
# University of Utah
# Summer Internship/Senior Project 2021/2022
# Scott Talbot Ph.D. – May 4, 2021

# Presentation Outline

- Project Introduction
  - Hardware
  - Software
  - Modem Design
- PRBS
- Task #1
- Introduction to Digital Communications
  - Transmission of bits
  - Modulation methods
  - Signal Power, Noise Power, SNR
- Task #2
- To be continued…

# Internship/Senior Project Breakdown

- The Zeta Associates sponsored project consists of two pieces
  1. Build an OFDM modem
  2. Use the OFDM modem to run tests on POWDER
- The POWDER portion will likely not happen until Fall

- OFDM – Orthogonal Frequency Division Multiplexing (Quite a mouthful. More on this later.)
- Modem – stands for modulator/demodulator, which is basically a communication device that can transmit and receive data.

# Project Tools

- Hardware
  - System76 Galago Pro laptop running Pop!_OS/Ubuntu Linux
  - USRP B205mini

- Software
  - Linux
  - Python
  - ~~GNU Radio~~
  - Other packages as the need arises

# Important!!!

- When you get the USRP
  - It is for your convenience so that you do not have to wait for resources on POWDER
  - Don't just start transmitting stuff – In general, it's illegal
  - Don't receive certain signals – It might be illegal
    - FM/TV broadcasts are safe
  - Take care with the RF cables
    - They don't like to be bent at sharp angles

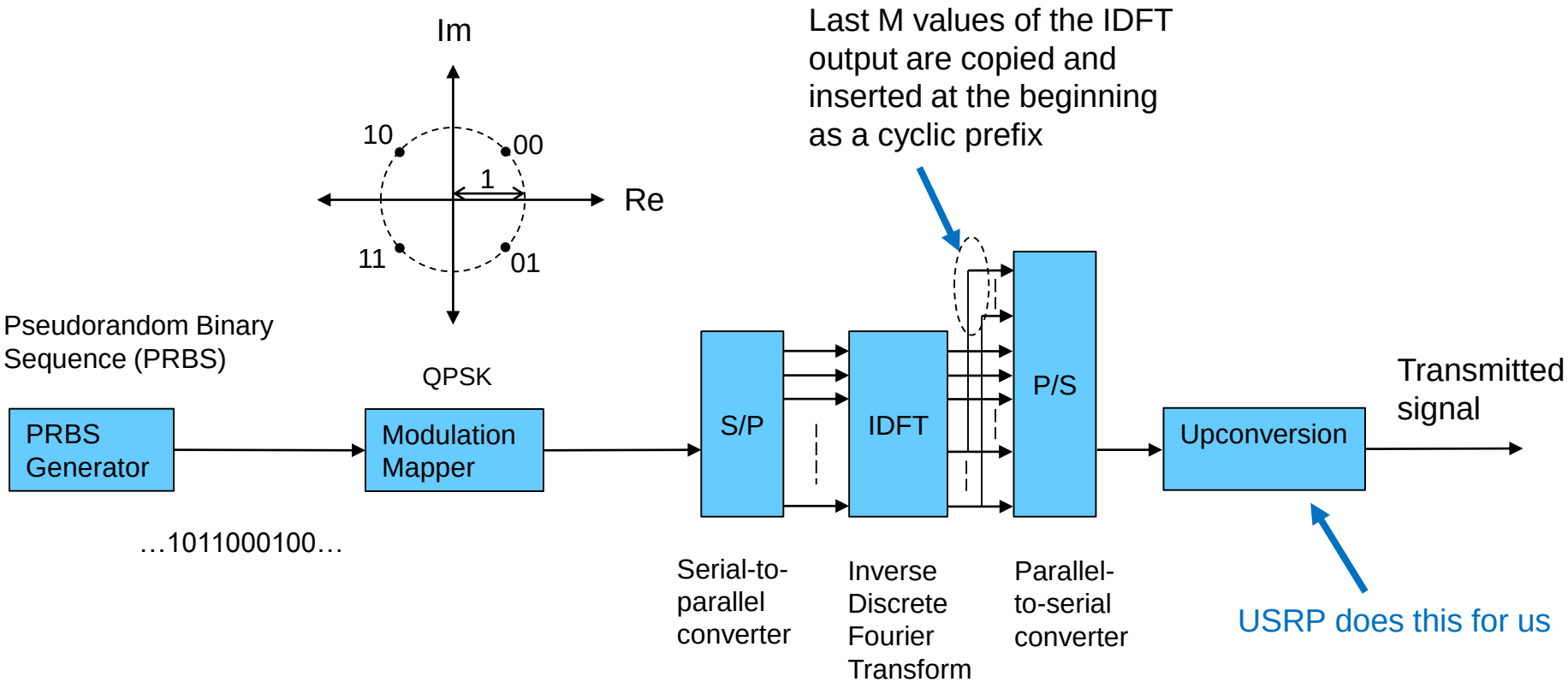
# Computing Resources

- Linux:
  - You'll want to learn the command line interface (CLI)
  - https://ryanstutorials.net/linuxtutorial/
- Python:
  - https://automatetheboringstuff.com/
- ~~GNU Radio~~
  - ~~https://wiki.gnuradio.org/index.php/Tutorials~~
  - After a deep-dive into GNU Radio, I decided that we didn't need yet another framework to learn. We will control the B205-mini directly from python.
  - We may use it to build a simple spectrum analyzer

