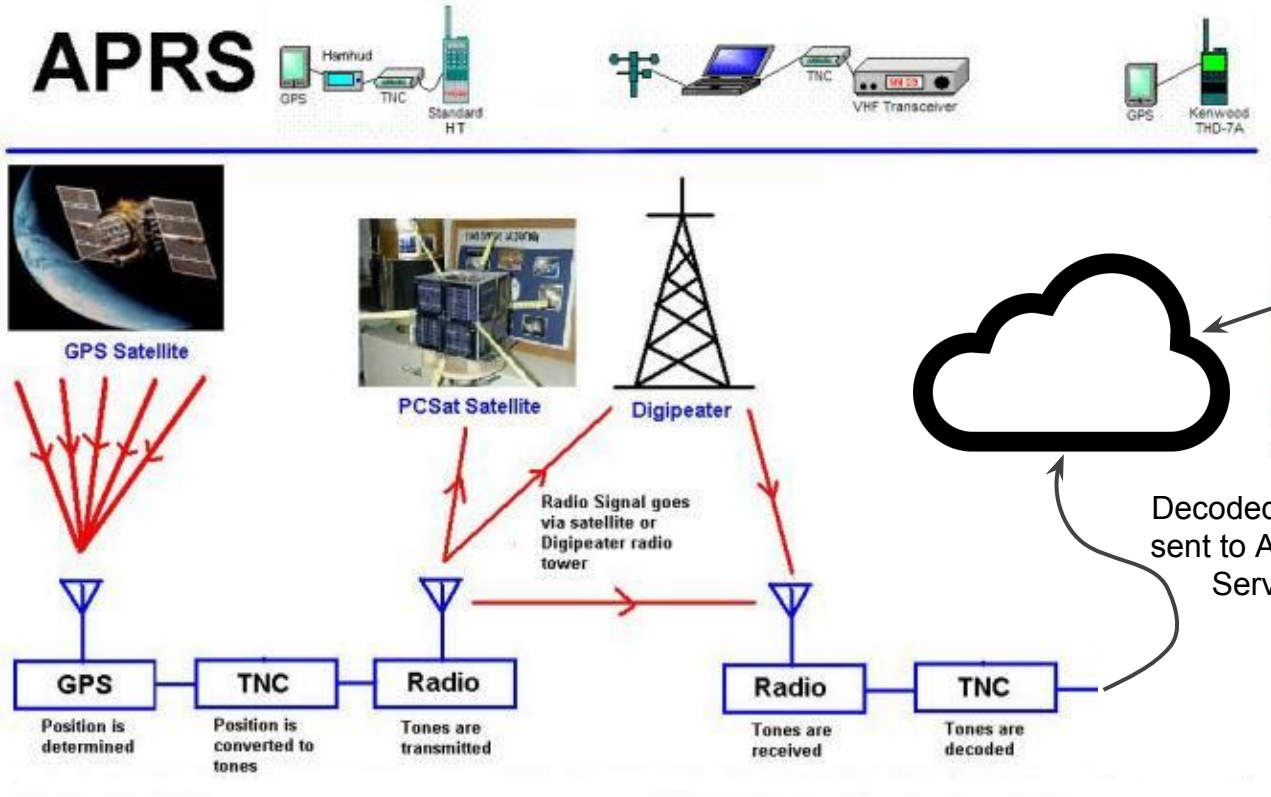
Tracking System

Theory

- Payload includes:
  - Arduino Mega 2560
  - UBlox GPS Neo 6M
  - Radiometrix HX-1 Transceiver for 144.39 MHz
  - Assorted Sensors
- Front end includes
  - Internet connected computer

Theory of Operation: Data is collected from sensors and GPS module, encoded in data packets, and sent through a radio transceiver to the APRS, where it can then be read from the internet.

APRS Network



Challenges

- GPS
  - Understanding serial communication between module and Arduino serial port
  - Understanding and parsing NMEA sentences for data
- Sensors
  - Understanding I^2C communication protocols
  - Trying to write our own code for the sensors is like trying to reinvent the wheel
- Radio
  - Encoding data in AX.25 format
  - Frequency generation and modulation
- Front End
  - Connecting to online servers using “sockets” with appropriate credentials
  - Parsing decoded data packets for information

Radio Challenges

- AX.25 UI Frames
  - Requires start with a 0x7E flag, followed by call signs and SSIDs
    - SSIDs need to be constructed using 011 - SSID in 4 bit representation - 0 or 1 depending on if last ID or not.
  - Cyclic Redundancy Checks
    - Goes through each bit in a message and XORs it with the remainder from the last XOR and shifts it over through the entire message.
    - High and low FCS is calculated and appended to the end of the message.
  - Bit Stuffing
    - Every 5 1s bits that are not in a flag need to be broken up with a 0 during sending.
  - Ends with another 0x7E flag

Radio Challenges Continued...

