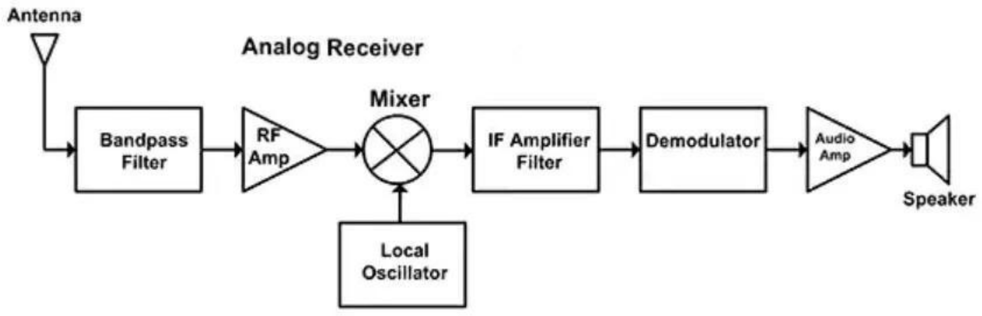
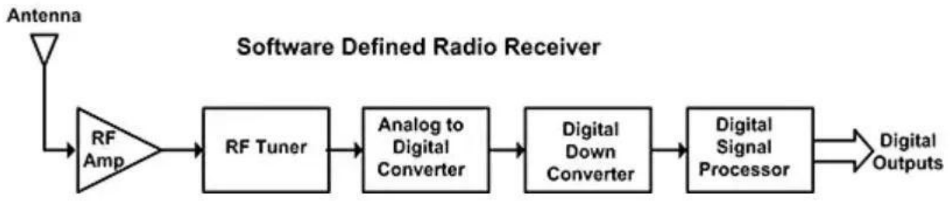
Project: Drone Communication Relay

Aim: To design and implement a drone communication relay system by interconnecting multiple drones in a relaying configuration to extend communication range and ensure robust connectivity in remote or inaccessible regions. The system employs Software Defined Radios (SDRs), specifically USRP (Universal Software Radio Peripheral) devices, to establish and maintain radio frequency communication links between the drones. This setup enables efficient data transmission overcoming the limitations of direct line-of-sight communication.

Software Defined Radio (SDR): Software-defined radio (SDR) is a radio communication system where components that conventionally have been implemented in analog hardware (e.g. mixers, filters, amplifiers, modulators/demodulators) are instead implemented by means of software on a computer or embedded system. A basic SDR system may consist of a computer equipped with a sound card, or other analog-to-digital converter, preceded by some form of RF front end. Significant amounts of signal processing are handed over to the general-purpose processor, rather than being done in special-purpose hardware (electronic circuits). Such a design produces a radio which can receive and transmit widely different radio protocols (sometimes referred to as waveforms) based solely on the software used.





Universal Software Radio Peripheral (USRP): USRP stands for Universal Software Radio Peripheral. It's a range of software-defined radios (SDRs) designed and sold by Ettus Research and its parent company, National Instruments. They can be used for a wide range of applications, including designing, prototyping, and deploying wireless communication systems. They are compatible with various software tools and environments, including LabVIEW, GNU Radio, and MathWorks MATLAB.

The USRP™ B205mini-i delivers a 1×1 SDR/cognitive radio in the size of a business card. With a wide frequency range from 70 MHz to 6 GHz and a user-programmable, industrial-grade Xilinx Spartan-6 XC6SLX150 FPGA, this flexible and compact platform is ideal for both hobbyist and OEM applications.

This USRP is used as a trans receiver for relaying the received messages forward.

