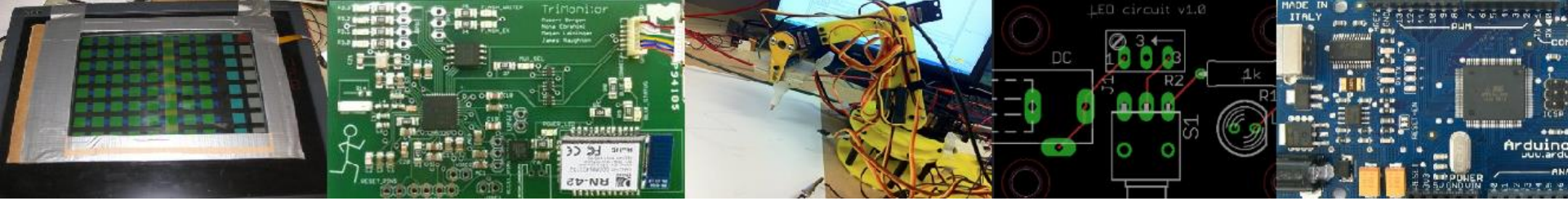



# EECS 473

Review etc.

MICHIGAN

# HW2 date

- Homework 2 is due 12/8 @7pm
  - Answers will be posted 12/9 by noon.

# Schedule

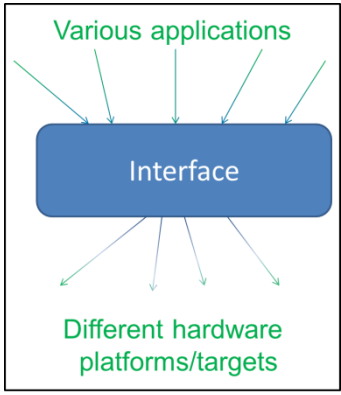
- We'll have some office hours to answer exam questions (plus Piazza) starting Thursday.
  - Details out tomorrow.
- Project reports will be graded by the exam time.
- Final project grades will be out by 12/12 or before.

# Returning stuff etc.

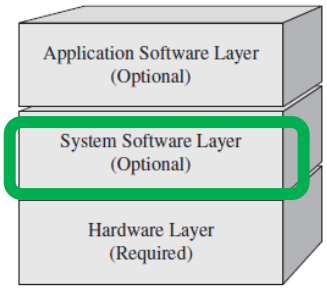
- Project disposal

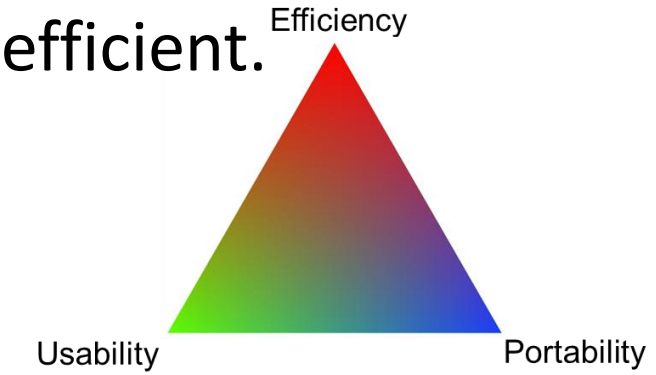
- We *do* need you to return *equipment* purchased and anything Matt thinks we could reuse (ask him after your demo).
  - Put that stuff in the lab, in a box (you supply) with your group name on it.
- Also, *all* reimbursement stuff due 12/12 unless you work something else out.
  - 12/14 at the very latest.

# Topics: Interfacing




- Writing software interfaces for hardware
  - Ideally have a standard interface for both hardware and programmer.
    - Makes it easy to port software.
    - Also means it's obvious what hardware control to provide.
  - Like any interface, standardization here is very powerful, but comes at a cost. Abstracting away interface issues makes things less efficient.
    - » Examples?



# Real-time systems and scheduling

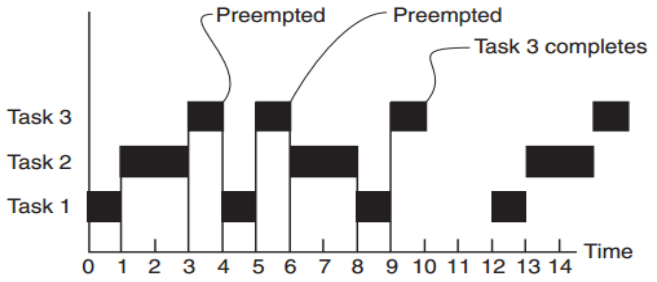
"those systems in which the correctness of the system depends not only on the logical result of the computation, but also on the time at which the results are produced";

- Time matters
  - Hard, soft, firm deadlines
- Validation if very difficult
  - How do you know the worst case timing?
    - Really difficult to prove worst case. Cache misses, branch prediction, etc. make for a very complex situation.
    - For safety critical things, even a “large engineering margin” isn’t enough.
      - Need to actually figure it out.

