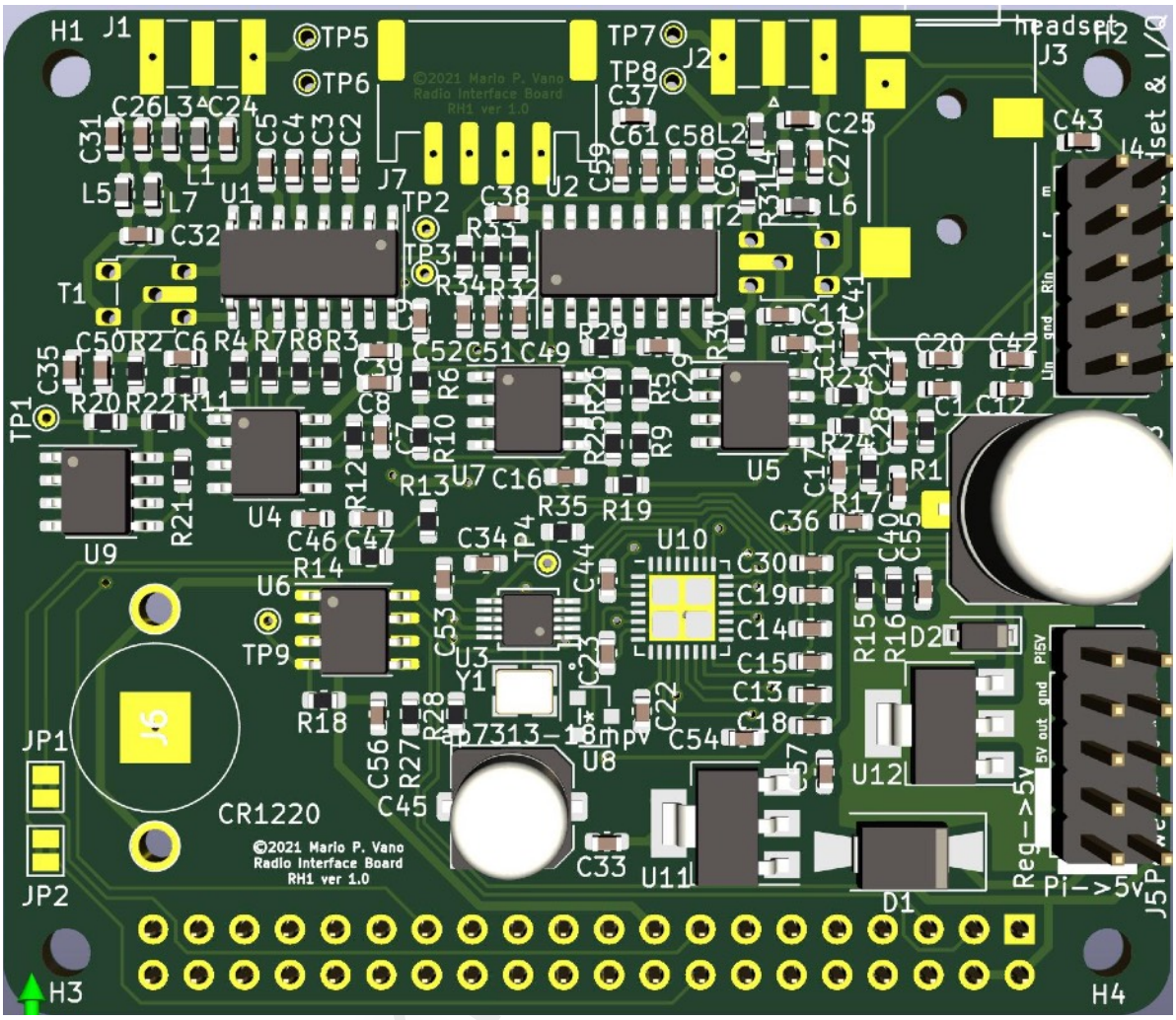


RadioHat 1.0

a User's Guide

By Mario Vano AE0GL
1 March 2022

RadioHat is a 3.5-32 Megahertz radio receiver front end and transmitting exciter for use with Raspberry Pi and other Micro-controllers.