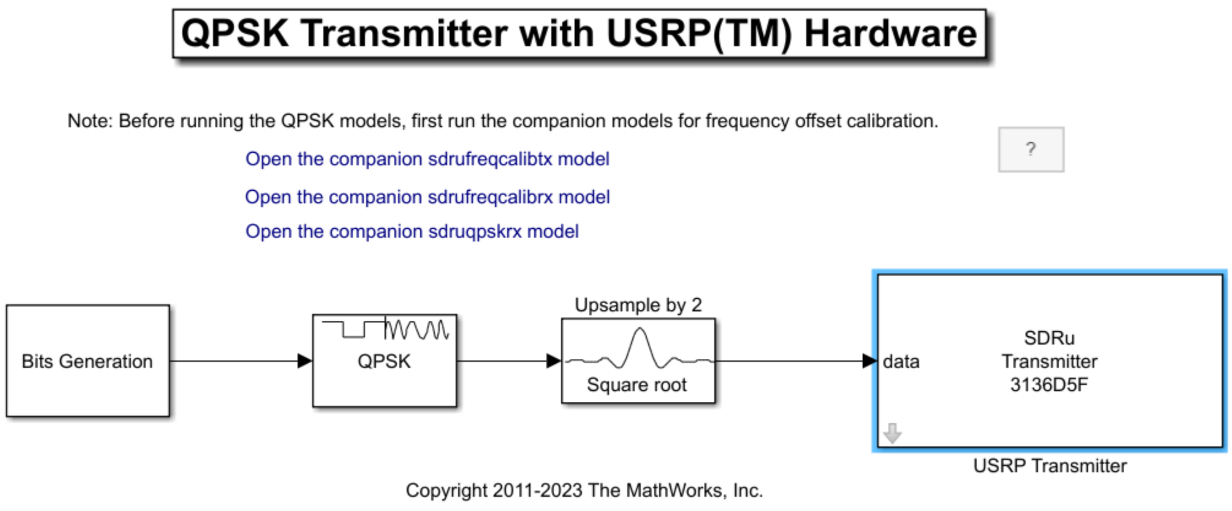
GNU Radio: GNU Radio is a free and open-source software toolkit used for designing, simulating, and implementing radio systems, particularly software-defined radios (SDRs). It provides signal processing blocks that can be combined to create radio applications, either with or without external RF hardware.

GNURadio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

Essentially, it allows users to build and experiment with radio technology using software instead of dedicated hardware. We were able to transmit and receive analog signals using GNU Radio and USRP at centre frequency of 5.2 GHz. We also observed how changing the gain of the transmitter and the receiver effects the SNR of the transmitting signal.

But transmission and reception of digital data like text, image, video etc posed challenges in the current version of GNU Radio due to depreciation of 'Packet Encoder Block' after version 3.7 due to the block's inherent problems, including random data dropping and being difficult to fix. All the resources available on the internet including research papers and video links were based on earlier versions of GNU Radio using the Packet Encoder block which are no longer available for use. There are some tutorials available on GNU Radio regarding digital transmission without using packet encoder but when we tried to implement those in our systems, we encountered lots of errors in the python codes with some missing data as well. We posted our issues on the Global Forum for some assistance from some experienced users but none were able to clarify it. We, therefore shifted to MATALB.

MATLAB: In MATLAB Simulink we first transmitted text signal from one computer to the other using USRP B210 and B205 mini. The transmission was done on a centre frequency of 5.2GHz. Two different Simulink models were used, one for transmission and one for reception. In order to use the model, we have to install Communication Toolbox Support Package for USRP Radio.


QPSK Transmitter with USRP(TM) Hardware

Note: Before running the QPSK models, first run the companion models for frequency offset calibration.

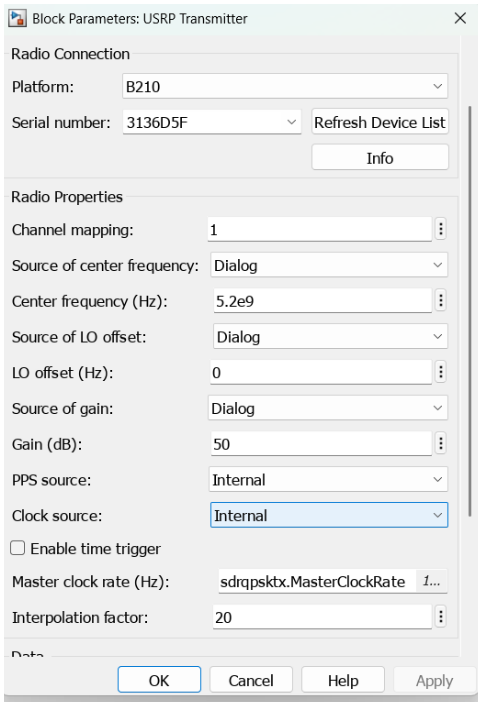
Open the companion sdrufreqcalibtx model

Open the companion sdrufreqcalibrx model

Open the companion sdruqpskrx model

Copyright 2011-2023 The MathWorks, Inc.