- You can always ask me

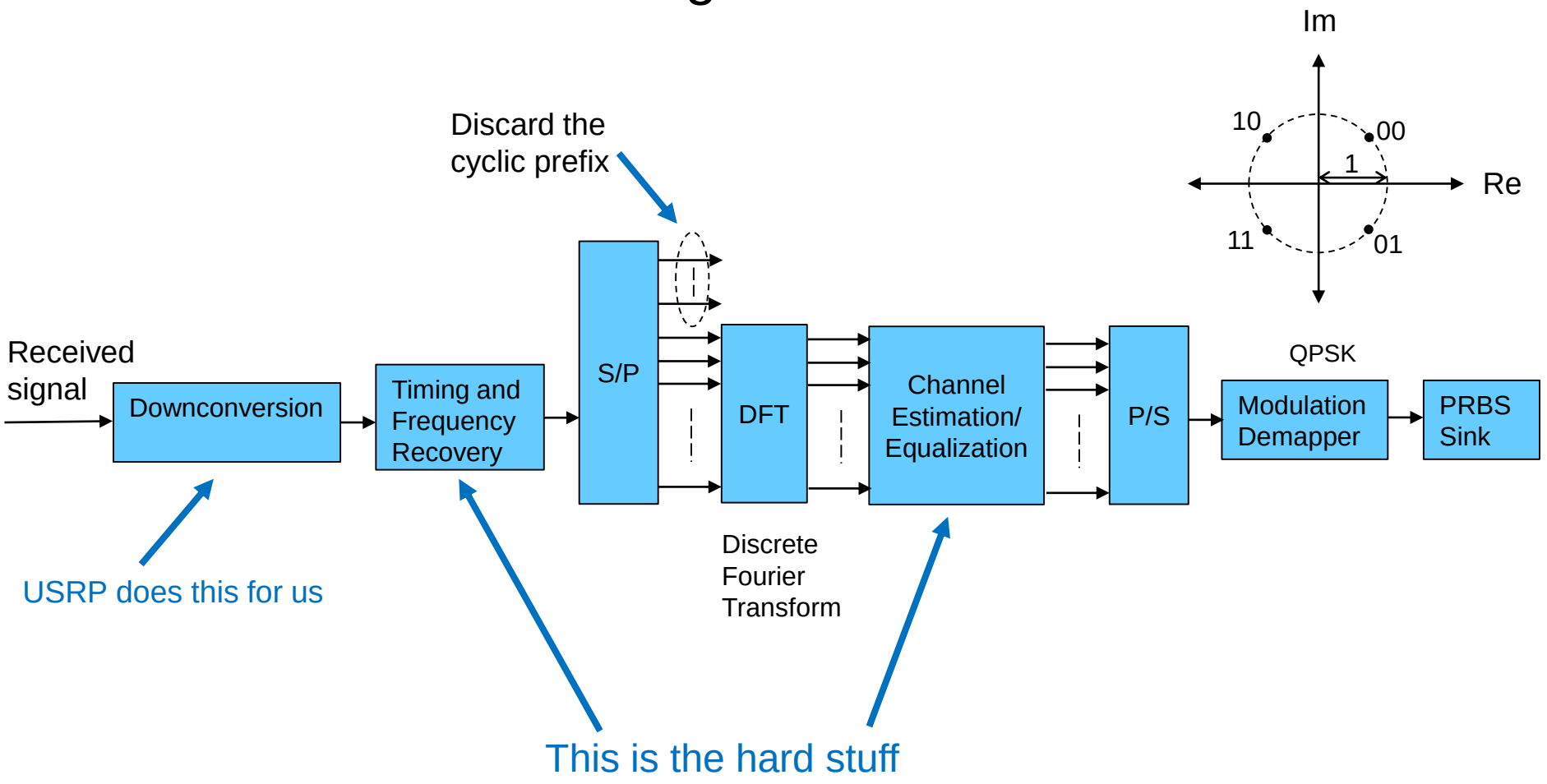
# OFDM Modem - Modulator

- OFDM Modem Design



# OFDM Modem - Demodulator

- OFDM Modem Design



# Modem Details

- We'll go over the details as we progress
- Probable order of development
  1. PRBS generator and PRBS sink
  2. Modulation Mapper/Modulation Demapper
  3. IDFT/DFT
  4. Cyclic prefix insertion/cyclic prefix removal
  5. Timing recovery
  6. Frequency recovery
  7. Channel estimation/equalization

# Presentation Outline

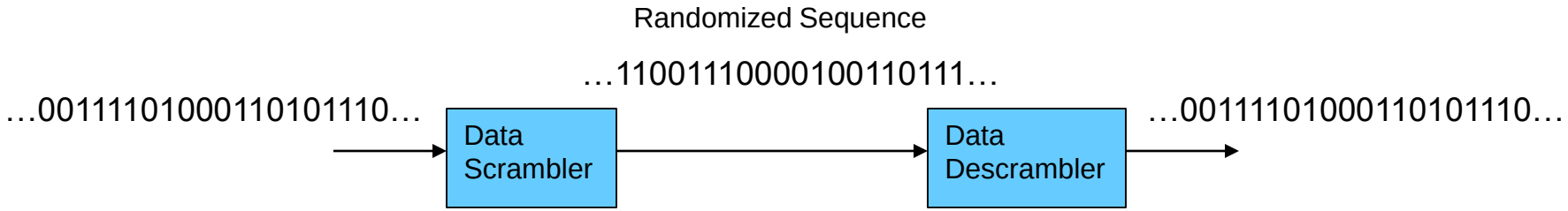
- Project Introduction
  - Hardware
  - Software
  - Modem Design
- PRBS
- Task #1
- Introduction to Digital Communications
  - Transmission of bits
  - Modulation methods
  - Signal Power, Noise Power, SNR
- Task #2
- To be continued…

# PRBS

- Pseudo-random bit stream (PRBS)
  - It's hard to get machines to generate truly random numbers (or bit sequences)
  - Eventually, the numbers (or bits) repeat
- Why are we transmitting a random bit sequence?
  - It's just for testing purposes
  - We can always swap out the PRBS with a "real" data payload later

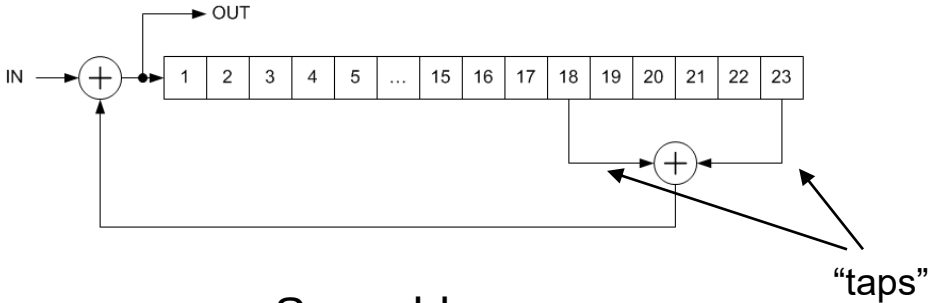
# Data Scramblers

- What's a data scrambler?
  - Takes an input bit stream and randomizes it
  - The many purposes of data scramblers are beyond the scope of this presentation and/or the project
  - We are going to use data scramblers descramblers to generate our PRBS

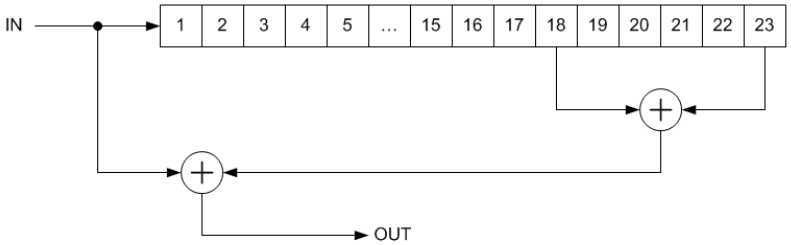


# Multiplicative Data Scramblers

- Multiplicative data scramblers are interesting because the descramblers are self-synchronizing
  - The descrambler doesn't have to do anything other than take in the input and it will properly descramble the data
- The general structure is as shown here:




Scrambler

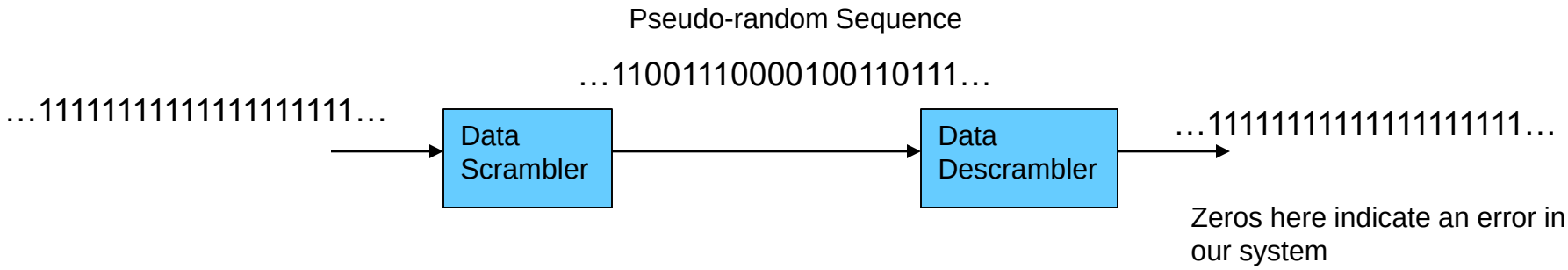



Descrambler

- See https://en.wikipedia.org/wiki/Scrambler

# Scramblers/PRBS

- If we feed all ones into a scrambler with properly chosen taps, we will get a PRBS sequence that repeats every $2^N - 1$ bits, where N is the length of the scrambler shift register
- Convenient for testing a communications system, because we transmit “random” data and get all ones at the output
  - Output bits that are zeros indicate system errors



# Presentation Outline

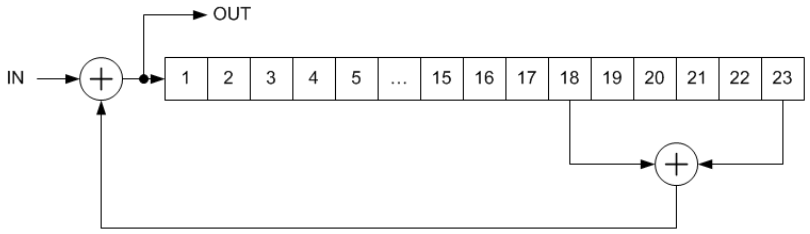
- Project Introduction
  - Hardware
  - Software
  - Modem Design
- PRBS
- Task #1
- Introduction to Digital Communications
  - Transmission of bits
  - Modulation methods
  - Signal Power, Noise Power, SNR
- Task #2
- To be continued…

# Task #1

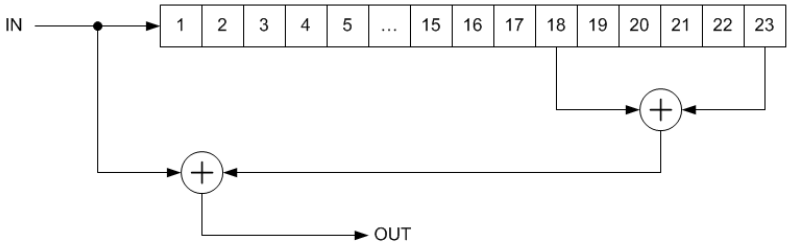
- Write a python implementation of a multiplicative scrambler and a multiplicative descrambler.
- Test the scrambler/descrambler by inserting all ones into the scrambler and seeing if all ones come out of the descrambler.
- Depending on the length of the shift register, there may be errors (some zeros) at the initialization of the descrambler.  This is fine.

