An Open Source Meta-Design for an APRS™ "Appliance"

Bob Bruninga WB4APR, (silent key), Tim D'Apice KA1YBS, Lynn Deffenbaugh KJ4ERJ, Jeff Hochberg W4JEW, David Kostin, John Langner WB2OSZ, Jeff Marden N1JCM, Jason Rausch K4APR, Don Rolph AB1PH, Stephen Smith N8AR, John Tarbox WA1KLI

(names are in alphabetical order by last name)

Draft of Oct. 17, 2022

Acknowledgement:

Bob Bruninga (WB4APR) has been driving APRS™ since perhaps 1982. Sadly, Bob passed during the development efforts of this project but not before heavily influencing the testing efforts to quantify the performance of this design. He was a tremendous leader in the APRS™ field and will be sorely missed.

Positioning of Effort:

This effort resulted in what might be termed a "meta-design". The effort was critically dependent on the Dire Wolf TNC software written by John Langner (WB2OSZ) and should more correctly be thought of as an application note for John's software. It is a meta-design because it has a few core principles, but the implementation can and has utilized a variety of hardware configurations and run successfully

Summary:

An APRS™ implementation, nicknamed the APRS™ "Appliance", was developed, underwent performance testing, and then field testing, including use at the Appalachian Trail Golden Packet event. The APRS™ "Appliance" offers the potential of a high-performance open-source software and open hardware design which can be readily assembled from purchased parts (only the case may need 3D printing) and can provide a low cost (~$130) alternative for deploying APRS™ implementation. It is called the APRS™ "Appliance" because once configured, one needs only to turn it on and off for everyday use.

Introduction:

APRS™, the Automatic Packet Reporting System is an amateur radio digital infrastructure built upon the AX-25 packet standard. A prototype was demonstrated in 1984 (using a Commodore VIC-20 computer). The APRS Protocol Reference Standard 1.0 dates from Aug 2000, and the technology has been widely deployed internationally and by several commercial vendors. This Technology is stress-tested yearly as part of the Appalachian Trail Golden Packet initially started by Bob Bruninga (WB4APR) and now led by Jeff Hochberg W4JEW). During the event APRS™ stations are positioned on mountain tops the length of the Appalachian Trail. The goal is to send the "Golden Packet" from Springer Mountain in Georgia to Mount Katahdin in Maine

- Appalachian Trail Golden Packet web site: https://atgoldenpacket.net/
- Appalachian Tral Golden Packet Wiki: https://atgp.wiki/

The Appalachian Trail Golden Packet traditionally depended on Kenwood TH-D72A and TM-D710 A/DM-D710GA equipment based on their well-designed APRS™ capabilities. Specifically, they can serve as digipeaters with no extra equipment. As the Kenwood radios have become less available, the group asked whether an alternative approach could be employed that provided equal or better performance. Since the deployments are man portable in some cases, they also need to be light to support human transport. This project is one answer to that question.

Conceptual Design:

Traditionally APRS™ systems were purchased as integrated commercial systems with the Terminal Node Controller (TNC) and often the global positioning system (GPS) integrated into the transceiver. Systems were constructed using hardware TNCs, computers running APRS™ software, and outboard transceivers. For this project, we evolved to the following system decomposition (see Figure 1).