# Real-time systems and scheduling

- Rate monotonic scheduling
  - Static priority scheme
  - Assumes "all" tasks are periodic.
    - Give priority to tasks with lower period.
  - Total utilization helps figure if schedulable.
    - If is less than $n(2^{1/n}-1)$ (n=number of tasks) it is schedulable.
    - If over 100% not schedulable
    - If neither is true, do critical instant analysis.
- EDF
  - Requires dynamic priorities
  - Works if less than 100% utilization

| Task | Execution Time | Period | Priority |
|---|---|---|---|
| T1 | 1 | 4 | High |
| T2 | 2 | 6 | Medium |
| T3 | 3 | 12 | Low |



# Licensing

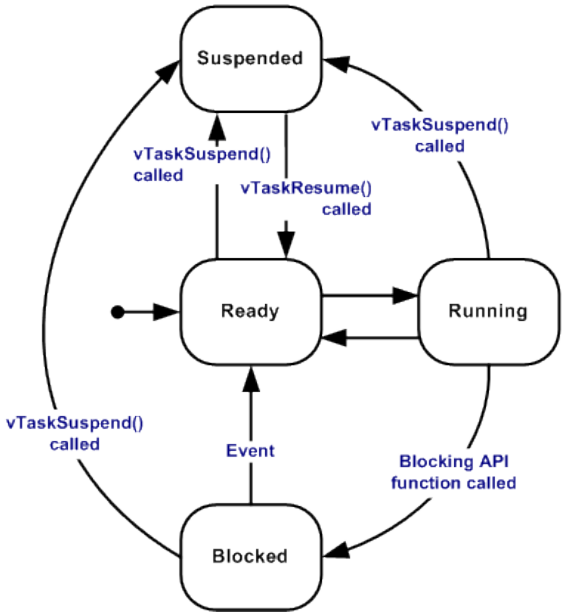
- What a viral license is
  - Why it matters in embedded perhaps more than elsewhere.
    - LKM
    - Impact on business model
    - Hardware people tend to use a lot of other people's code (legally).
      - Vendor's driver code etc.
  - Libraries.

# Topic: Software platform

- We covered three or four basic platforms for software development for an embedded system.
  - Barebones
    - Write everything yourself
  - Barebones plus libraries
    - Import some useful libraries but otherwise write it all yourself.
  - RTOS
    - Basic scheduler with a lot of control
    - Generally a fair bit of support.
      - I/O devices, memory management, etc.
    - Fast interrupts processing possible/reasonable/"easy"
  - Full OS
    - Give up a lot of control
    - Have to deal with a very complex system
    - Get lots (and lots) of software support
      - Vision, databases, etc.

# Free RTOS

- Tasks and scheduling
  - Creating tasks (xTaskCreate)
  - Semaphores
  - Deferred interrupt processing.
  - Can dynamically change priority.



# FreeRTOS

- *Likely* your final design problem will involve using FreeRTOS in some way.
  - Review lab 4
- We'll provide sample code and/or basic functions
  - You don't need to memorize syntax, but you do need to understand.

