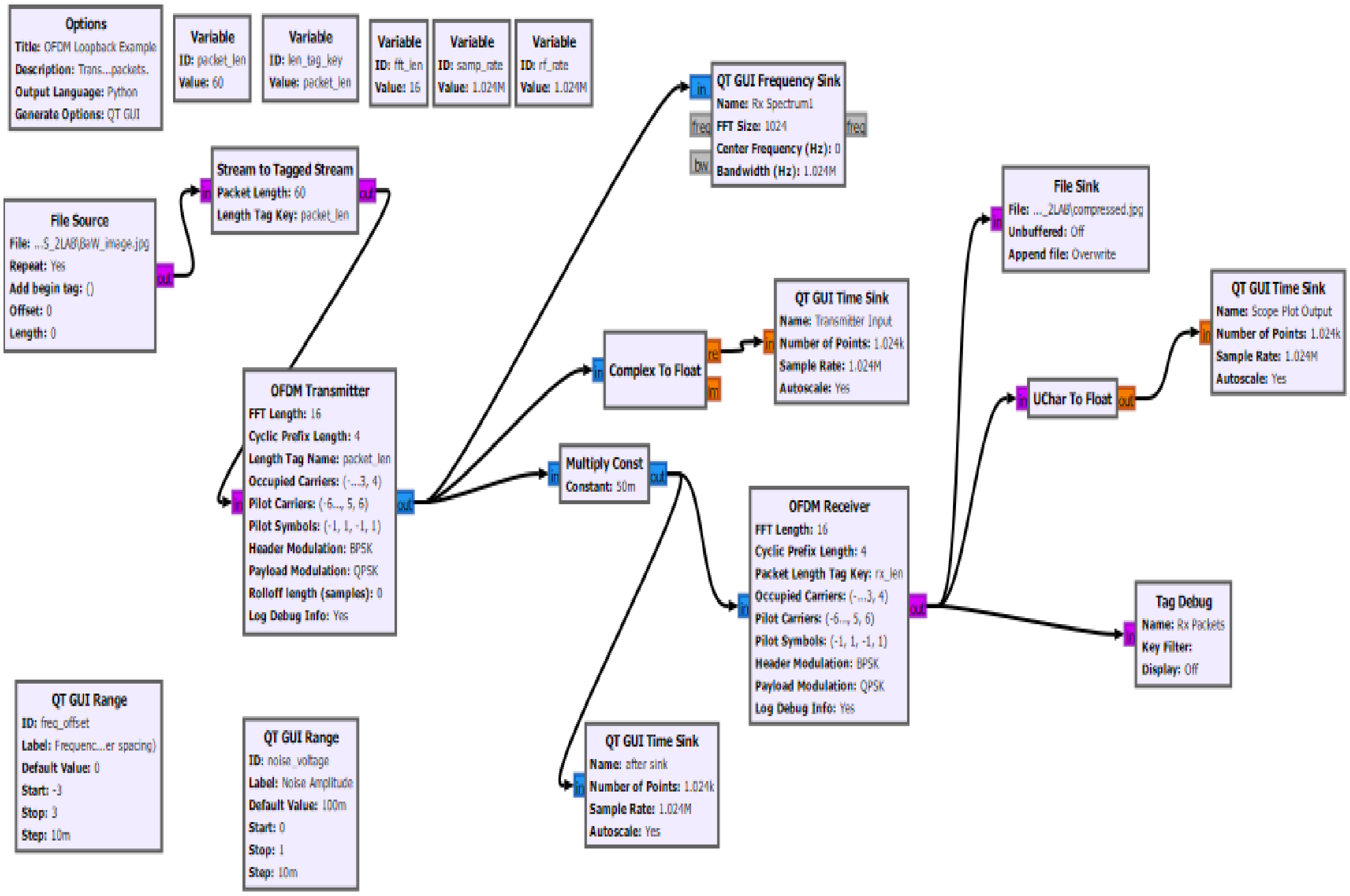
Transmitter Overview With USRP



Receiver Overview With USRP

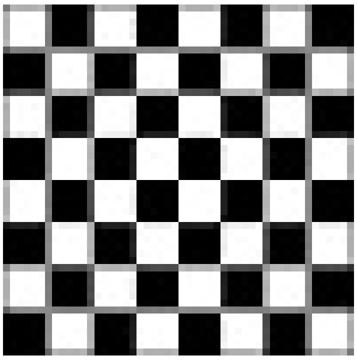


WITHOUT USRP:

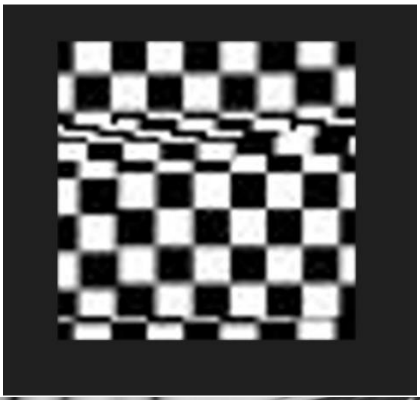


RESULT: -

Original Image or Transmitted Image With USRP: -



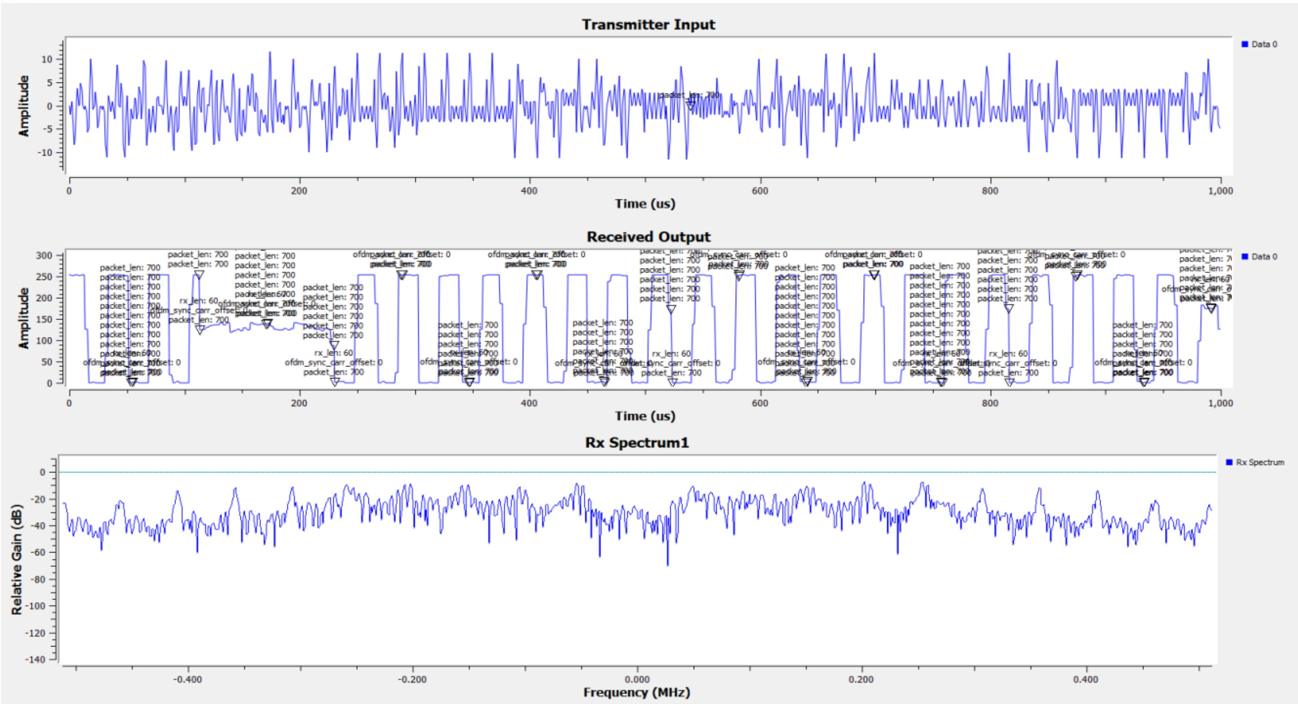
Received Image or Reconstructed Image With USRP: -



Transmitted Image Without USRP: -

Reconstructed Image Without USRP: -

WAVEFORMS



FINAL UNDERSTANDINGS:

1. We used a separate tool to produce .raw file of .jpg image we are transmitting as we observed that .raw image extension is easier to process by file source and file sink and reconstruction happens for .raw only using USRP.

2. The packet length plays an important role in reconstruction. Select an appropriate packet length.

3. The .raw reconstructed file can be converted back to jpg using any standard image tool or utility to convert

CONCLUSION: -

This GNU Radio flowgraph demonstrates an effective end-to-end system for image transmission and reception using Orthogonal Frequency Division Multiplexing (OFDM) with Universal Software Radio Peripheral (USRP) devices. It showcases the processes of packetizing, modulating, and over-the-air transmission, serving as an insightful resource for learning about Software Defined Radio (SDR) and real-time wireless communication techniques.