- Sine Wave Generation
  - Sine wave generated using a look-up table of values, and varying the duty cycle of fast PWM so that when passed through a low pass filter it produces a sine wave.
    - Figuring out non-traditional timers and PWM in Arduino is REALLY hard.
- Frequency Modulation
  - Messages are sent bit by bit. No change in frequency results in a 1. A change in frequency results in a zero. Modulation needs to happen on the same time frame as the lowest. Frequency varies between 1200 and 2200 Hz.
- Pre-emphasis
  - Higher frequencies tend to be attenuated more, and are usually half the voltage when produced by the Arduino. This is not acceptable by packet TNCs. (terminal node controllers)

Summary: All of the code for parsing the data into the correct AX25 UI frames is done, however there is conflicting documentation for the CRC check and whether it should be calculated LSB or MSB first (our code uses MSB) and whether bytes should be reversed. The code for generating sine waves is done and tested. Code is written to modulate the frequency but has not yet been tested since we don't have a good packet timing system done yet.

Sensor Challenges

- Writing our own code.
- Writing our own code for the sensors is more challenging than it seems.
- The adafruit library is somewhat universal, in the sense that it is meant to work with all of adafruit's sensors.
- Trying to connect the sensors to the same terminal on the Arduino using I2C communication protocols.

GPS Challenges

- Understand and read what is coming from the given GPS fix
- Separating the GPS National Electronics Association (NMEA) data and only print the GPRMC
- GPRMC would print:
  - GMT time
  - Receiver warning A: active and V: void
  - Longitude
  - Latitude
- Separating the GPRMC structure:
  - Writing a code to assign each variable of the GPRMC into different variables
- Debugging the code and trying to make it work:
  - Understanding the errors received and make the code work

GPS Output

```
GPRMC,202536.00,A,4530.85354,N,12240.77100,W,1.983,,040618,,,A*
GPRMC,202537.00,A,4530.85383,N,12240.77041,W,2.188,,040618,,,A*
GPRMC,202538.00,A,4530.85411,N,12240.77023,W,2.617,,040618,,,A*
GPRMC,202539.00,A,4530.85482,N,12240.77247,W,1.436,,040618,,,A*
GPRMC,202540.00,A,4530.85563,N,12240.77434,W,0.570,,040618,,,A*
GPRMC,202541.00,V,,,,,,040618,,,N*
GPRMC,202542.00,V,,,,,,040618,,,N*
GPRMC,202543.00,V,,,,,,040618,,,N*
GPRMC,202544.00,V,,,,,,040618,,,N*
GPRMC,202545.00,V,,,,,,040618,,,N*
GPRMC,202546.00,V,,,,,,040618,,,N*
GPRMC,202547.00,A,4530.86660,N,12240.81367,W,2.124,,040618,,,A*
GPRMC,202548.00,A,4530.86822,N,12240.82630,W,0.874,,040618,,,A*
```

Front End

- Needed to connect to an APRS-IS server using the particular APRS TCP protocol
- Used sys/socket.h library to connect to the server with my call sign and unique generated key to read packets.
- Use server filter commands to search just for packets that bear our call sign.

```
Parsing Packets

K5TRL-11>APRS,WIDE1-1,WIDE2-1,qAR,KI7PDX:143023,@4840.58N/00851.04E!356.23,4,t-23.36,p630.01,a357.30,h31.25%,#Hi Dr. Wong!#

Callsign: K5TRL-11
Path: WIDE1-1, WIDE2-1
Relayed by: KI7PDX

Time: 2:30:23pm
Latitude: 4840.58 N
Longitude: 851.04 E

GPS Altitude: 356.23 meters
4 Connected Satellites

Temperature: -23.36 C
Pressure: 630.01 Torr
Altitude (2): 357.30 meters
Humidity: 31.25%

Message: Hi Dr. Wong!

NOTES: At high altitude, there will only be WIDE2-1 as the path
qAR or qAC are q codes and we don't have to worry too much about them.
Temperature may be positive or negative
All measured values should have 2 decimal places
We'll use symbols or letters to designate different measurements or values. (i.e. t for temperature)
```

Front End Continued...