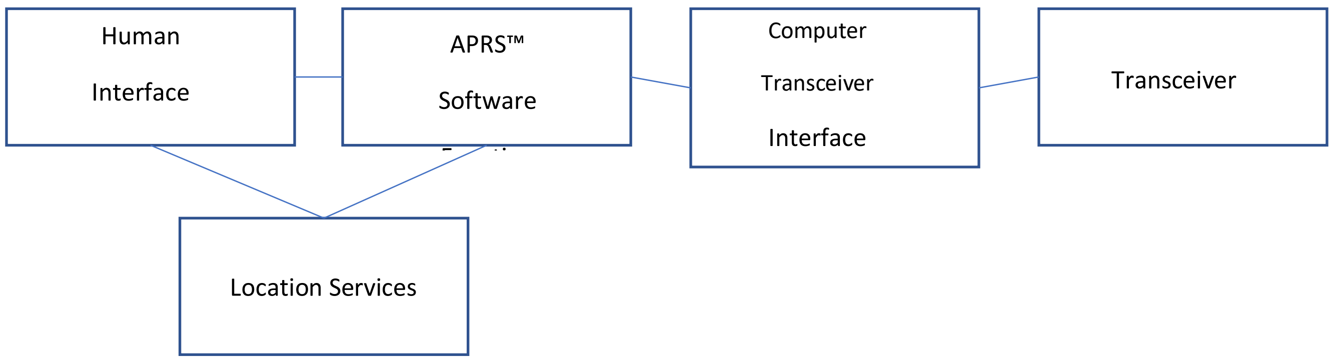


Figure 1: Block diagram of APRS™ System

These components have the following functions:

1. Human interface: this handles all human interactions with the APRS™ functionality e.g.:
    a. Listing of packets/stations heard
    b. APRS™ messaging
    c. Possibly map display of stations heard
2. APRS™ software functions:
    a. Handling APRS™ packet interactions
    b. Beaconing
    c. Digipeating
    d. Internet gateway: IGate
    e. etc.

3. Computer Transceiver interface: traditionally, a hardware TNC:
    a. Converting packets to tones for modulating the transceiver
    b. Converting received tone sequences to packets
    c. Providing push-to-talk (PTT) control
    d. Possibly providing carrier-on sense controlling (COS: is there a signal on frequency)
4. Transceiver:
    a. Packet transmission modulating RF with provided tones
    b. Demodulation of received RF signal to produce tone sequences
    c. Possibly hardware squelch carrier sensing
5. Location Services: position of APRS™ station:
    a. Traditionally
        i. Entered manually
        ii. From a GPS
    b. Potentially can be more generalized location services using multiple different sources for station location

Design Decisions:

One of the first design decisions was whether to use a hardware TNC or a software TNC. Traditionally a hardware TNC has been needed to obtain good packet performance. A prototype using Dire Wolf on a Raspberry Pi was developed and tested for performance by comparison with a Kenwood TM-D710. John Langner (WB2OSZ) performed the decoding tests documented at:

- https://github.com/wb2osz/direwolf/blob/master/doc/A-Better-APRS-Packet-Demodulator-Part-1-1200-baud.pdf
- https://github.com/wb2osz/direwolf/blob/master/doc/A-Better-APRS-Packet-Demodulator-Part-2-9600-baud.pdf

The test results suggest that Dire Wolf implementation can be superior in decoding AX.25 signals when compared to hardware TNCs, including the internal TNC in the Kenwood TM-D710A/GA.

Bob Bruninga (WB4APR) proposed the first test of the encoding capabilities of the various implementations. This used some reasonably sophisticated shielding and filtering equipment which is not generally available. Jeff Marden (N1JCM) drove an effort to compare the encoding of AX.25 packets using the APRS™ "Appliance" compared to a Kenwood TM-D710A/GA and a simplified version of Bob's testing approach. Test report:

- https://github.com/APRSFoundation/aprsappliance/blob/main/encoding-test-for-various-aprs-implementations-v0.3-20211220.pdf

This procedure used distance to reduce filtering requirement and a low-cost step attenuator to adjust levels. Based on this testing the APRS™ "Appliance" is at least as good as the Kenwood TM-D710A/GA at encoding AX.25 packets. With some conversations between Lynn Deffenbaugh (KJ4ERJ) and John Langner (WB2OSZ), some tweaks to the digipeating algorithms were implemented to meet the needs of the Appalachian Trail Golden Packet, and this development version of Dire Wolf became the core of the project.