# Task #1

- For now, implementing the below diagrams is fine
- We might make it more generic or choose another shift register length with different taps later




Scrambler




Descrambler

# Presentation Outline

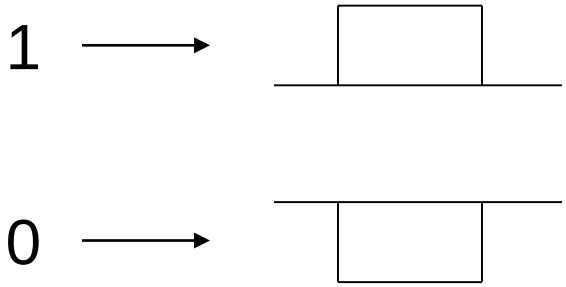
- Project Introduction
  - Hardware
  - Software
  - Modem Design
- PRBS
- Task #1
- Introduction to Digital Communications
  - Transmission of bits
  - Modulation methods
  - Signal Power, Noise Power, SNR
- Task #2
- To be continued…

# Digital Communications

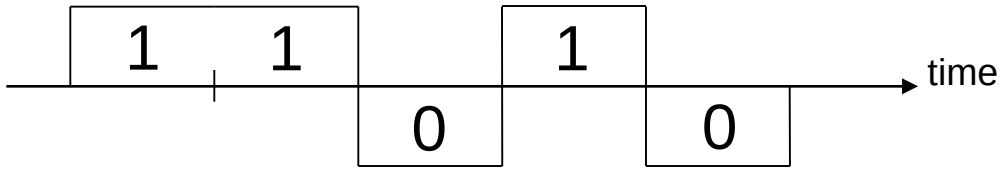
- What is digital communications?
  - In general, digital communications is a wired or wireless communications method that uses an analog signal of a finite duration to convey a finite number of data bits.
  - A series of these finite duration analog signals allows a user to send a message of bits of arbitrary length.

# An Example

- Let the binary digits 1 and 0 be mapped to the analog signal/pulses shown below.




- To send the sequence 11010, would look like the signal below.

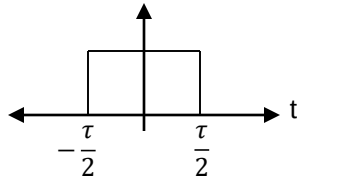


# Rect Pulses

- Is that it? Just use rectangular pulses to transmit bits of information?

No!

- Won't focus on this for now… but…
- Remember Fourier Transforms?



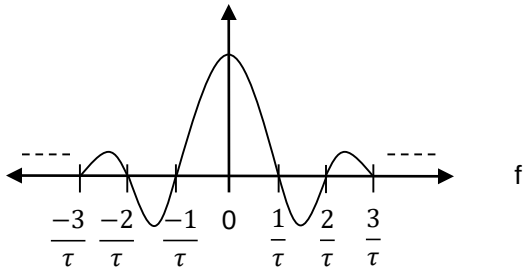

$$\Pi(t) = \begin{cases} 1, & |t| < \tau/2 \\ 0, & \text{otherwise} \end{cases}$$

Fourier Transform

$\mathcal{F}$

$\mathcal{F}^{-1}$

Inverse
Fourier Transform




$$\tau \times \text{sinc}(f\tau) = \frac{\sin(\pi f \tau)}{\pi f \tau}$$

Continues in both directions, slowly decaying, forever

Undesirable

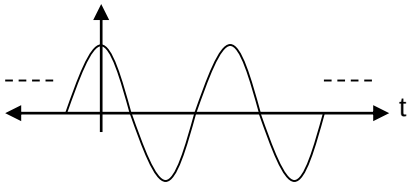
# Rect Pulses Continued

- As I said, we won't focus on this for now, but…
  - See section 3.2 "Pulse-Shape Designs for Band-Limited Communications" in *Signal Processing Techniques for Software Radios, 2nd Edition* by Prof. Farhang.

# Sinusoids

- What types of signals should we use for digital communications?
- Let's take a closer look at sinusoids.

- Do you remember this from your signal processing class?




$\cos(2\pi f_0 t)$

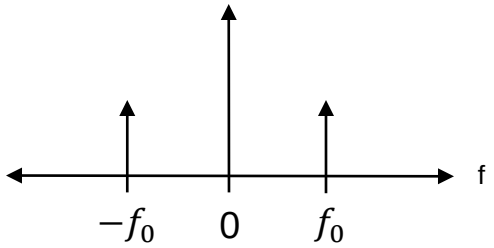
Continues forever in both directions

Fourier Transform

$\mathcal{F}$

$\mathcal{F}^{-1}$

Inverse Fourier Transform




$\frac{1}{1}[\delta(f - f_0) + \delta(f + f_0)]$

Continues in both directions, slowly decaying, forever

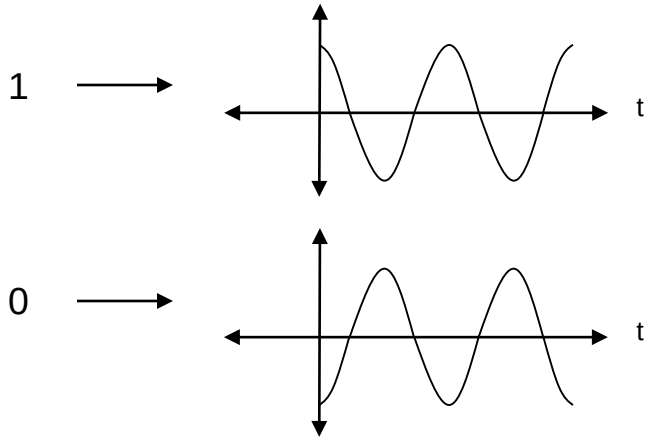
# Conveying Information with Sinusoids

- How do we convey information (bits) using sinusoids?
- There are a few basic ways to do this:
  - Input bits determine sinusoid amplitude
  - Input bits determine sinusoid phase
  - Input bits determine sinusoid frequency

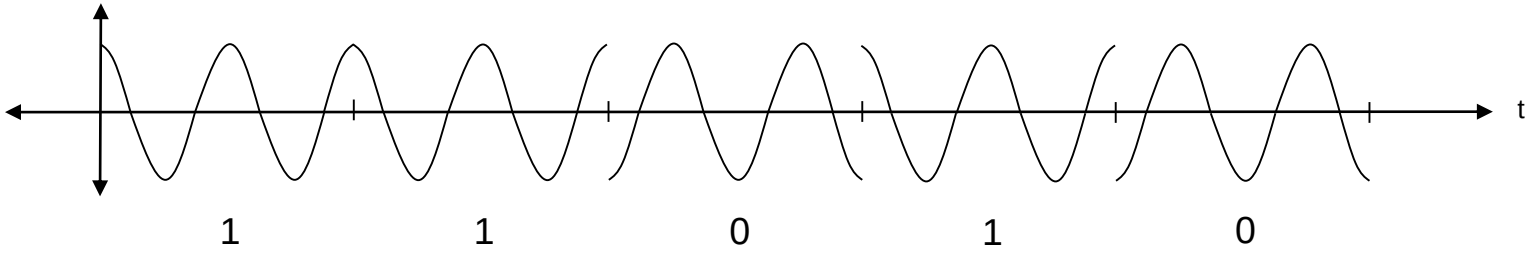
# Pulse-Amplitude Modulation

- In pulse-amplitude modulation (PAM) we adjust the amplitude of the sinusoid based on bit input
- Example: Let's convey one bit of information per sinusoid segment (2PAM).