- Packets are saved to a text file which is parsed by the front end console when new packets arrive (approx. every minute)
- All telemetry and location data will be displayed.
- Future additions will possibly include an ASCII map or at least a version of the code that can work with a hand-held computer with its own GPS module and calculate the direction and distance to travel.

Still to Come

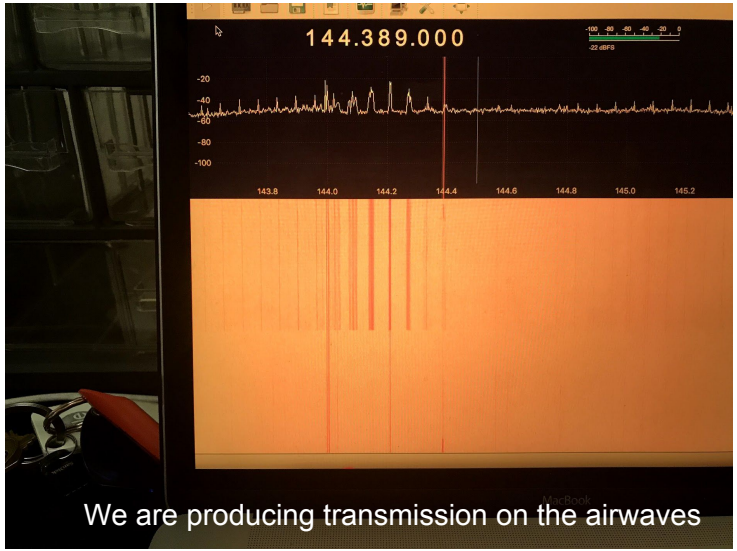
Back End

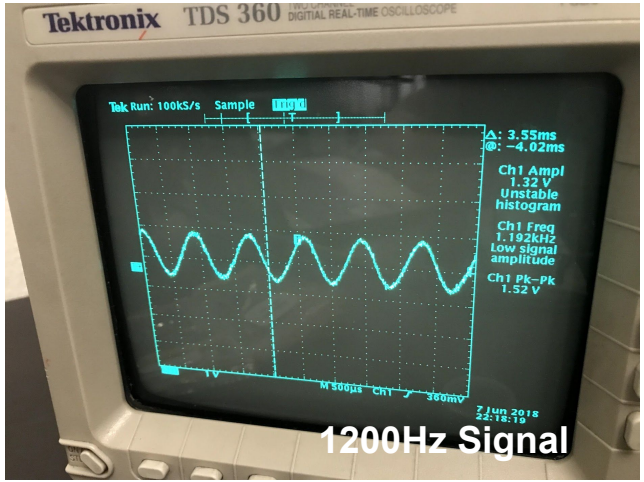
- Bringing together all of the disparate pieces - sensors, GPS data, and the modulation.
- Parsing GPS sentences for data to pass to APRS script and timing
- Testing the passing of data from sensors to the APRS script
- Testing our transmission with a commercial TNC to see if it decodes the packet properly, has the correct CRC.
- Testing out that transmission is strong enough to reach digipeaters and is parsed correctly by APRS servers

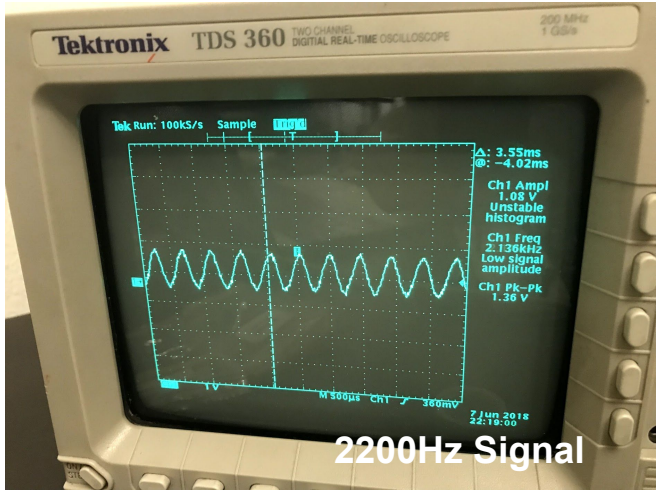
Front End

- Actually calculate position on ascii map and blink indicator
- Use a more zoomed in map of just PNW.