Here is the block diagram of the transmitter Using Simulink. The transmission has been done using QPSK modulation scheme. QPSK is a digital modulation technique that transmits two bits of data at a time by using four distinct phases of a carrier wave.



Block Parameters: USRP Transmitter

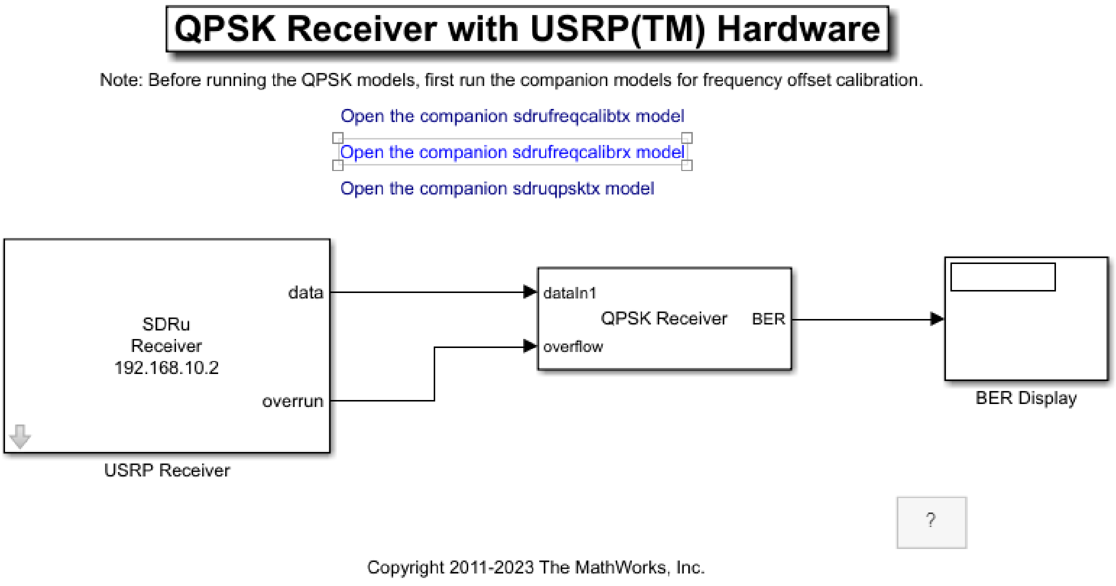
Radio Connection

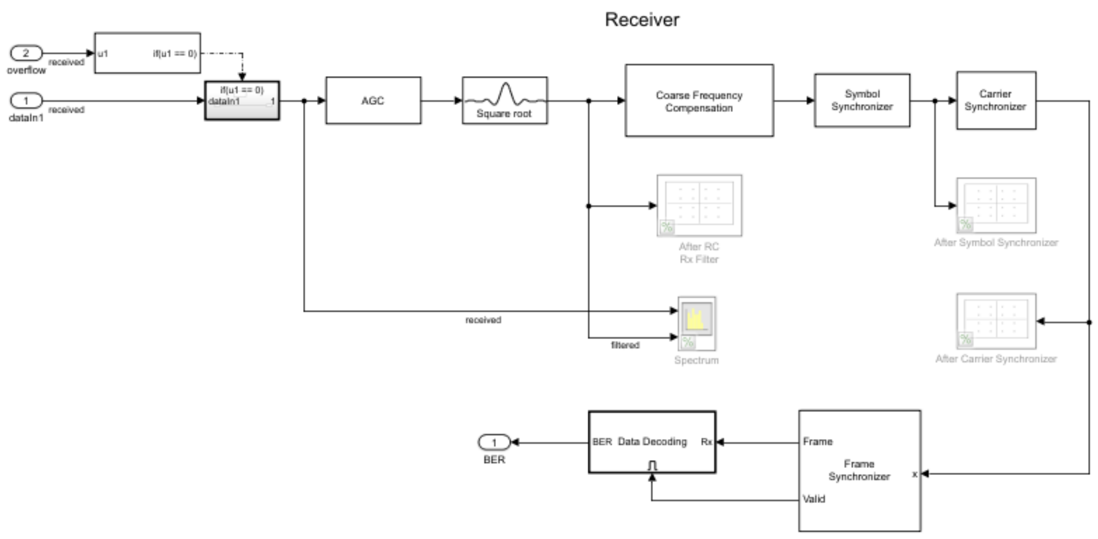
Platform:	B210
Serial number:	3136D5F

Radio Properties

Channel mapping:	1
Source of center frequency:	Dialog
Center frequency (Hz):	5.2e9
Source of LO offset:	Dialog
LO offset (Hz):	0
Source of gain:	Dialog
Gain (dB):	50
PPS source:	Internal
Clock source:	Internal
Enable time trigger	☐
Master clock rate (Hz):	sdrqpsktx.MasterClockRate
Interpolation factor:	20

Refresh Device List
Info
OK
Cancel
Help
Apply

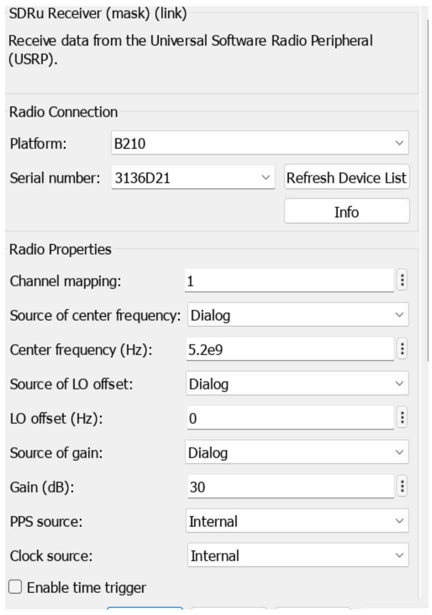

QPSK Receiver with USRP(TM) Hardware
Note: Before running the QPSK models, first run the companion models for frequency offset calibration.
Open the companion sdrufreqcalibtx model
Open the companion sdrufreqcalibrx model
Open the companion sdruqpsktx model
SDRu Receiver 192.168.10.2
data
overrun
USRP Receiver
dataIn1
overflow
QPSK Receiver
BER
BER Display
?
Copyright 2011-2023 The MathWorks, Inc.



This is the Simulink model for QPSK Receiver using USRP. The second figure is the internal working of the QPSK Receiver block in the first diagram. Using this system, we were able to transmit 100 'Hello World' messages at a time.

The gain on both sides had to be kept optimal. On applying a very large gain (~80 dB) resulted in a lot of noise in the signal decreasing the SNR because of which no message was received on the receiver.

By making suitable changes we can send any message, but to calculate bit error rate reference message has to be given in the receiver block's MATLAB code as well.


SDRu Receiver (mask) (link)

Receive data from the Universal Software Radio Peripheral (USRP).