Choice of compute engine:

Since Dire Wolf runs nicely on Linux, and Linux is well implemented on Raspberry Pis, an obvious design choice was to leverage the Raspberry PI as the compute engine. In deference to power consumption, and cost, at the suggestion of Jeff Hochberg (W4JEW) the design was based on the Raspberry Pi Zero W. The current typical APRS™ Appliance utilizes a Raspberry Pi Zero 2 W (works at both 1200 baud and 9600 baud). It will, however, work with a Raspberry Pi Zero W (1200 baud only), Pi 3, or Pi 4,

Computer to transceiver interface:

Traditionally the interface between the APRS software and the transceiver has been a hardware TNC. The hardware TNC typically controls sending the sound signals between the computer and the transceiver and provides rudimentary control of PTT and COS (squelch). With a software TNC, the interface is functionally a sound card, but a traditional sound card will requires synthesizing a PTT signal. There are a variety of possible solutions, but the AllStar interface specifically addresses this issue by providing a sound card and internal GPIO PTT signal creating an integrated interface.

- http://squirrelengineering.com/ham-radio/build-a-portable-allstar-link-hotspot-part-3-sound-card-fob-mods/

The AllStar interface can be homebrewed or is available commercially from numerous sources. Any AllStar interface should work successfully in this design, but we contacted Stephen Smith (N8AR) regarding the DINAH implementation of the AllStar interface. Initially we used Stephen's generic DINAH (cutting JP3 to disable hardware COS. It is critical to cut JP3 or, due to OS behavior, the transmit will be muted).

As the project progressed, Stephen designed a special DINAH board for this project, nicknamed the DINAH-Z, which has a form factor for stacking with the Pi Zero W/Zero 2W processor boards. The DINAH-Z has JP-3 as a physical jumper, switching between 1200 baud and 9600 baud as a physical jumper, and ferrite beads for filtering. Both perform identically for this effort.

Connection to transceiver:

The goal was to duplicate the functionality of the Kenwood TM-D710A/GA. The Kenwood TM-D710A/GA support both 1200 and 9600 baud AX.25 packet sessions. Dire Wolf will also support 1200 baud and 9600 baud (and 2400 bits-per-second and 4800 bits-per-second). The AllStar interfaces, DINAH among them, support the 8 KHz or so audio bandwidth required for 9600 baud:

John Huggins (KX4O) john.huggins.ee@gmail.com via groups.io
to atgp

Here's a view of the 9600 "audio" waveform without the convenient marker, but showing the real signal (green).

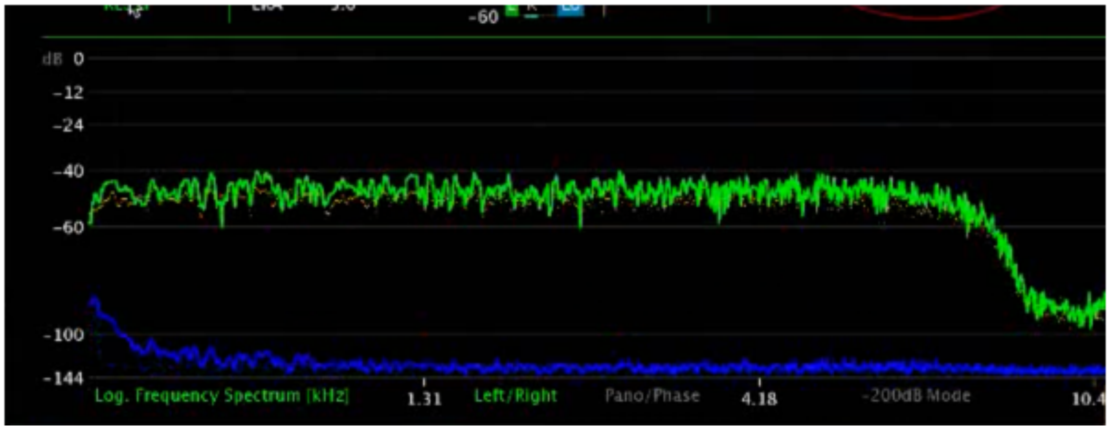


Fig. 2 audio bandwidth of 9600 baud signal: measurement by John Huggins KX4O

PLACEHOLDER

However, when we get to the transceiver, we have an issue: the audio chain has a lower cutoff of perhaps 300 Hz and an upper cutoff of perhaps between 2.5 KHz and 3 KHz. This will support 1200 baud functionality but is insufficient for 9600 baud. For 9600 baud support we need direct access to the modulator input and to the demodulator output so we can bypass any audio shaping and filtering. Many transceivers have the 6 Pin mini-DIN misnamed data port which provide direct access to the modulator and demodulator. Further these connections generally provide a standardized sound level so that a single sound setting can be used across multiple radios.