| Bit Input | Sinusoid Amplitude |
|---|---|
| 1 | 1 |
| 0 | -1 |



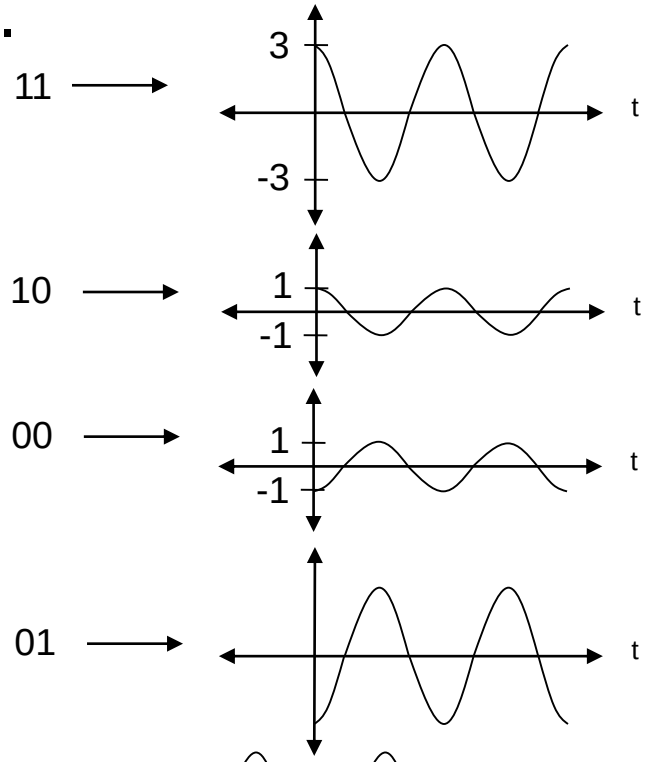

Transmitted 11010 sequence



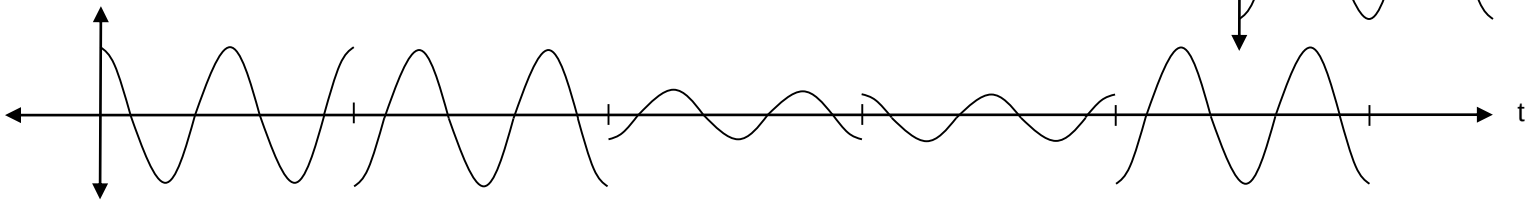
# 4PAM

- Example: Let's convey two bits of information per sinusoid segment (4PAM).

| Bit Input | Sinusoid Amplitude |
|---|---|
| 11 | 3 |
| 10 | 1 |
| 00 | -1 |
| 01 | -3 |



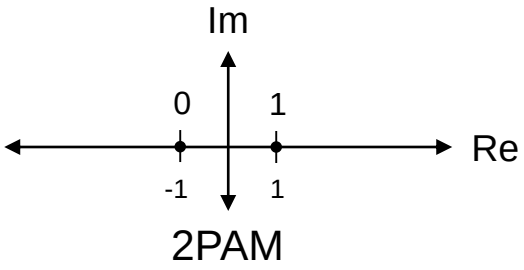

Transmitted 1101001001 sequence



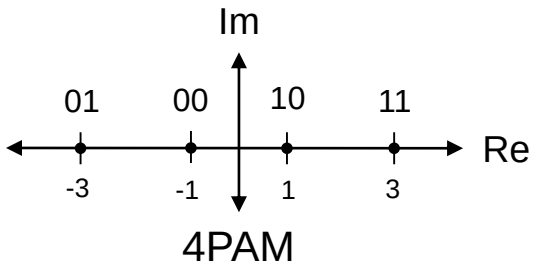

11 01 00 10 01

# Constellation Maps

- Each sinusoid segment is multiplied by a “symbol” the represents the underlying bits
- We can take the set of symbols for a modulation type and display those as dots on a plot.
- The set of dots is called a constellation map



2PAM



4PAM

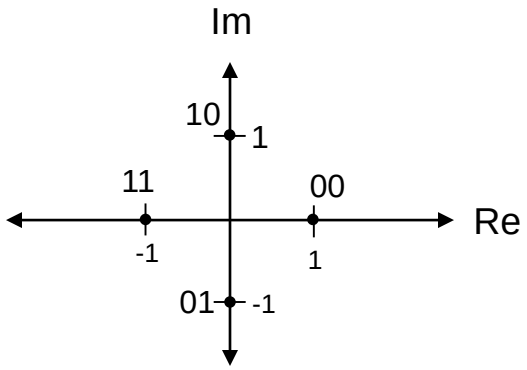
Re - Real

Im - Imaginary

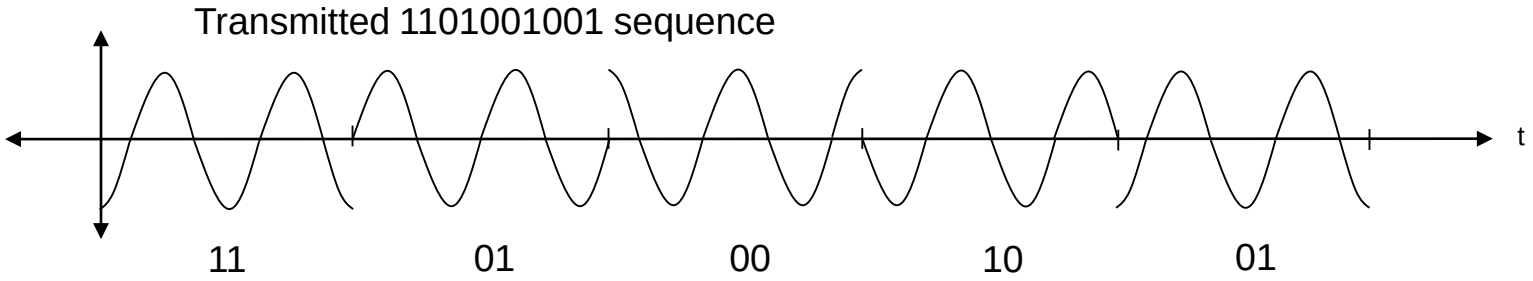
# Phase-Shift Keying

- With phase-shift keying (PSK), we adjust the phase of the sinusoid based on the bit input
- Example: Let's map two bits of information on four phases

| Bit Input | Sinusoid Phase (deg) |
|---|---|
| 11 | 180 |
| 10 | 90 |
| 00 | 0 |
| 01 | 270 |

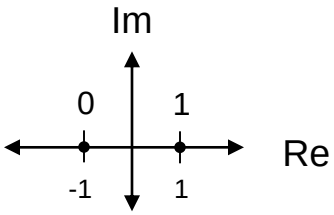



QPSK (4PSK)

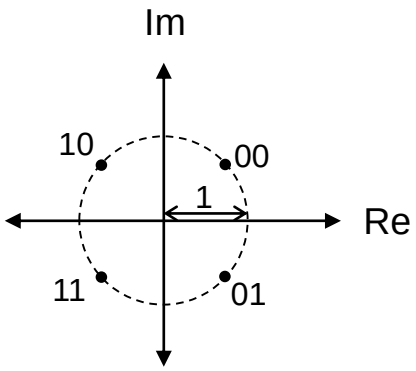


# Other Flavors of PSK

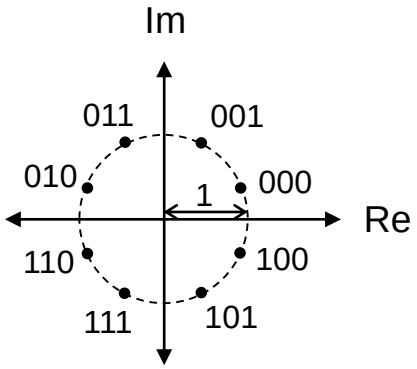
- The symbols of a PSK constellation are, in general, spaced evenly on the unit circle in the complex plane
- Note that BPSK (2PSK) is the same as 2PAM
- QPSK symbols more commonly take the phases shown on this slide as opposed to the previous slide