Table of Contents

1. Overview

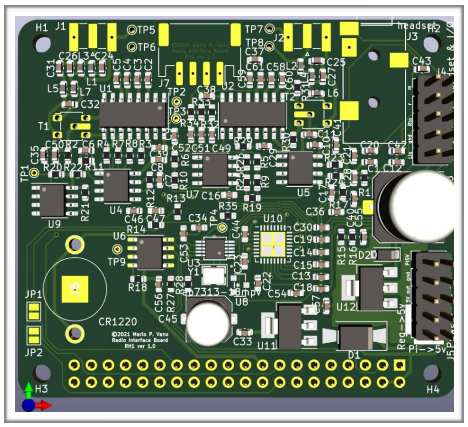
Scope

This document provides general information about the *RadioHat* "HAT" for Raspberry Pi board version 1.0, designed by Mario Vano AE0GL. It provides installation and configuration information for that board and for software needed to operate it.

Description

A Raspberry Pi "HAT" (for Hardware Attached at Top) is a peripheral board designed to stack on a Raspberry Pi computer in the space alongside the USB and Ethernet connectors. The Pi Foundation sets the requirements for building such devices.

The *RadioHat* was designed by Mario Vano, AE0GL to help bring amateur radio applications (particularly digital radio applications) to the Raspberry Pi. It incorporates many of the missing bits one needs to configure systems for operating digital modes like FT8, PSK31 and others.



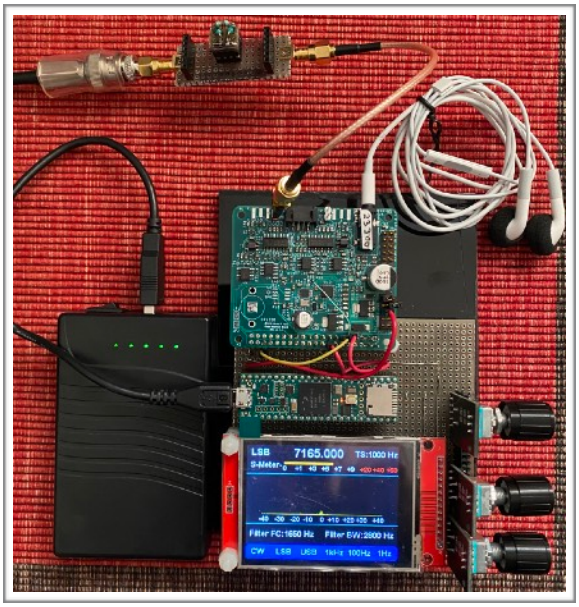
The board (shown here in an engineering rendering for clarity) contains:

- battery backed clock/calendar (optional)
- 24 bit 48 KHz full duplex audio interface
- 3 to 30 MHz zero-IF IQ demodulator
- 3 to 30 mhz zero-IF IQ modulator
- 3.3 Volt STEMMA I2C interface connector.

The *RadioHat* was intended to allow easy access to Linux tools for software development. This allows amateur application programs easy access to its devices that simplifies the development of transceivers and software.

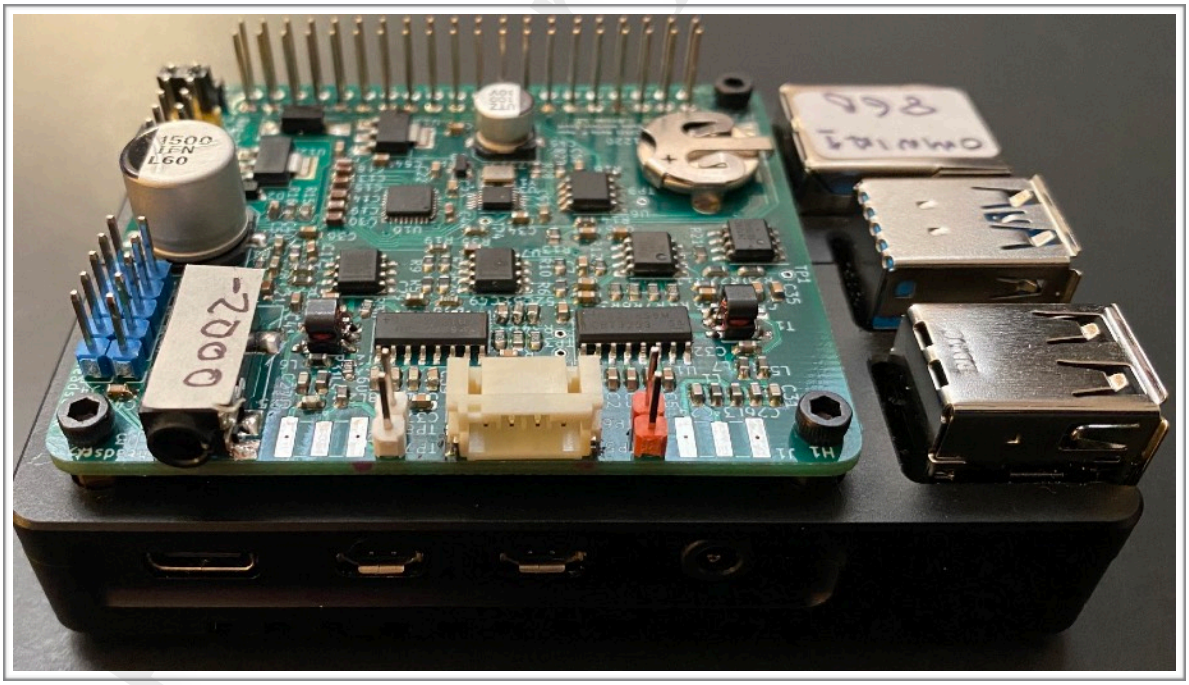
RadioHat is not a complete radio, it is a module providing the receiver and transmitting exciter components needed to to connect the host to the analog power amplifiers, filters, switches and other components used to construct complete radios.