For all these reasons the 6 Pin mini-DIN interface (or equivalent) was chosen for the project. A number of radios have been tested; the results are shown below.

Radio	1200 baud	9600 baud	notes
Kenwood TM-V71A	Y	Y	
ICOM IC-208	Y	Y	
Yaesu FT-817ND	Y	Y	
ICOM IC-7000	Y	Not tested	
Alinco DR-135T	Y	N	The radio uses a DE-9 connector. The 9600 baud TX connection still has audio filtering on it and does not appear to properly support 9600 baud

Fig. 3: Radios tested with 6 Pin mini-DIN

It is expected that any radio which properly supports the 6 pin mini-DIN interface should work properly, but new radios will need to be tested.

One of the goals of the effort was to avoid requiring soldering. We found that cables from Hammadeparts to be of high quality and the Hammadeparts team does the engineering of the cable design very nicely.

- https://hammadeparts.com/shop-for-cables/ols/categories/amateur-radio-tnc-cables

This provides an additional level of flexibility in that any Hammadeparts cable designed to work with the 6 Pin mini-DIN interface can also be used with other radios (although typically at only 1200 baud since the interface is usually through the audio chain. As an example the Kenwood speaker mic cable can successfully connect the APRS™ "Appliance" to a Baofeng UV-5R, admittedly at only 1200 baud.

- https://hammadeparts.com/shop-for-cables/ols/products/baofeng-kenwood-ht-handheld-tnc-cable-to-nw-digital-draws-hat-6-Pin-mini-din-packet-port-plug

Location services:

For an APRS™ beacon to be most useful it needs as a minimum a location. Frequently this is manually entered into the APRS™ configuration. If one is moving, then a continuous stream of location information is useful. Traditionally this has been provided by GPS devices. As we move forward, it is anticipated that location can be provided by a variety of different sources (location of Wi-Fi hotspots, Bluetooth location identifiers, GPS devices etc.). Since the APRS™ "Appliance" runs on Linux, the gpsd daemon is a convenient way to abstract the Dire Wolf application from the source of location (in this case a GPS). When using the Bullseye version of Raspberry Pi OS, Dire Wolf, and the gpsd daemon, a variety of API instabilities were encountered. These have been mostly resolved by providing a version of gpsd 3.23-2 which addresses most of the issues (there is one transient problem with one type of GPS unit, but it self-corrects after one minute). A copy of the required executable can be found at:

- https://drive.google.com/file/d/15-00bbwh1bIowV8va6XCiNz899y-Lt7c/view?usp=sharing

It is anticipated that as gpsd stabilizes its support within Bullstye this issue can be eliminated.

Graphical User interface:

There are a number of APRS™ actions which occur at the user interface level for example:

- List of heard stations
- APRS™ messaging
- Map display of nearby stations
- Dynamic adjustment of beaconing rate and comments

The APRS™ "Appliance" is designed to be just that: an appliance. You turn it on, and it works, and you turn it off, and it stops.

The human interaction is supported through any one of a number of APRS™ GUI applications. APRSIS32 on Windows, aprs.fi on iOS, and APRSDroid on Android have been tested, but any application which can connect to the device over TCP on port 8001 should work. We have implemented the service discovery of avahi to support the aprs.fi iOS app.

In the default design of the APRS™ "Appliance", two Wi-Fi network adapters have been configured. The first is used to provide a hotspot if there is no network present (operating in the field perhaps) and the second is configured to connect to a local WI-FI network.

An effort was made to explore connection via Bluetooth. But inconsistencies of Bluetooth connections protocols forced us to support only TCP/IP connections initially.