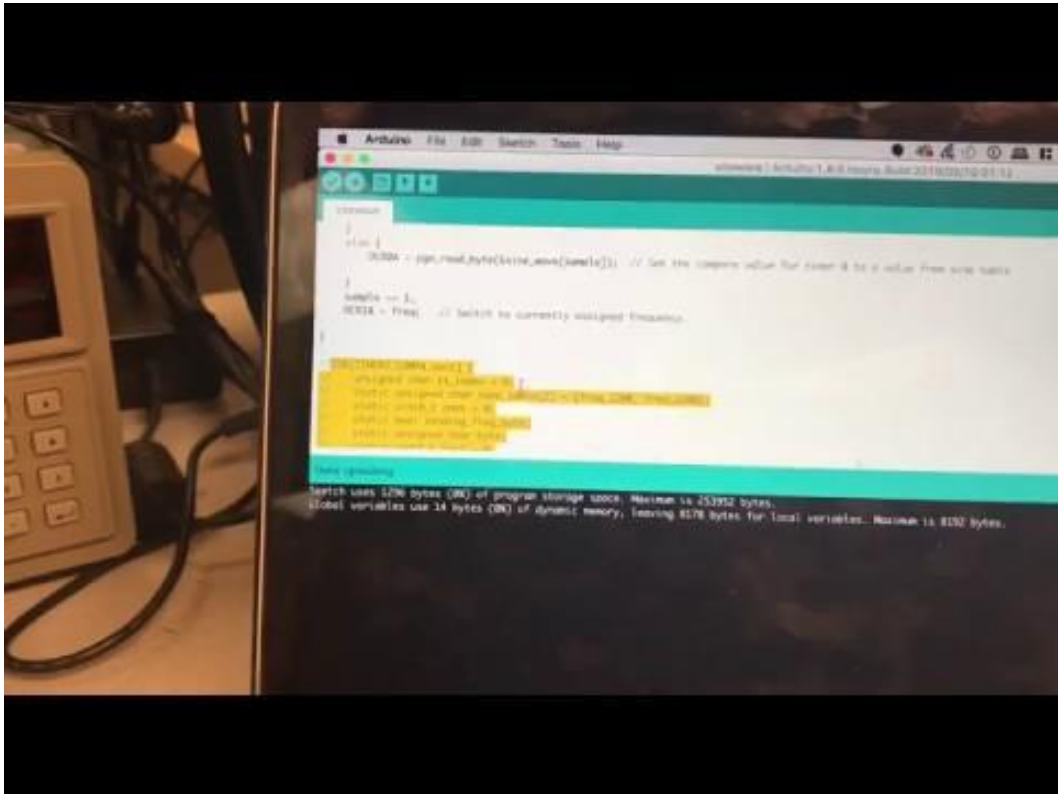
Some Photos


144.389.000
We are producing transmission on the airwaves

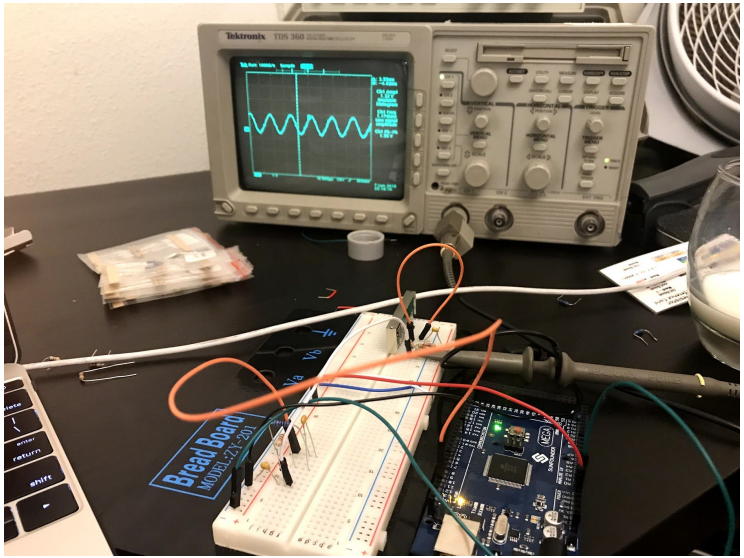

Tektronix TDS 360
Tek Run: 100kS/s Sample
Ch1 Ampl
1.32 V
Unstable histogram
Ch1 Freq
1.192kHz
Low signal amplitude
Ch1 Pk-Pk
1.52 V
1200Hz Signal