While it was designed with the Raspberry Pi model 4B in mind, there is nothing to prevent *RadioHat* from being used with other advanced micro-controllers such as those from Arduino, Adafruit and PCJR if a suitable transition board is available or if directly cabled connections are used.



The picture on the left shows a complete 40 meter DSP receiver prototype based on a Teensy 4.1 with, touchscreen LCD and rotary encoders operating from a USB power pack.

This is a picture of a *RadioHat* fastened to a Raspberry Pi 4b in a heatsink enclosure:



Users

The *RadioHat* is designed for experimenters. It may not be particularly end user friendly to assemble or operate. By using this design, you take full responsibility for the final result you get. Hopefully, you will find that the design helps advance your projects.

In most places, government regulations require that you obtain a license and comply with published rules if you wish to connect an antenna and legally transmit radio signals. You will need an Amateur Radio Operator's License or other authorizing document. RadioHat was developed primarily to suit the needs of licensed radio operators.

It is possible to use some of the features of the board without connecting a transmitting antenna. Such use does not normally require licensing. If you are not sure whether what you want to do needs a license, check with your country's regulatory agencies first.

Disclaimer

It is very important to understand that this is not an end-user product! It is a component intended to be used in the construction of radio equipment by experimenters and OEMs and is not FCC type approved. It must be installed and operated within the limitations of and under the rules for experimental devices in Parts 15 and 97 of the FCC regulations or those appropriate to the country you live in.

Applications

This "hat" can be directly plugged in to a Raspberry PI or similar computer or it can be used with other high end micro-controllers such as the PI Pico or the Teensy® 4.0 via simple adapter boards or jumper wires. Surprisingly, some of the simplest CPUs are capable of performing the DSP operations required!

Once connected with some sort of host computer there are three styles of integration it was intended for:

- Stand-alone use, with minimal external hardware - (typically just additional band-pass or low pass filters).
    - as a GnuRadio/LiquidDSP testbed for training and software development
    - as an HF radio receiver
    - as a generator for specialized test signals
    - as a µQRP platform for WSPR and other beacon systems
    - as a simple clock/calendar and audio interface
- Use as part of a modular radio transceiver

In this picture, the Raspberry Pi and an early prototype RadioHat are shown installed in a metal cabinet along with a QRP Labs Linear Power amplifier, a set of relay switched low pass filters, a VSWR detection module, a voltage regulator card for feeding the Pi from 12 volts and a level shifting module for driving the relay and power amplifier control.

Here, everything was controlled via direct GPIO manipulation except for the VSWR bridge which was read via the Pi I2C pins.

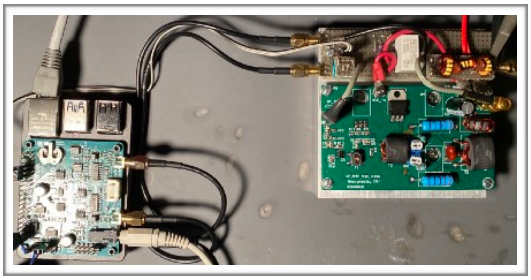
This complete 5 Watt transceiver has been operated in the field with several different user interface devices as consoles.