Raspberry Pi OS:

If the world was perfect, one could simply download Raspberry OS, install it on a Raspberry Pi. Install the missing packages, and you would be good to go. In practice we need:

- Base Raspberry OS Bullseye version
- Development versions 1.7 D or later of Dire Wolf
- The avahi package for automatic discovery
- Automatic start at boot of Dire Wolf
- The 3.23-2 version of gpsd
- Configuration of hotspot
- Configuring the file system to avoid corruption if power is removed

Instructions for these changes are provided at:

- https://github.com/APRSFoundation/aprsappliance/configs/allstar-based-build/aprs-appliance-build-instructions-bullseye-rev-0.3-20220305.pdf

Alternatively, you can download a prebuilt executable which will only require tailoring for your call sign and local Wi-Fi configuration:

- https://drive.google.com/file/d/1ZkZQmRpCHOKm-dbgMWxH5YxCqe-QIvHT/view?usp=sharing

This requires a 16 GB micro-SD card.

Power Supplies:

Generally, any power source which can power the appropriate Raspberry Pi is suitable. The Raspberry Pi was tested with:

- Raspberry Pi wall warts
- USB ports on computers
- Powered USB hubs
- USB outlets in cars
- USB power supply using two AA batteries: Single use 1.5 volt nominal Lithium AA cells are required to maintain adequate voltage

Jason Rausch (K4APR) has also worked to successfully power these devices with a variety of buck and boost power modules.

We have had problems powering these APRS™ "Appliances" from USB power block power supplies. The nature of the problem is unclear.

But generally, any power supply which can stably power the appropriate Raspberry Pi should work.

Design evolution:

Mark 1 Figure 5:

The first prototype, called the Mark 1 prototype was the simplest version. It uses:

- A Raspberry Pi 3 or 4

- The traditional DINAH interface (figure 4 below)
- A USB/GPS puck
- Additional Wi-Fi USB adapter

These can be simply plugged together to get working hardware, which has been the most common implementation of the APRS™ "Appliance".

Mark 2 Figure 5:

The Mark 2 prototype was developed to confirm that the design would work on the Raspberry Pi Zero. The Raspberry Pi Zero has only one USB port, and this port is somewhat touchy. The addition of a USB hub resolves the USB port limitations and increases the stability of the USB subsystem. It then consists of:

- Raspberry Pi Zero W (1200 baud only) or Raspberry Pi Zero 2 W (1200/9600 baud)
- The traditional DINAH interface (figure 4 below)
- GPS puck
- Additional Wi-Fi USB adapter

And Either:

- MakerSpot 4 port Stackable USB hub HAT: https://www.amazon.com/gp/product/B01IT1TLFQ/
- 3D printed case

Or:

- Commercial extender and case – FunElec PoE Ethernet USB HUB Module Box with ABS Case: https://www.amazon.com/gp/product/B09KY11X86/

The Mark 2 is a Mark 1 but typically uses the Raspberry Pi Zero 2 W instead. The Pi Zero W will also work and runs at lower power consumption, but the processor is too slow to support 9600 baud.

Mark 3 Figure 5:

The final iteration of the design, the Mark 3, was based on a new version of the DINAH interface, the PI Zero W stackable DINAH sometimes known as the DINAH-Z (see figure 4). Its configuration is a stack consisting of:

- Raspberry Pi Zero W (1200 baud) or Pi Zero 2W (1200/9600 baud)
- MakerSpot 4 Port Stackable USB Hub HAT: USB stackable hub: https://www.amazon.com/gp/product/B01IT1TLFQ/
- Additional Wi-Fi USB adapter
- Stackable DINAH board, DINAH-Z
- A 3D printed case