# Embedded Linux

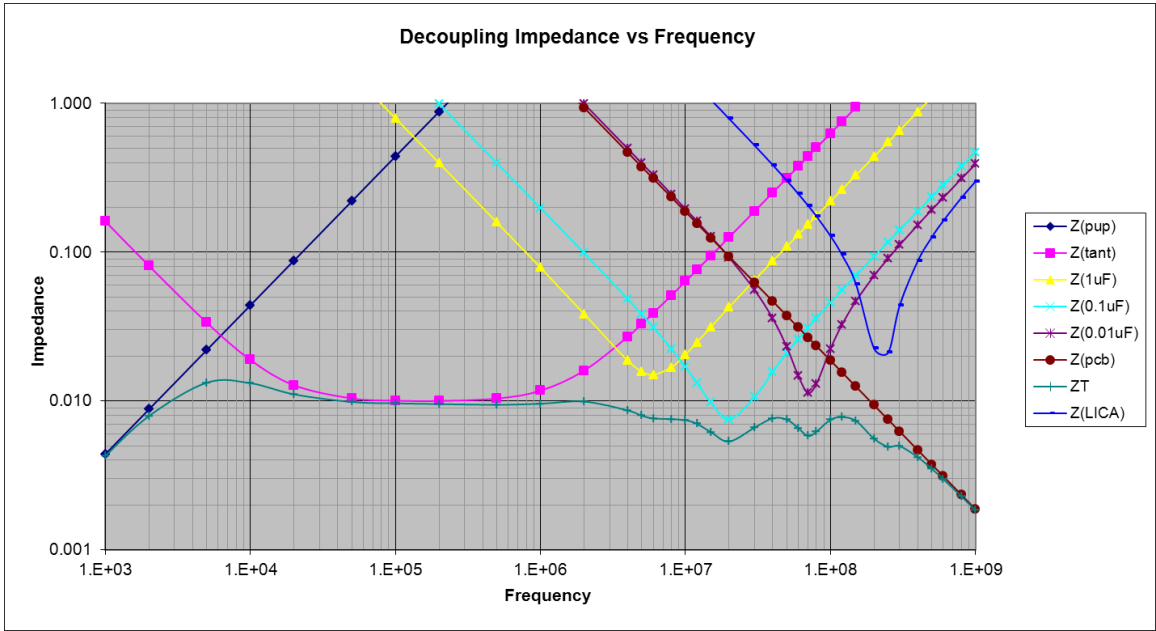
- What limitations on realtime you might have
- Can be fairly small footprint (not much memory)
  - Things like busybox help
- I/O has a standard interface
  - File model
    - Not always ideal.
- But there is a lot of complexity here
  - We spent a fair bit of time writing drivers.

# Sample OS question

- What are the pros and cons of using a “full” OS (Linux, Windows etc.) in an embedded application? Give an example where you would certainly want to use such an OS and where you certainly would not want to.

# Power integrity

- Discuss keeping Vcc/GND constant as possible.
  - Recognize that our devices can generate current draw variations at a huge number of frequencies.
  - Spikes or droops could break our device.
- Need caps.
  - Small and large
  - Get right values

