Networking & Communications

- Individual Assignment:
  - Bluetooth, sircad, stepdance, etc all doing something with networks
- Group assignment:
  - Send a message between any two projects

Why Network?

- Location → not in the same place
- Run tasks in parallel (ex/ splitting between memory using threads of a process → give each core a microcontroller so each doing either system integration, motion, UI, etc
- Modularity: can debug each task separately, add tasks without changing central processor
- Interference: Might have low current low noise sensor in high noise environment that you want to keep separate

Wired

- UART/USART: universal asynchronous/synchronous receiver transmit
- SERCOM: more general modules in arm processors that can do any type of process
- Also PIO, bit-bang

Asynchronous Serial

- Bus = single string of nodes, but all have addresses on the bus
- RS-232 standard for serial, RS-422/485 make that multidrop → serial bus
  - Usually transmit goes to receive and vice versa; but sometimes people flip it, so check datasheet
  - Talk line goes out and all receivers listen; one host line going to receiver for all transmit
    - But how do you share that? Use tri-stating
  - But main limitation is need to thrash each node with address
- Better to use hop-count:
  - Each node has TR in and TR out
  - One way to do it (RGB LEDs do it) is make a packet, and each note strips of bytes in the packet
    - By sending message down and back up you get addresses, but don't have to assign it yourself because it'll strip info until it reaches the last, and comes back → serial bus without making addresses
  - But issue is to get to end of string have to go through many hops
- So can use broad-hop:
  - Single transmit line out everyone listens to, but hop back → host talks to everybody, but use hop to give them unique addresses
  -

Synchronous Serial

- RS-232 has one bad attribute which is that a node listening has to catch the start bit to know it;s listening; and have to agree in advance on the bit time → if two sides differ you're doomed
- I2C: has two lines, SCL and SDA (example detects device plugged in and address)
    - No Tx/Rx switching, connected to each other; can have multiple devices connected to these and either side can drive it
        - Bc lines get pullup resistors that pull it high, and you pull it low – so rather than tristating you switch who's pulling it low
        - Clock line starts ticking, tells you "here comes data", then read data synchronized to clock time → don't have to hope to find start bit or find agreement on timing
- TWI: same as I2C but not violating phillip's patents, but doesn't matter anymore
    - Qwiic, STEMMA → lots of interfaces that connect to it
    - Arunio has wire library with transmission system send and receive; also bit-banging
- Tend not to use it when making own network bc easier to get serial bus
- I3C emerging with a lot more capability, but not gotten much traction yet
- SPI: has a clock line and MISO/MOSI (SDI/SDO/CS/SCK all version to not call it slaves)
- PICO/POCI adds a secondary line, actively driven, can drive much faster bc of the pullup
- Classic use of SPI is SD memory cards

Asynchronous Unlocked

- Not standard in any way, just illustrating – token passing where asynchronous and unlocked → line that says 1 and 0 and async that says data present" through tokens should route it
- TinyUSB
- Ethernet used for high speed networking; generally don't write to them

A simple serial bus already has key ideas for a multiprocessor system!

Physical Media

- Amount of info range of frequencies time log of ratio of signal to noise; more signal is good, more bandwidth is better
- Wired bandwidth start with one wire but immune to noise; better is two wires and measure difference between them
- Great table of frequency and who owns what part → care about ISM bands, frequencies available to do whatever you want twitch
    - 2.4ghz was originally a junk band bc next to microwave ovens, but colonized for WiFi and since become important, but overloaded and now moving up to higher frequencies
    - Higher bands let you push more data through, but as you go up it's more line of sight and can't go through walls (and vice versa)
- Wifi: 801.11 another set of bands for wifi

- Bluetooth BLE: almost unrelated to what i think of as bluetooth, not interoperable; in BLE slew of acronyms that turn into declared services → apps to work with it, stacks for desktop, arduino/micropython have libraries, and **web bluetooth from browser**
- **Can also send data through sound**