And either

- USB/GPS puck

Or

- Adafruit Ultimate GPS with USB: https://www.adafruit.com/product/4279

DINAH and DINAH-Z

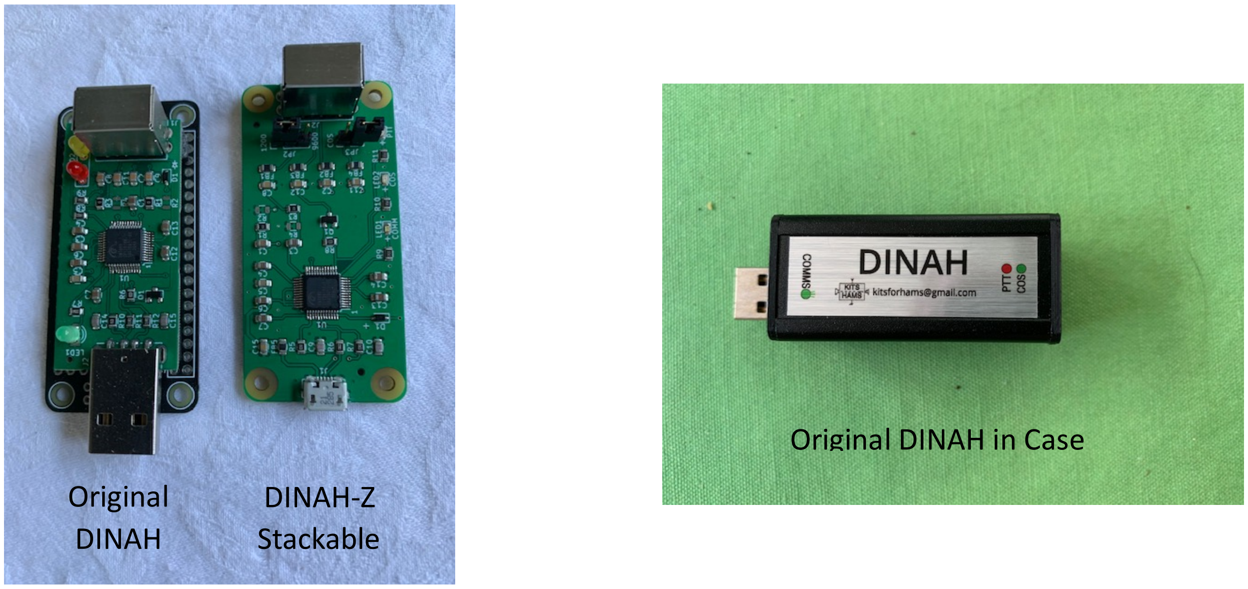
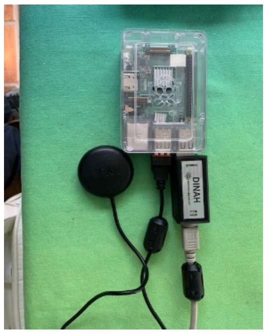
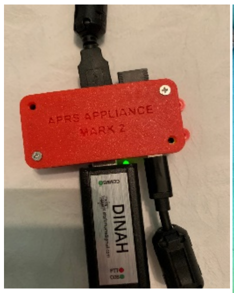


Figure 4: DINAH Interface Configurations

Mark 1



Mark 2



Home

Commercial

Mark 3

External GPS

Internal GPS

Figure 5: Mark 1, 2, and 3 APRS™ "Appliance" versions

Construction notes:

1. The APRS™ Appliance, and especially the USB subsystem, is very sensitive to RFI. All cables need to have ferrite bead on them to minimize RFI impacts.
2. The system is assembled with inexpensive stacking hardware sold in kit form by Amazon:
    a. M2.5 Series Hex Brass Spacer/Standoff + Nuts + Screws
    https://www.amazon.com/gp/product/B0756CW6Y2/