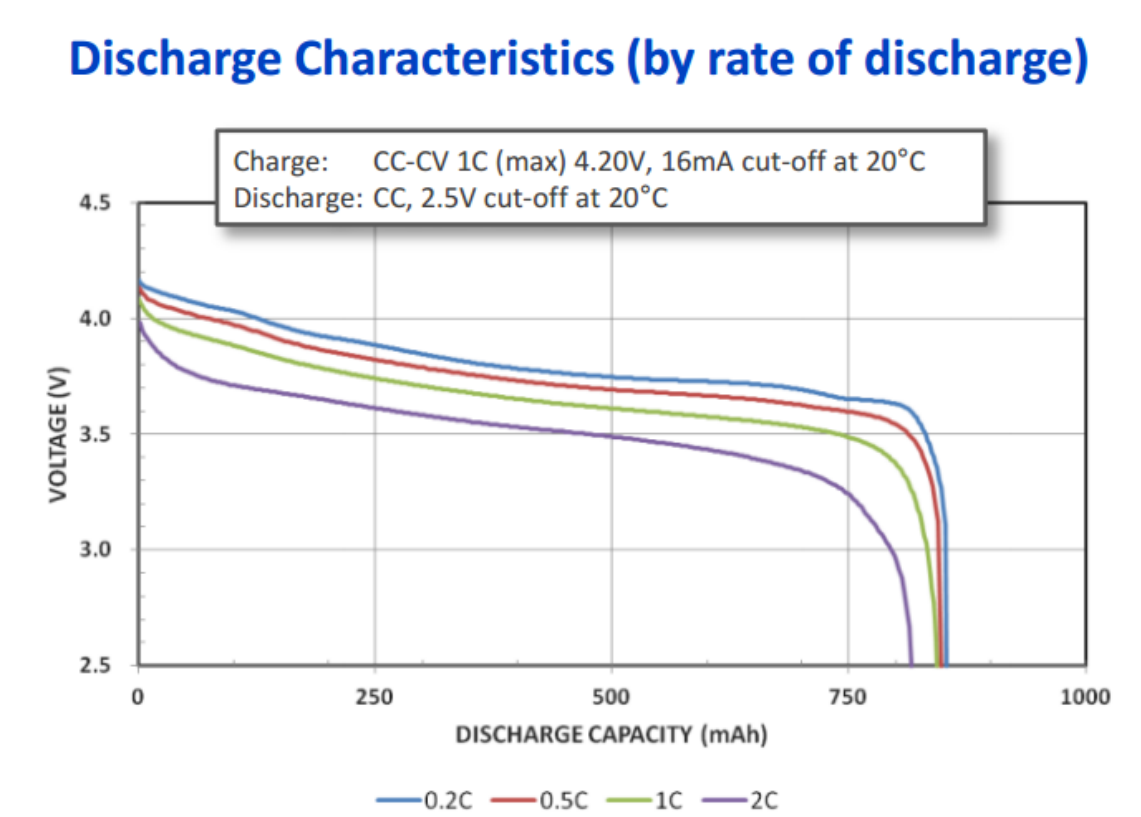


# Batteries

- Understand mAh
  - Understand that mAh will be less if draw too quickly.
  - Be able to work basic math using specific battery properties.

# Example

- 800mAh battery.
  - If we need 3.5V (or more) how long will this battery last at a 1.6A draw?