BPSK



QPSK (4PSK)



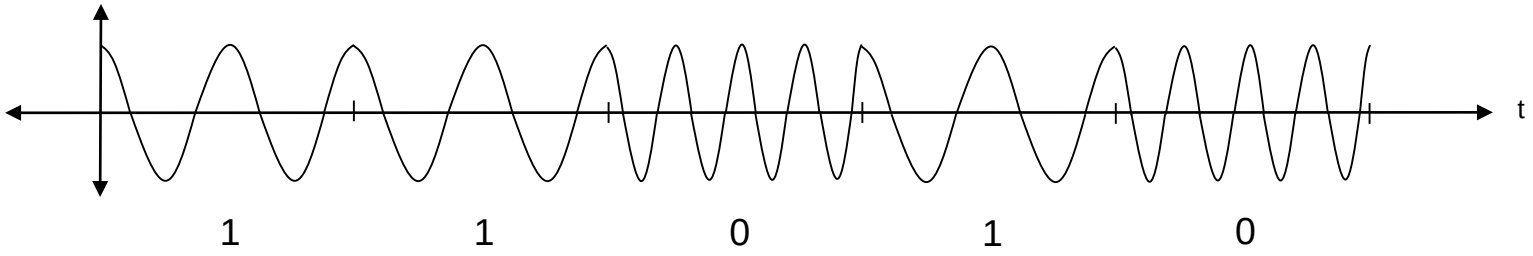
8PSK

# Frequency Shift Keying

- With frequency-shift keying (FSK), we adjust the frequency of the sinusoid based on the bit input

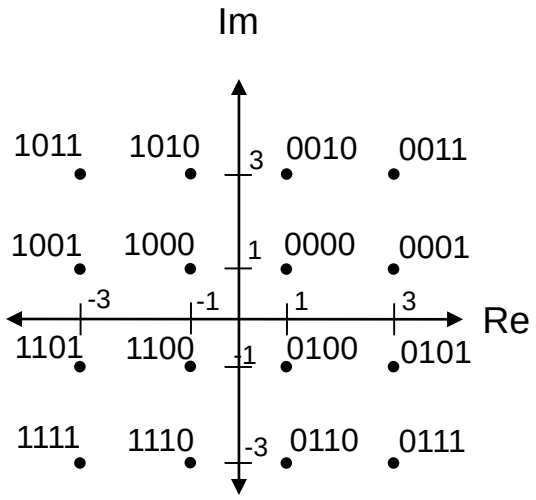
| Bit Input | Sinusoid Frequency |
|---|---|
| 1 | $f_1$ |
| 0 | $f_0$ |

Transmitted 11010 sequence



# Quadrature Amplitude Modulation

- It is possible to adjust various parameters (amplitude, phase, frequency) simultaneously
- Quadrature Amplitude Modulation (QAM) adjust phase and amplitude depending on input bits
- It's like PAM in 2D, real and imaginary



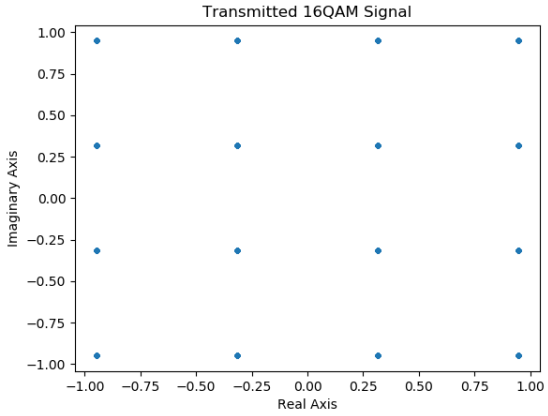

16QAM

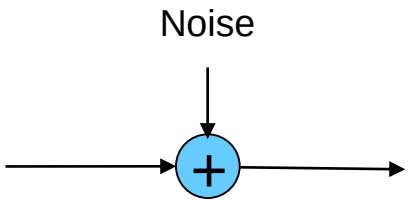
# My Experience

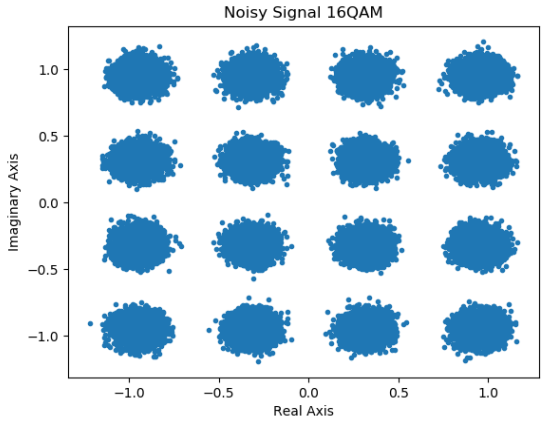
- What have I seen in “real” digital communication systems?
  - BPSK, QPSK, 8PSK
  - BFSK, 4FSK, and higher
  - 8QAM, 16QAM, 64QAM, 256QAM, 1024QAM
  - And a bunch of other more complicated varieties

# Noise

- This seems fairly easy so far. What problems do we encounter in a real system?
  - There are a lot of problems, but we will focus on noise for now
- Let's say we are sending a "long" message of composed of many, many 16QAM data symbols
- We will see noise on the symbols at the receiver

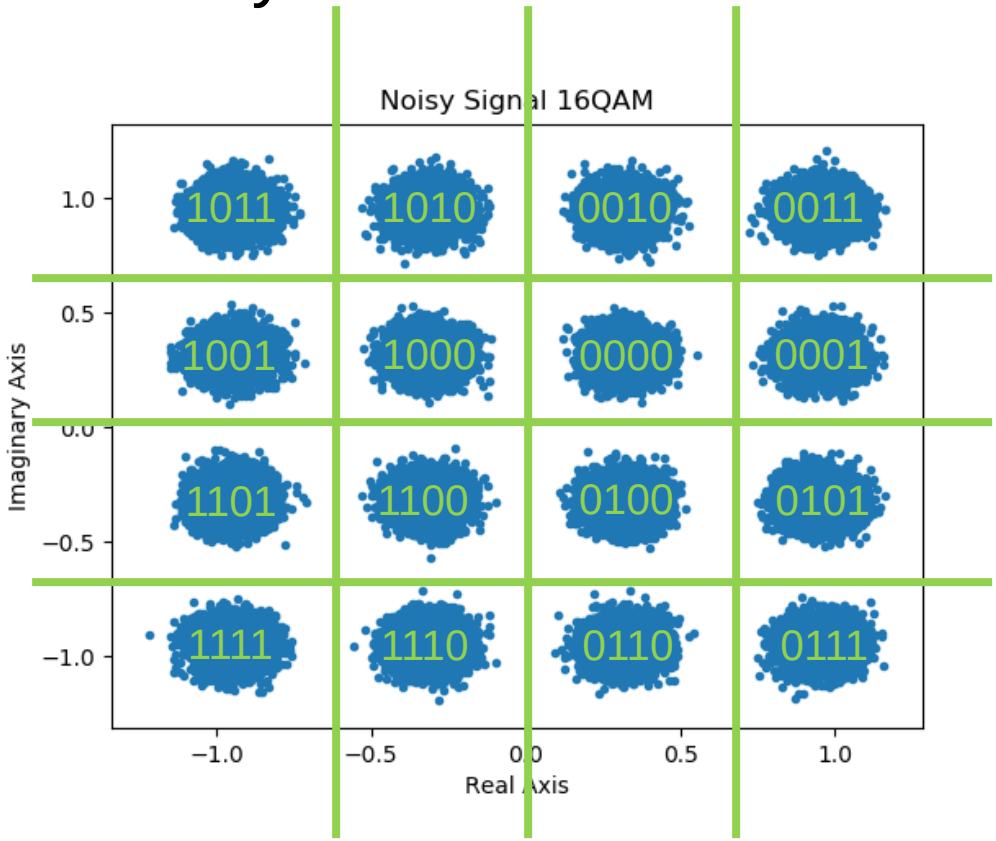


# Noise

- You can convert symbols back to bits by defining decision regions and outputting the appropriate bits for each symbol