3. All internal interconnects are USB connections. Flat USB cables with offset USB connections are used. See figure 6

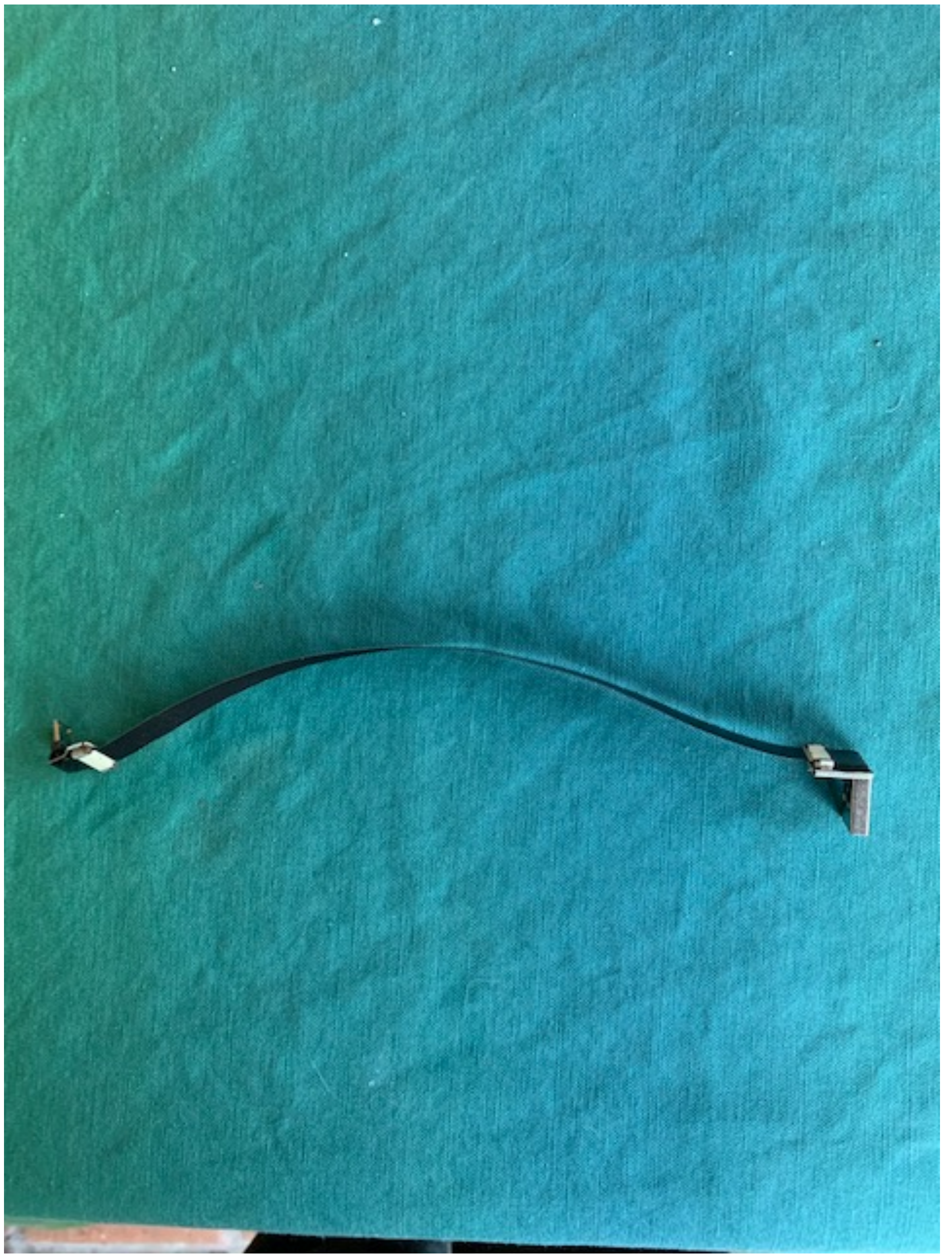


Figure 6: flat USB connectors with angled USB connectors

4. The internal Adafruit GPS unit is quite sensitive to the noise from the rest of the system.   Left unshielded, the GPS will never achieve lock. The shielding design is documented in a separate document.

Resulting System:

The resulting system is shown in figure 7.