Radio Connection

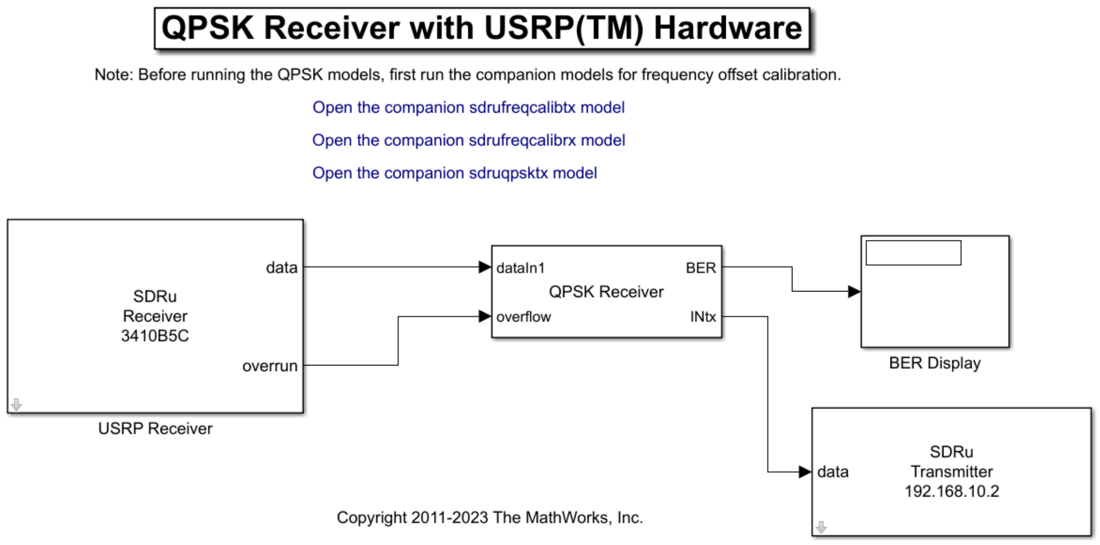
Platform:	B210
Serial number:	3136D21

Radio Properties

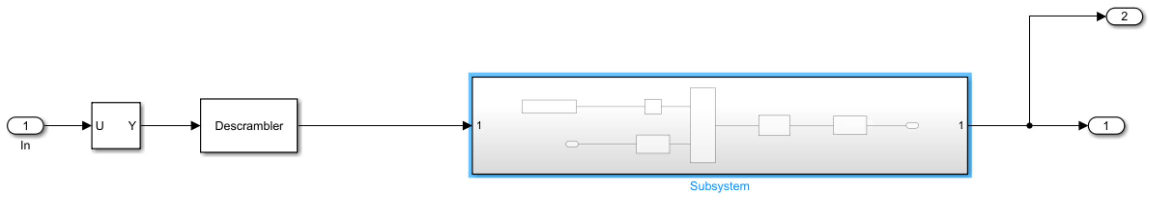
Channel mapping:	1
Source of center frequency:	Dialog
Center frequency (Hz):	5.2e9
Source of LO offset:	Dialog
LO offset (Hz):	0
Source of gain:	Dialog
Gain (dB):	30
PPS source:	Internal
Clock source:	Internal

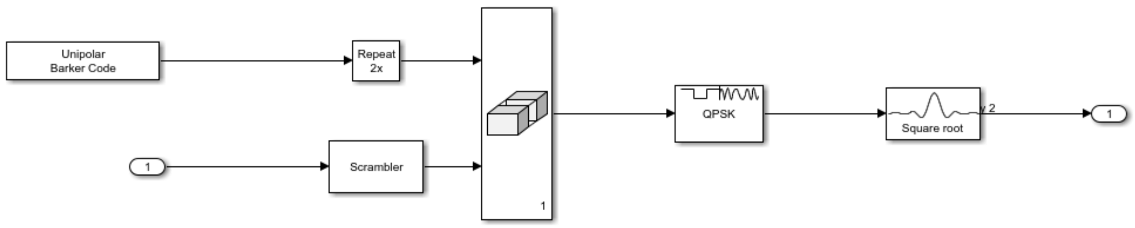
☐ Enable time trigger

Relay system: After transmitting text, we made one USRP as a transceiver to relay the received message signal to the third device. For that we made changes in the above receiver model so that it can relay the received message.


QPSK Receiver with USRP(TM) Hardware
Note: Before running the QPSK models, first run the companion models for frequency offset calibration.
Open the companion sdrufreqcalibtx model
Open the companion sdrufreqcalibrx model
Open the companion sdruqpsktx model
SDRu Receiver 3410B5C
data
overrun
USRP Receiver
dataIn1
overflow
QPSK Receiver
BER
INtx
BER Display
data
SDRu Transmitter 192.168.10.2
Copyright 2011-2023 The MathWorks, Inc.