# Power and Noise

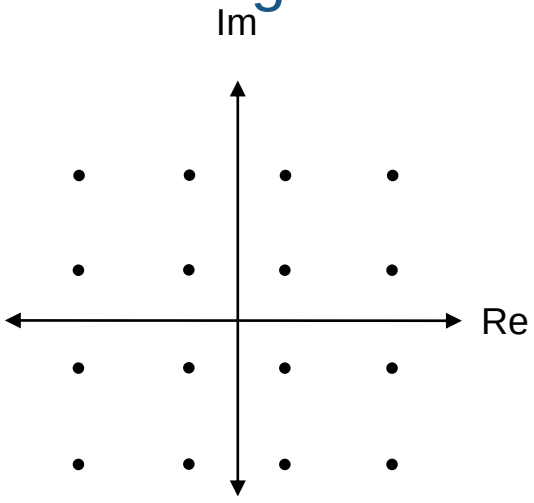
- If you can transmit 4, 6, or even 10 bits at a time, why would you every transmit just one lousy bit at a time?

Power limitations and noise

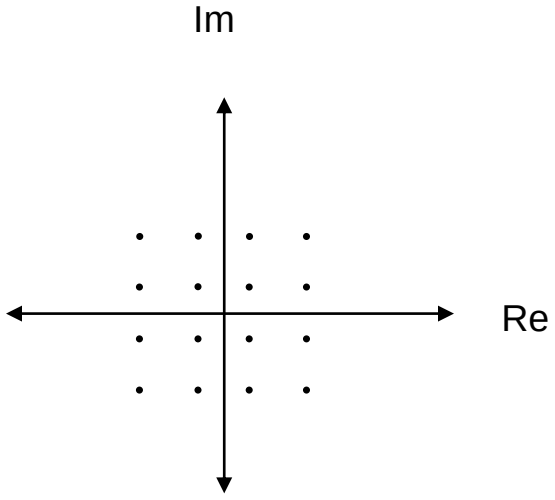
- Communication systems are power-limited
  - Finite power source (such as a battery)
  - Regulatory requirements (FCC)
  - Etc.

# Power and Signal Constellations

- How does transmit power affect the signal constellation?
  - More signal power moves the constellation points farther apart
  - Less signal power moves the constellation points closer together




16QAM – Higher Power




16QAM – Less Power

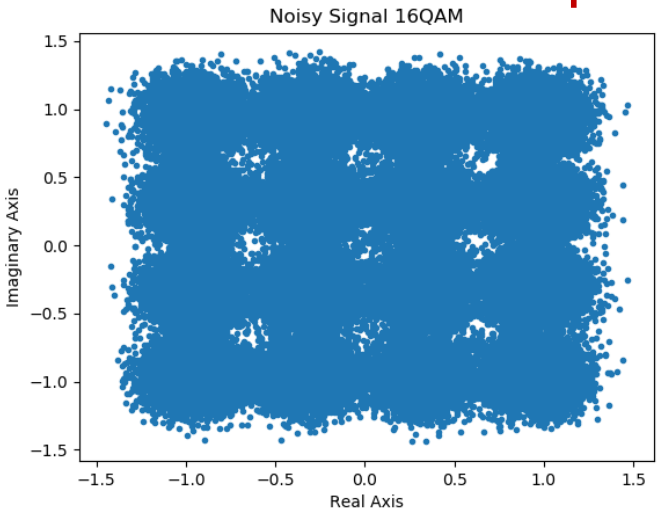
# Noise

- In general, noise in a system is relatively constant.
- The ratio between signal power to noise power is called the signal to noise ratio (SNR)
- Higher SNR is better

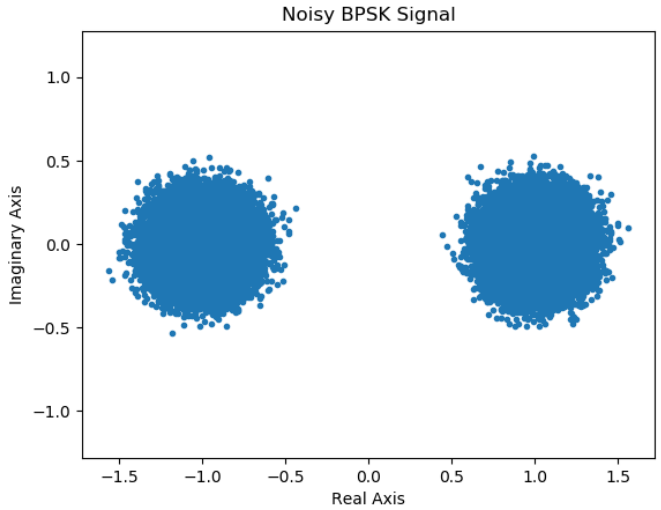
# Noise

- What if the noise is too big for our constellation?
  - Symbol to bit conversion mistakes are made
- Many things you can do to overcome this, but for the same transmit power, you can just change the constellation and transmit at a lower data rate

Mistakes in bit output




No mistakes in bit output




Same transmit and noise power for 16QAM and BPSK

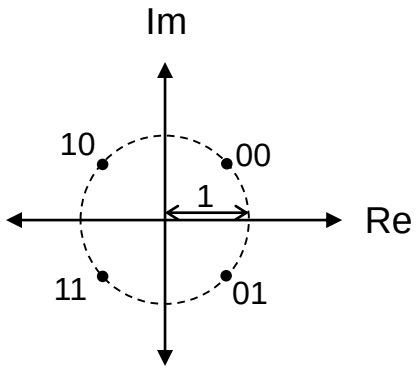
# Presentation Outline

- Project Introduction
  - Hardware
  - Software
  - Modem Design
- PRBS
- Task #1
- Introduction to Digital Communications
  - Transmission of bits
  - Modulation methods
  - Signal Power, Noise Power, SNR
- Task #2
- To be continued…

# Task #2

- Implement a modulation mapper and demapper for a QPSK constellation in python
  - Arrange the input bits into groups of two bits per group
  - For each group of two input bits there will be one complex number as output, i.e., if there were 100 input bits, there should be 50 complex output numbers

| Bit Input | Complex Output |
|---|---|
| 00 | $e^{j\pi/4}$ |
| 10 | $e^{j3\pi/4}$ |
| 11 | $e^{j5\pi/4}$ |
| 01 | $e^{j7\pi/4}$ |

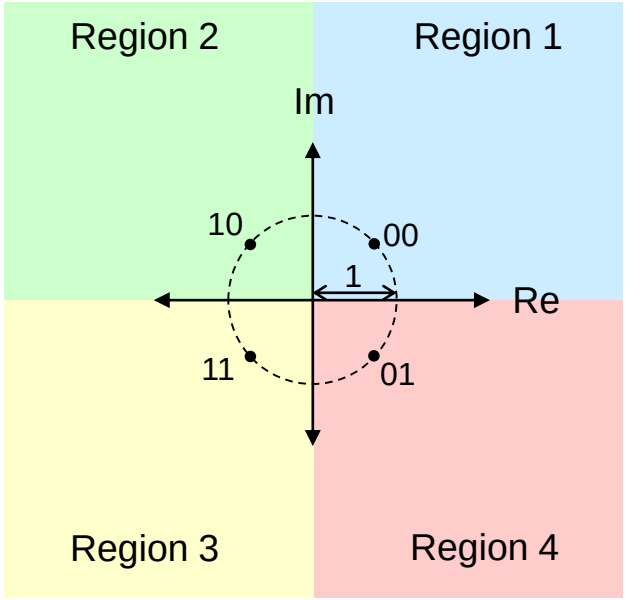



QPSK (4PSK)

# Task #2

- For the demapper, we map numbers in decision regions to bits
  - For QPSK there are four decision regions
  - If an input number lands on a boundary, it can go into either region that it touches

| Complex Input | Bit Output |
|---|---|
| Region 1 | 00 |
| Region 2 | 10 |
| Region 3 | 11 |
| Region 4 | 01 |