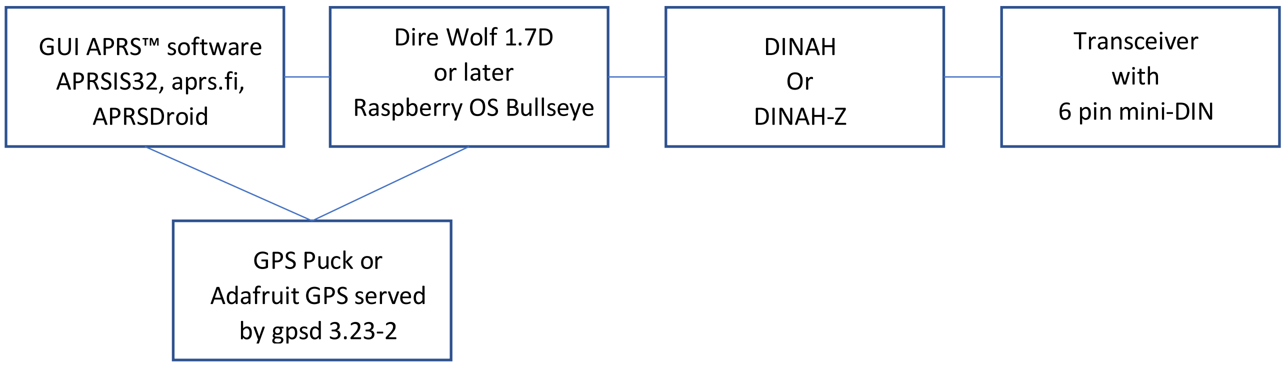


Figure 7: Block diagram of final design

Now this is the canonical design. It has been replicated with variations by about 11 people who have produced perhaps 15 implementations. In addition, different GPS units and different sound card interfaces have been successfully used. So long as the GPS works properly with gpsd and the sound card is a solid implementation of the AllStar interface, the devices have worked as expected. The key is using Dire Wolf 1.7D or higher, a good quality AllStar interface and gpsd 3.23-2.

Special Thanks:

- John Langner (WB2OSZ) for his efforts with Dire Wolf: this was THE critical part of the effort
- Bob Bruninga (WB4APR) (silent key) for his guidance on performance issues during the early part of the design
- Jeff Marden (N1JCM) for his inestimable support throughout the project and his driving of the encoding tests
- Stephen Smith (N8AR) for his guidance on using DINAH and his design and building of the DINAH-Z Raspberry Pi Zero W stackable DINAH board
- Chad Baldi for producing the design for the 3D printed cases and producing several of the cases
- Dave Kostin for printing some of the 3D printed cases
- Jason Rausch (K4APR) for his work on power supplies
- John Tarbox (WA1KLI) for his many long discussion and guidance through the process
- Jeff Hochberg (W4JEW) for including the APRS™ “Appliance effort in the Appalachian Trail Golden Packet
- Lynn Deffenbaugh (KJ4EREJ) for his technical support in all things APRS™ and for providing long term testing of the APRS™ “Appliance” under field conditions.

My special thanks to all of you. It could not have happened without the contributions of all of you!

Don Rolph (AB1PH)

Links to supporting documentation:

A variety of supporting documentation can be found at:

- atgp-aprs-appliance (Google Drive): https://drive.google.com/drive/folders/1Eovb7e1UGtZRL8z5visjHCG3oz1jon00?usp=sharing
- AllStar-Based Build (ATGP APRS Projects on Github):
- https://github.com/APRSFoundation/aprsappliance/configs/allstar-based-build

Appendix 1: APRS™ “Appliance Usage during the Appalachian Trail Golden Packet 2022:

APRS Participation for 2022 ATGP					
Count	**Person**	**Call Sign**	**Assignment**	**1200 Baud**	**9600 Baud**
1	Jeff Hochberg	W4JEW	Mount Oglethorpe SPRNGP-1		
5	Jason Rausch	K4APR	Apple Orchard Mountain AOMTP-5	Y	
9	Rory Shaffer	NJ3U	Camelback CAMLBP-9	Y	
10	Steve Bossert	K2GOG	Sam's Point SAMSPP-10	Y	
11	John Rudolph	AB1PH	Bovina BOVINP-11	Y	
12	Don Rolph	N2YP	Greylock GRYLCP-12	Y	
13	Jeff Marden	N1JCM	Equinox EQINOP-13	Y	Y
15	Tim D'Apice	KA1YBS	Katahdin KATHDP-15		Y