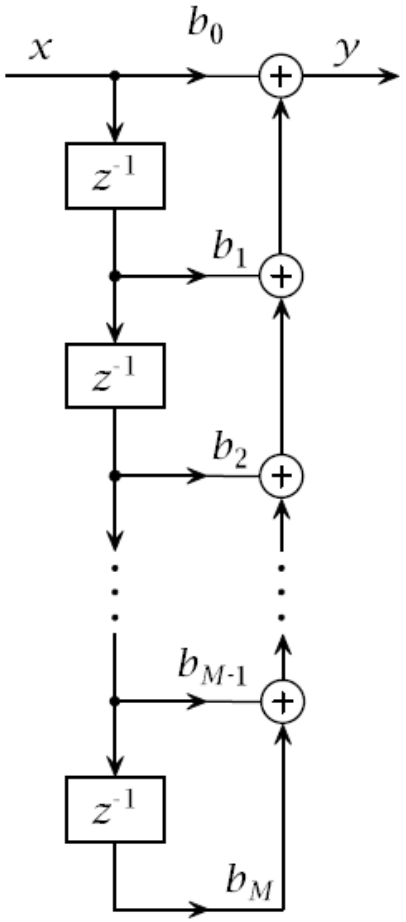


# DSP

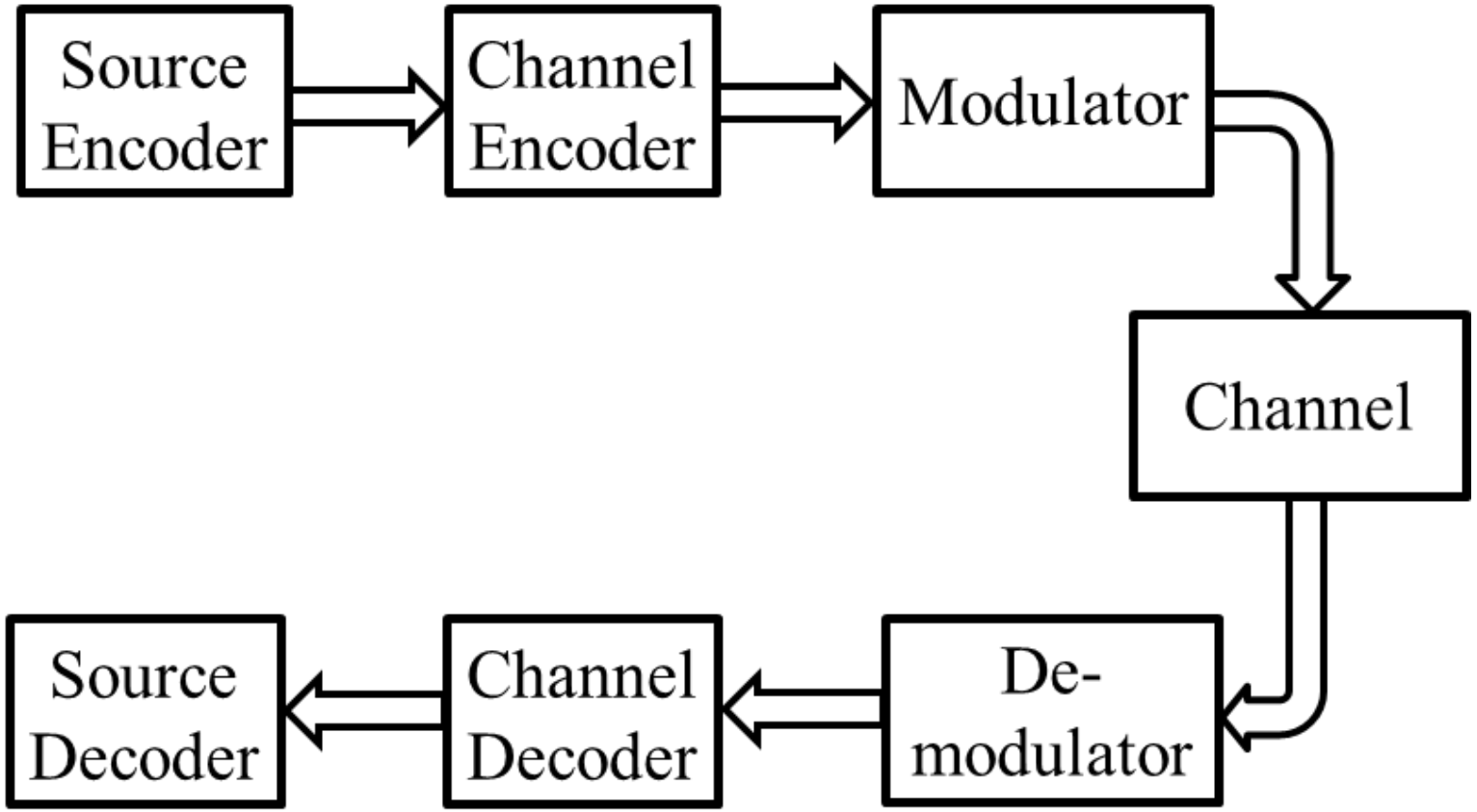
- We covered this for three reasons
  1. To give you a sense of what digital signal processing involves
     - What are the characteristics of
  2. To make it clear that there can be specialized computational engines out there.
     - What are some common special-purpose processors we discussed?
  3. An excuse to show fixed-point.

# DSP and FPGA

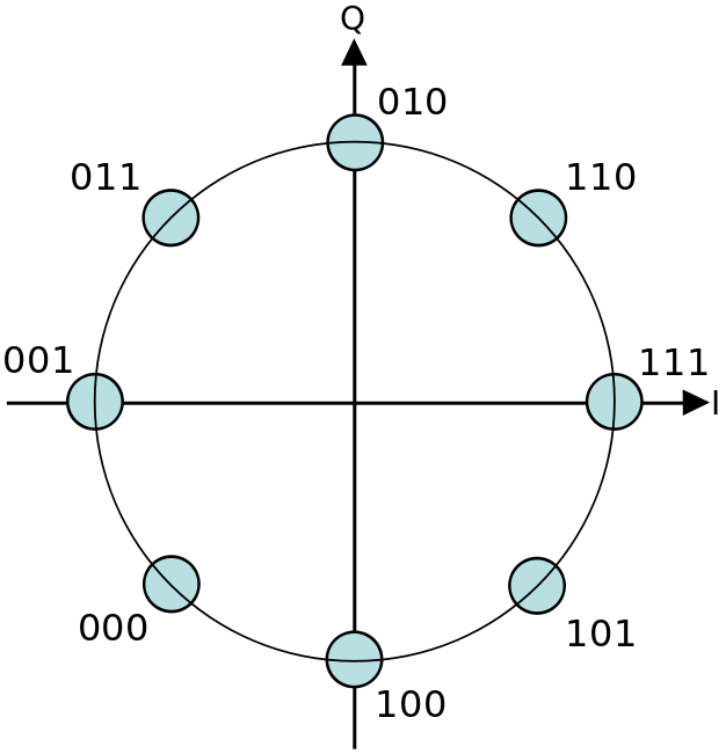
- Example (utterly unfair) question:
  - Consider the structure on the right.
    - If a flip-flop has a delay of 1ns
    - A multiply has a delay of 10ns
    - An add has a delay of 2ns
  - What is the lowest clock period you could get for a 64-tap (64 multiplies) implementation of the structure on the right?
    - (Yes, this involves 270 stuff)