It has been used with a notebook computer by means of an ethernet cable, by an iPad via a USB cable using the Pi as an RNDIS "gadget", and also without any computer at all with a touchscreen HDMI display and keyboard. It works well from a 12 Volt battery pack.

- Use as part of a system involving additional external devices. Here are some examples:

  - as a driver and receiver for outboard transverter modules. Little other hardware is usually required. This example shows the Hat installed on a Pi4 that is connected to a "Ukrainian" transverter module with two coaxial cables and a GPIO line for transmit control. This transverter was modified by adding an optical isolator with a 1K series resistor inside its case. (wrong picture)

  - as a transceiver system built with a separate package containing the power amplifier and filter, T/R switch, and power supply. Typically, in addition to the hat's radio input and output cables, this would use a GPIO or STEMMA connected cable for controlling the rest of the components.



2. Technical Details

Interfaces to the host computer.

This hat communicates with its host computer using standard hardware methods:

- **I2C** is used to control the VFO frequency, the Audio Interface, the optional clock/calendar and the configuration EEPROM and it is also passed through to a "STEMMA" external bus connector for controlling other hardware. The I2C host hardware and unix device driver support are all built into the Raspbian OS and require no other driver.

- **I2S** is used to handle high speed DMA transfers to and from the Audio Interface. The hardware and low level driver support for this is built into the Raspbian OS.

- **GPIO** is only used by this Hat to switch the transmit and receive mixers on and off via GPIO 22 and 23 on pins 15 and 16 respectively. A logic Low on each pin will enable the associated mixer. The signals are open collector with pull ups available that default to disabling them.

  The two GPIOs required for these functions can be disconnected and reassigned by cutting a pair of jumper traces (JP1 and JP2 respectively) and adding patch wires to connect the pads to alternate pins on the Pi interface connector.

- **Power** input and output connections are provided on a 10 pin header to ensure routing of adequate clean power between this device and the host computer. These can power the board from the Raspberry Pi's 5V supply (usually very noisy) or from an input pin that can be fed from an 8-15 volt source (recommended).

  The power header also provide pins to feed the Raspberry Pi from an external 5.1 Volt power supply but use of these requires extreme caution to prevent damage! Although this is the recommended way to back-power the Pi, these pins are DIRECTLY connected to the its internal 5 Volt bus with no current limiting, reverse flow or overvoltage protection. If you do not intend to use these pins, I suggest keeping a bi-pin jumper connector in place on the two 5V pins to prevent accidental contact as the Pi is extremely intolerant of incorrect voltage feeds or shorts to ground on them.

Interfaces to external devices

- **RF input** is available by means of an optional SMA connector, by a directly soldered cable, or by .025 inch pin connectors. For some applications this can be directly connected to a receiving antenna.

  It is a 50 ohm input, band pass limited to 3.5-32 MHz to help reject local broadcast

transmitter signals, but additional filtering will be required to prevent spurious signals from being received at harmonics of the direct conversion frequency, especially below 15 MHz. In transceiver applications, these filters are usually the same ones needed to filter the transmitter output.

Note that both of the RF input output connectors and all filter components ahead of the input transformer and after the output transformer are isolated from the circuit ground. This is can be useful in controlling audio and RF sneak paths. RF ground connections were not included in the PCB design because it is much easier to add them yourself where your design requires them than to try to remove them if they are in the wrong place!

These portions of the circuitry will need to be grounded *somewhere*. If your system's *somewhere* turns out to be at the *RadioHat* (rather than at the antenna switch, for example), you may easily connect a ground to the bottom PCB layer. Do this by carefully removing a small area of the solder mask adjacent to the SMA socket outer pads and then soldering a piece of wire to bridge the gap between the pad and the exposed copper.

- **RF output** can likewise be provided by means of an optional SMA connector, by directly soldered cables, or by pin connectors. Around to +6 DBM at 50 ohms is available depending on the transmitted modes supported by the host computer's software. There is a simple 32 MHz low pass filter provided to simplify operation with transverters, but for most other uses additional filtering will be required. Practical use as a transmitter will require outboard circuitry to further filter and amplify the signals and to share an antenna connector with the receiver.