Tektronix TDS 360
Tek Run: 100kS/s Sample
Ch1 Ampl
1.08 V
Unstable histogram
Ch1 Freq
2.136kHz
Low signal amplitude
Ch1 Pk-Pk
1.36 V
2200Hz Signal

Explanation of Sine Wave Generation

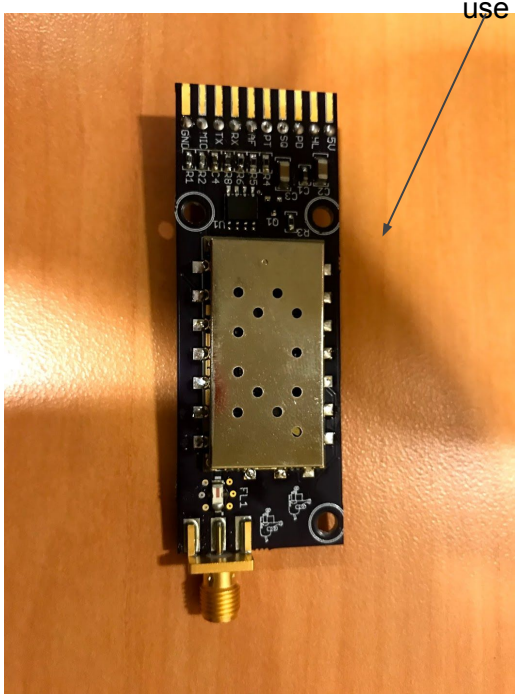


More Photos

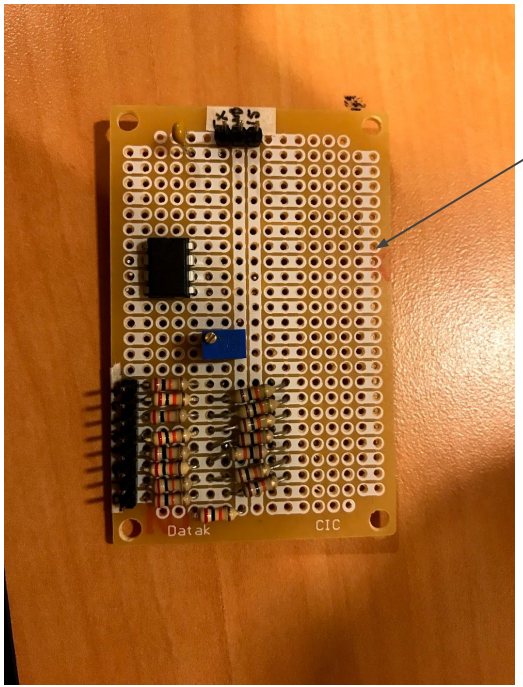


Failed Starts

Radio module we constructed from a PCB using surface mount components and the reflow oven -- no good documentation on how to use



R-2R Ladder constructed to smooth out sine wave signals -- seemed to just filter out everything but peaks.



References

“PIC-et Radio: How to Send AX.25 UI Frames Using Inexpensive PIC Microprocessors” - A very clear and concise explanation of the AX.25 Data Link Protocol

“Computation of Cyclic Redundancy Checks” - Wikipedia article with algorithms for calculating the CRC.

“APRS-IS” - The APRS Internet Service which hosts all of the APRS packets on various servers. Needed for documentation on the socket connect protocol.

“Sockets Tutorial” - Explanations and sample code of how to use the sys/socket.h library to connect to online TCP servers.

“PWM Sine Wave Generation” - A good article with some code on how to manipulate the timers to produce a duty cycle variations to approximate a sine wave.

“How to modify the PWM frequency on the Arduino” 3 part article series on all of the available PWM settings and configurations

https://github.com/tcort/va2epr-tnc - Well Documented Arduino Documentation -- Some code snippets and algorithms borrowed from here for AFSK modulation.