Modulation/Channel Sharing/Errors

- To send message through radio need to modulate it
- how to take radio carrier and put radio on it, figure out who gets to talk when
- as you push data rate and distance will make mistake so need to detect and correct errors

Networking

- Internet defined by RFCs – if can format data in that form and push it on network can go anywhere on earth
    - To implement, there's a stack:
        - Lowest is physical: wire to wireless
        - Then media access - who gets to talk to it
        - Ip layer deals with routing (knowing address)
        - Then UDP or TCP
            - UDP like rooms in a building, once packet comes need to know what application gets it
            - TCP manages sending multiple packets and checks to see if got through (better but adds overhead)
        - So use a sockets interface to talk over interface
        - Wireshark tool that lets you sniff networks
    - In early days of internet would send nodes over serial ports

Wireless

- Radios:
    - Has an oscillator that makes signal, mixer that modulates signal with your data, power amplifier that sends it out, low noise amp that takes signal back in, intermediate frequency stage that brings it down to the base band, INQ stages to recover the phase of signal, Demod gets it down to base band to communicate, typically also filters to clean it up and give you the data
        - Great thing is now implemented in single chip radios
    - Then need antenna:
        - If have heavy chain connected to skinny chain and shake it, will be reflection at the connection point → so too for circuits, so need to match the impedance of free open space to get it out of your board
        - Can have antennas that do the matching, or short antennas that match by electrical circuit network (like ESPs)
    - Single chip radios wonderful:
        - Has all those parts

        - ESP-32s do great job of integrating all of this (can even connect desktop computer to XIAO web server to host webpage to turn off functions on chip)
        - NRF radio vendor has nice app; many different protocols, but handiest one is when acts like a wire (serial port can put data through)
            - Example uses app from NRF, connect to node, then send commands over app to BLE to node then sending our through serial
            - really easy to use BLUART (neil wrote, his packaging based on the standards – takes slew of acronyms and bundles it up into a serial port) – just say “i want a wireless serial port” and send data through it
- ESP camera – streams video (showed in input devices)
- RFID:
    - Using library, tell it what pins to connect to, can read data on the card, write data to the card (very little in there, just chip and coil, can by stickers with those)
        - Commoditized, cheap, and cheerful → but trivially hacked!! Don’t use for anything of value
        - Difference between credit card tap? Neil’s lab spun off thingmagic but essentially very complex; collaborators in belgium use physics to defeat cryptosystem → don’t know of easy version of more secure RFID
- NRF: MakerFocus modules are radios in the same band → one can talk to a few, pair to each other, etc → all do is get message between them
        - Say racecar and controller to send message, don’t need complexity, just put it in and data goes in one side and out the other
- CYW: does wifi, bluetooth, and FM
- RA group → not yet in lab inventory, neil has a bunch bringing in for recitation
    - All speak LoRa (rapidly getting traction, like bluetooth but to go much longer ranges from many kilometers) → seed has a LoRa module aimed for meshtastic
    - Meshtastic takes lora modules and builds a mesh out of them; and can do this on the scale of cities
        - Dimitar worked at starry which does internet with meshes at high data rates; this does very very low data rates but also at very very low power → great for passing data around the farm, control system in a building, that adaptively work in a mesh
- Near field radios - like RFID but an active version - lose some privacy on phone because asking any tag in vicinity, then just communicating over phone network
    - In early days could make early phone system device, but gotten a lot harder to connect with them now that phones do a lot more hopping\
- BU01: ultra wideband module → instead of narrow frequency, uses all frequencies, but at low power underneath noise floor
    - UWB interesting for low data rate; the more frequencies you use the better you’re able to track position
- Software radio:

    - Take an antenna and amp and connect it to processor → can do most of radio protocol in software rather than hardware, so now like logic analyzer for the ether to demodulate and syhtezie almost any signal
    - openwrt.org is a project that takes radio access points and turns them into open source platforms → any open wifi turned into a server

Now we make things with addresses! Did it in machine week; simplest is simple wired network, but think about for your final project; SPY/I2C, bluetooth, wifi, etc → **but difference is devices need addresses, and should think about your system architecture where you break into modules**