# Presentation Outline

- Project Introduction
  - Hardware
  - Software
  - Modem Design
- PRBS
- Task #1
- Introduction to Digital Communications
  - Transmission of bits
  - Modulation methods
  - Signal Power, Noise Power, SNR
- Task #2
- To be continued…

# Digital Signal Processing

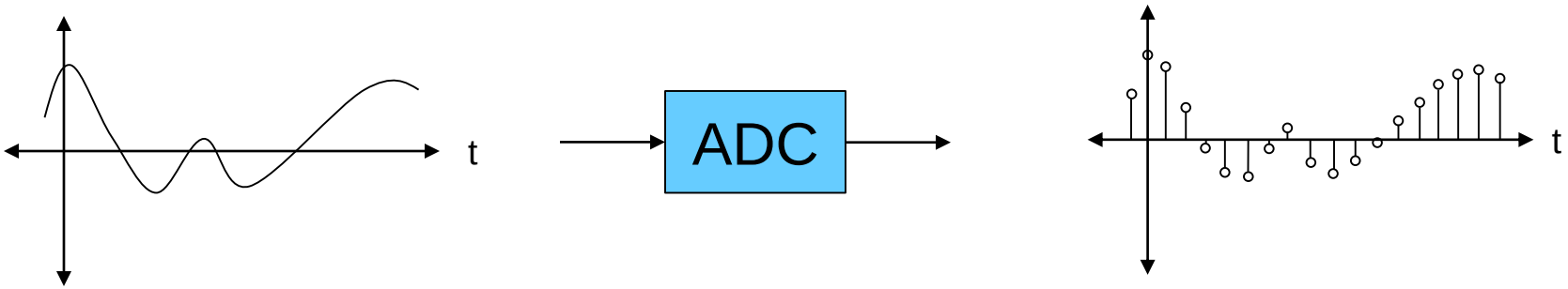
- In addition to working on a digital communication system, you will need to work with digital signals.

- In general, signals in the “real” world are analog signals.
- However, computers like to work on digital signals.

- What is a digital signal? How do we convert to/from digital signals?

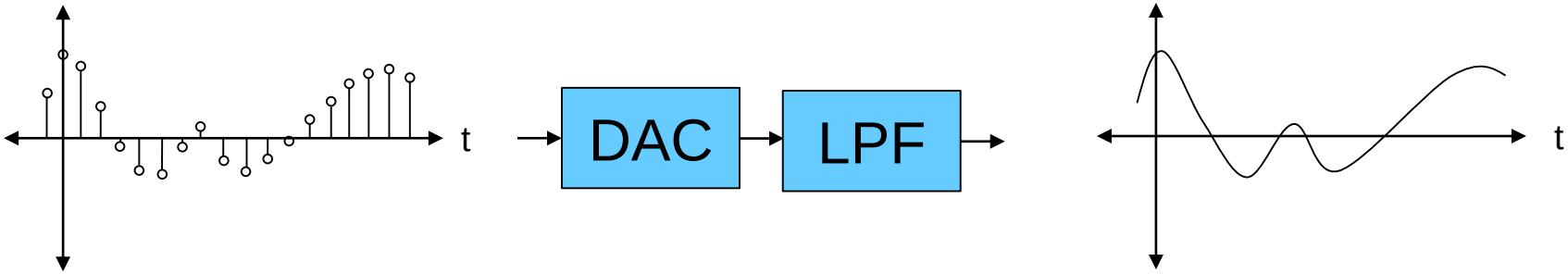
# Analog to Digital Conversion

- The process of converting an analog signal to a digital signal is performed by an analog to digital converter (ADC) or A/D and is called sampling.
  - The analog signal is fed into the ADC along with a clock.
  - At every tick of the clock, the ADC produces a “sample” of the analog signal
  - That sample is a number that matches the value of the analog signal at that point.



# Digital to Analog Conversion

- Converting a digital signal to an analog signal is performed by a digital to analog converter (DAC) or D/A.
  - This process is the reverse of the A/D conversion, but it requires a low-pass filter (LPF)



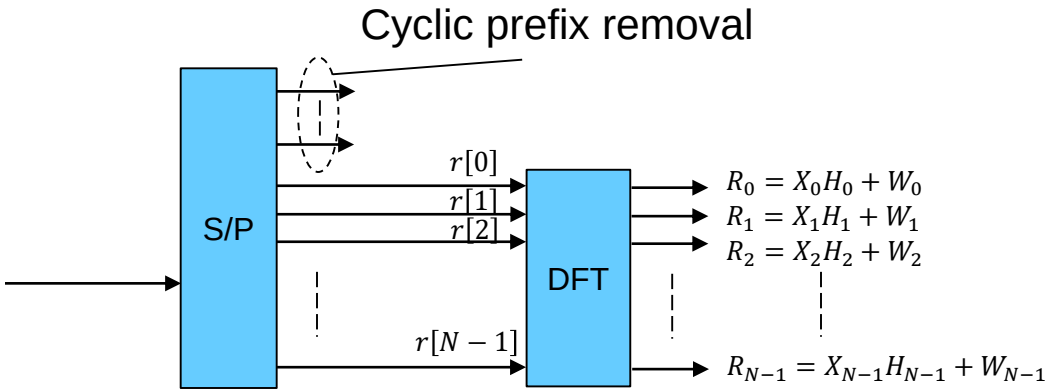
# Digital Signal Processing

- We will most likely cover this another day
  - See chapter 4 “Sampling and Discrete Time Systems” in *Signal Processing Techniques for Software Radios, 2nd Edition* by Prof. Farhang.

# CHANNEL EQUALIZATION

# Recovered Signal

- At this point, we have recovered burst timing and we have estimated and corrected any frequency offset
- We remove the cyclic prefix and feed our recovered signal $r[n]$ into the DFT block

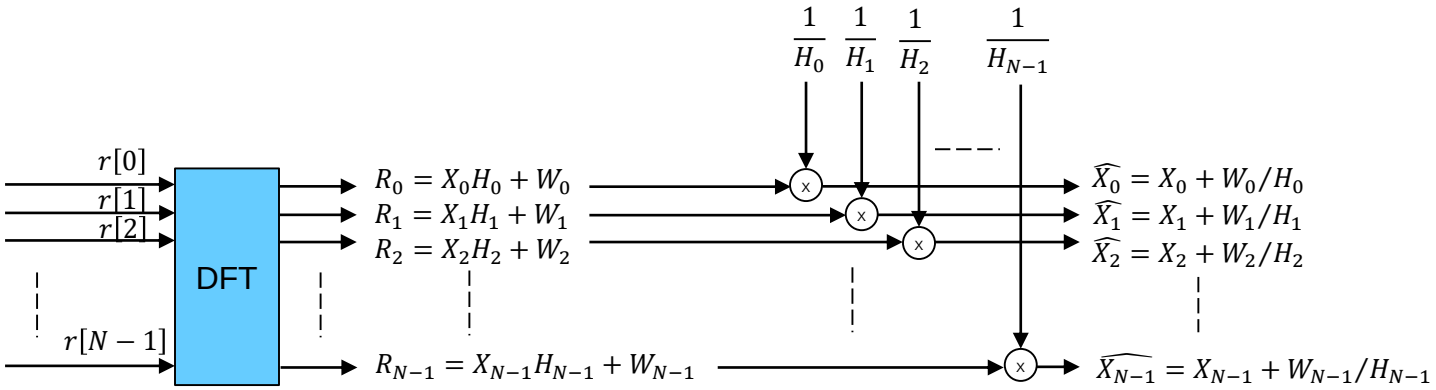



$X_l$ - Transmitted data symbol (QPSK) on subcarrier $l$.
$H_l$ - Channel frequency response on subcarrier $l$.
$W_l$ - Additive white Gaussian noise (AWGN) on subcarrier $l$.