The signal is received at 5.2 GHz by the transceiver and is further transmitted at 915 MHz to the final receiver.





These changes were made in the ‘Data Decoding’ subsystem in QPSK Receiver subsystem. The second figure shows the subsystem which has been added in front of the descrambler. A descrambler reverses the scrambling process, restoring the original data stream from a scrambled version. Scrambling is used to make data more robust against errors and ensure a more consistent bit stream. This subsystem again modulates the message signal using QPSK scheme and sends it to the transmitter to relay the signal.

Video transmission approach: After text transmission we tried video transmission. Transmitting video data over a SDR requires careful handling of the video format. Video files are 3D data structures, composed of dimensions representing height, width, and colour channels (RGB). However, for transmission over a communication system—especially using hardware like USRP—the data must be serialized, i.e., converted into a 1D or 2D format such as a column vector, so that the QPSK modulation scheme blocks can be used efficiently and further the video can be relayed on USRP.

In MATLAB Simulink, while there are video processing blocks available, they are primarily designed for simulation or display purposes rather than real-time wireless transmission. When attempting to use these blocks directly for transmission, several issues arise:

- **Improper Reshaping:** The 3D video data was not correctly reshaped into a serial stream format required for modulation and transmission.
- **Unsupported Block Behaviour:** The Simulink video blocks did not consistently support or maintain the structure and timing needed for USRP communication, resulting in frame synchronization errors or incomplete frame delivery.
- **Abrupt Output Post-Modulation:** After modulation, the video data received was abrupt, distorted, or incomplete, indicating a mismatch in how the data was prepared and how the USRP expected to process it.

These issues highlight the challenge of using Simulink for real-time video transmission without additional preprocessing steps to reshape the video, buffer frames correctly, and ensure synchronization.

DragonOS: One way to transmit video using USRP is through DragonOS. DragonOS is a specialized Linux-based operating system pre-configured for SDR and signal analysis tasks. It's built on top of Ubuntu (or Lubuntu for lighter versions) and comes with a wide range of open-source SDR and RF tools pre-installed and pre-configured. QradioLink is one such application included in DragonOS. QRadioLink enables digital voice and data communication using SDR.

It's a software modem that integrates with various SDR hardware (like USRP, HackRF, or RTL-SDR) and is designed to function as a two-way radio communication system.

The issue we faced is that USRP didn't get connected to DragonOS. Instead of connecting to DragonOS (set through virtual machine) USRP was detected only in bootloader mode, not in its fully functional mode with firmware loaded. When a USRP device powers up, it may show up as a USB device in bootloader mode and the virtual machine was not able to capture the connection of USRP.

Also, while trying to run QRadioLink executable file it kept halting the execution with segmentation fault error. This can be discovered more for real time video transmission.

Conclusion: In this project, we successfully designed and implemented a drone communication relay system using USRP-based Software Defined Radios to enable extended-range wireless communication. We began by establishing text transmission between two devices in MATLAB Simulink using QPSK modulation. This was further enhanced into a relay system where an intermediate USRP acted as a transceiver, forwarding the received message to a third device, effectively demonstrating a multi-hop communication setup.

Attempts to transmit video over USRP in Simulink highlighted critical challenges due to the 3D nature of video data and the limitations of Simulink's video processing blocks in real-time RF transmission. These challenges emphasize the need for additional preprocessing and custom handling of video data.

We also explored DragonOS and QRadioLink as alternative platforms for real-time communication. While DragonOS showed promise due to its pre-configured SDR environment, technical issues such as USRP connectivity within a virtual machine and segmentation faults during QRadioLink execution limited its utility during this project.

References:

- https://www.gnuradio.org/
- https://in.mathworks.com/help/comm/usrpradio/ug/qpsk-transmitter-with-usrp-hardware-in-simulink.html

- https://in.mathworks.com/help/comm/usrpradio/ug/qpsk-receiver-with-usrp-hardware-in-simulink.html
- https://www.youtube.com/watch?v=4uMLqRegjSQ