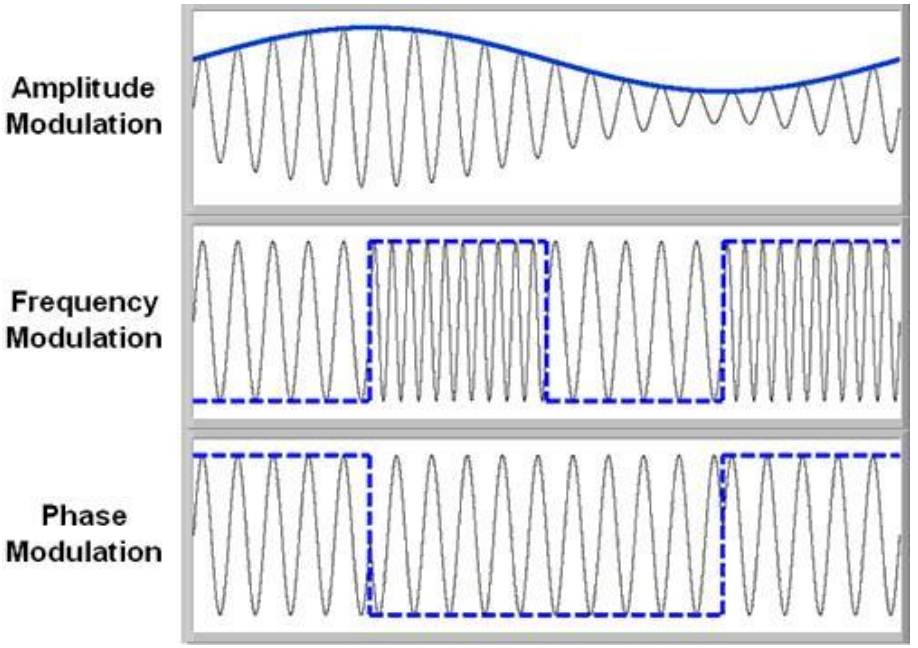


# Wireless


Source Encoder
Channel Encoder
Modulator
Channel
De-modulator
Channel Decoder
Source Decoder

# Modulation




Of frequency, phase and amplitude modulation, which are used in the above constellation?

# Shannon–Hartley theorem

- We'll use a different version of this called the Shannon-Hartley theorem.

$$C = B \log_2 \left(1 + \frac{S}{N}\right)$$

- **C** is the channel capacity in bits per second;
- **B** is the bandwidth of the channel in hertz
- **S** is the total received signal power measured in Watts or Volts$^2$
- **N** is the total noise, measured in Watts or Volts$^2$

Adapted from Wikipedia.

# Sample questions for wireless

Say you are sending a signal using the band from 2.45GHz to 2.5 GHz and at the receiver the power of the signal is three times that of the ambient noise. According to the Shannon–Hartley theorem, the highest achievable data rate is about ***25 / 100 / 250 / 1000 / 2500*** Mbits/sec

$$C = B \log_2 \left(1 + \frac{S}{N}\right)$$

dBi is the gain of an antenna compared with the hypothetical isotropic antenna. An isotropic antenna is an antenna which

- ***requires a 50Ω matching impedance.***
- ***uniformly distributes energy in all directions.***
- ***distributes energy in a toroid.***
- ***must have the same length on the transmitter and receiver.***

# Power received vs. power sent.

- The Friis Transmission Formula tells us how much power we'll receive. It is:

$$\frac{P_r}{P_t} = G_t G_r \left(\frac{\lambda}{4\pi R}\right)^2$$

Where:

- $\mathbf{P_t}$ is the radiated power
- $\mathbf{P_r}$ is the received power
- $\mathbf{G_t}$ is the gain of the transmitting antenna
- $\mathbf{G_r}$ is the gain of the receiving antenna
- $\boldsymbol{\lambda}$ is the wavelength
- **R** is the distance between antennas

- However, many of those terms aren't easily available from real spec. sheets.
- Instead we do some algebra and get the following equation:

$$r = \frac{10^{(p_t + g_t + g_r - p_r)/20}}{41.88 \times f}$$

- Where *f* is the frequency in MHz, $p_t$ and $p_r$ are in dBm and $g_t$ and $g_r$ are in dBi. r is in km.
  - As a note, this is a theoretic result. In reality we often divide by 4 or more.

https://webbrain.com/attach?brain=C0ABA1EA-DC45-635E-40CA-C0636235A853&attach=226&type=1

# Questions

- dBi
  - What is it exactly?
  - What do we use it for?
- What is dBm?
  - Why use dBm instead of dB or mW?
- Do lower or higher frequency signals go farther?