$$r[n] \qquad R_l = X_l H_l + W_l$$

# Channel Equalization

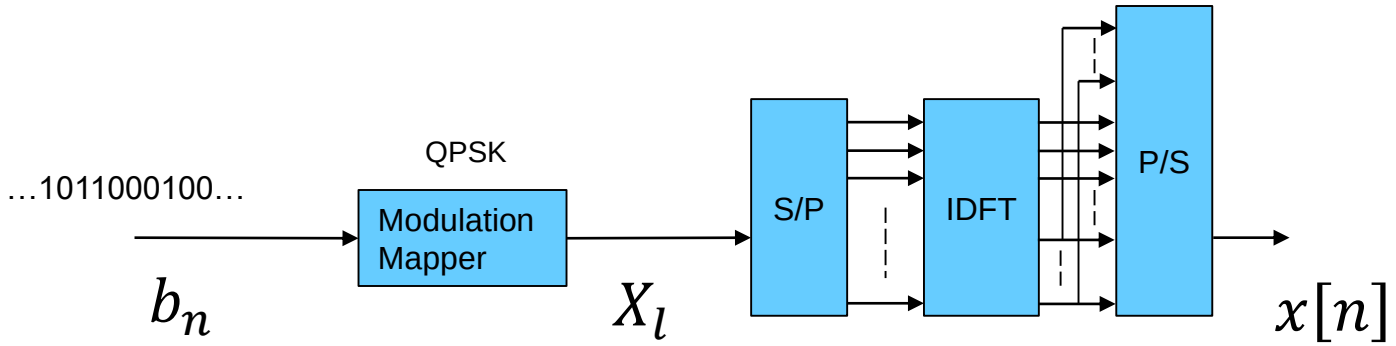
- We will follow the methodology in the diagram for channel equalization to get estimates $\widehat{X_l}$ of the QPSK symbol transmitted on subcarrier $l$.




- But the channel is unknown, so we need to find an estimate of the channel $\widehat{H_l}$ for every used subcarrier $l$.

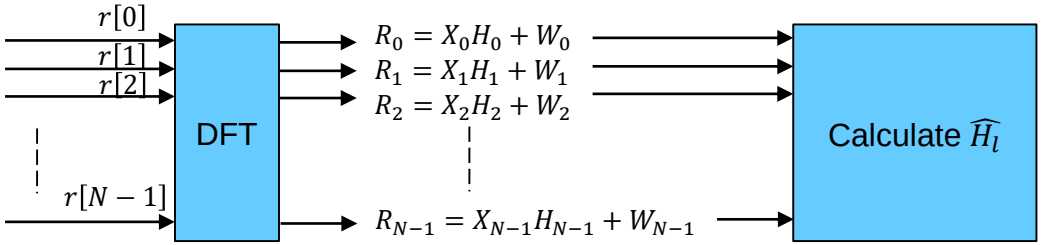
# Using the Preamble for Channel Equalization

- We will use the preamble to estimate the channel
- What do we know about the preamble?
  - We know the preamble bits $b_n$
  - We know the QPSK symbols $X_l$
  - We know the time-domain samples transmitted over the air $x[n]$
- We will use the preamble QPSK symbols $X_l$ to estimate the channel.



# Per Subcarrier Channel Estimates

- We will use the preamble QPSK symbols to generate an estimate of the channel $H_l$ on subcarrier $l$.

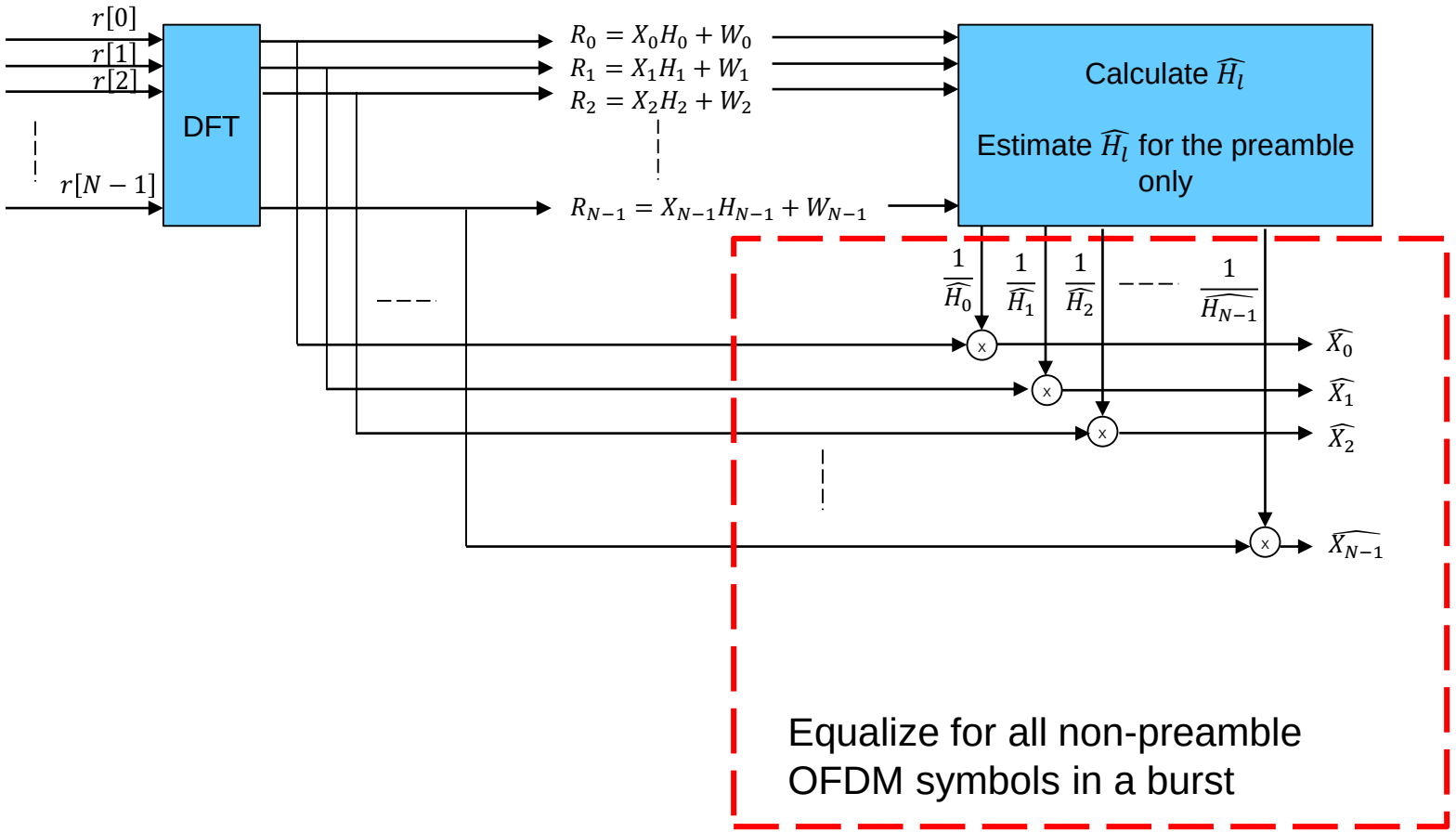



$$\widehat{H_l} = \frac{R_l}{X_l} = H_l + \frac{W_l}{X_l}$$

- With channel estimates in hand, we can equalize.

# Channel Equalization

$$R_l = X_l H_l + W_l \qquad \widehat{H_l} = \frac{R_l}{X_l} = H_l + \frac{W_l}{X_l}$$



# Equalization Mathematics

- Let's look at the equation for the QPSK symbol estimates $\widehat{X_l}$ by using the channel estimates $\widehat{H_l}$.

$$\widehat{X_l} = \frac{R_l}{\widehat{H_l}} = \frac{X_l H_l}{\widehat{H_l}} + \frac{W_l}{\widehat{H_l}} = \frac{X_l H_l}{H_l + \frac{W_l}{X_l}} + \frac{W_l}{H_l + \frac{W_l}{X_l}}$$

- We will assume that SNR conditions are sufficiently high, such that $\widehat{H_l} \approx H_l$.

$$\widehat{X_l} = \frac{X_l H_l}{H_l + \frac{W_l}{X_l}} + \frac{W_l}{H_l + \frac{W_l}{X_l}} \approx \frac{X_l H_l}{H_l} + \frac{W_l}{H_l} = X_l + \frac{W_l}{H_l}$$