  Like the RF Input, the output is floating - see the notes on in the preceding section.

  It is very important to understand that quadrature switching modulators usually produce ***legally unacceptable*** levels of harmonic output. This aggravates out-of-passband IMD production in subsequent amplifiers. The built-in low pass filter should allow 10 meter amplifiers to be driven safely, but below that frequency you will need to add an additional filter before your power amplifier (if it does not already include one).

- **STEMMA-4** is an educational and hobbyist method for "daisy-chain" connecting I2C devices to host computers and each other. This connector can be used for any purpose desired, but one way to use it is to control and monitor the rest of the circuits one might add to build a complete radio system.

  Such things as band switching relays, VSWR monitoring bridges and power amplifier controls can be conveniently connected this way. This is especially useful if the Hat is used with devices located in a separate box from the host computer.

Care is required to avoid address conflicts since most I2C devices have predefined bus addresses built in to them! The Raspbian OS provides utility programs for managing this bus and scanning it for free addresses.

This bus is powered and interfaced for use only with 3.3 Volt I2C devices. Older 5V devices will require external power sources and level shifting circuits to be added.

Because I2C is a serial bus, this connector can also be used in applications where the board is run without a Raspberry PI as it provides external I2C access to the optional clock/calendar, VFO and sound card control registers from an I2C host device. Note however, that the clock/calendar is normally powered directly from the 40 pin Raspberry PI connector. 3.3 Volt power to operate it must be supplied by feeding this pin from another source you intend to use the board without a Raspberry PI.

- **IQ modulator line level input**
- **IQ demodulator line level output**

These are provided for testing software and also for applications where access to the modulator and demodulator are needed from an external system like a desktop or notebook computer. The phase characteristics of this IQ input are limited by the loading of the built in audio card inputs, but it is adequate for most use.

More demanding uses may require removing the coupling capacitors feeding the modulator from the two SGTL5000 line outputs, but quite acceptable performance is possible without doing this. There's no need at all to do anything like this to the demodulator outputs.

- **Line level audio output**
- **Headset audio input and output**

With suitable software, these connector pins allow use of the Hat as a 48 KHz full duplex 20 bit interface (see the note above about the IQ modulators loading the inputs).

The Headset audio I/O is normally intended to provide the microphone and speaker connections for Amateur Radio voice operation, but is available for other uses, as well.

In transceiver applications, the controlling software will need to manipulate the sound card routing, muting and volume registers to reconfigure the card when switching between receiving and transmitting, and perform any needed morse code signal shaping.

- **Power supply inputs**

The board should be connected to a clean outboard power supply for best performance, but for some uses it can be operated from the host computer's 5 Volt bus if noise is not a problem.

A connection to provide "Pass through" power from an external source is also available. This allows running the Raspberry Pi without a separate power cube by using the *radiohat* as a power junction block. This is especially useful when operating the Pi as an RNDIS gadget. Otherwise, the Pi would need to be fully powered by the RNDIS host since its normal power source is the same USB-c connector that works best for RNDIS.

3. Software

Communicating with the Hat

There are four ways that information is exchanged between the Hat and the Host computer's software:

1. Startup EEPROM

   The Pi Foundation requires all devices sold as "Hats" to provide an onboard ROM device identifying the board and describing the resources it needs and can provide. This information is used by Raspbian and other Pi operating systems at the time they are loaded and is normally not available after that.

   Since this is not designed for end users, programming the device is the system builder's obligation - there is no default information provided. If you will be using it. Your own ID information and the details of the rest of your system must be added to it. It is not currently required for prototype systems.

   Tools are provided by the Pi Foundation for self assigning a GUID, building the needed files, and transferring them into the EEPROM. The test point labelled TP1 must be grounded to allow the Pi to write to this chip for programming it.

2. Direct access

   Because the Raspberry Pi computer is intended for use by "Makers" and OEMs, there is nothing stopping direct access from user level programs to all the devices on this card except for the configuration ROM without using "sudo" or other root privileges.

   Nearly complete control and operation of these devices is possible simply by manipulating them directly from application programs, and this is often the best way to do so. Of course, this places the burden of avoiding conflicts on the end user and trusted programmers, since the operating system does not protect the devices from improper sharing.

   Direct access is enforced by membership in various authorization "groups" automatically created by the Raspbian (or other) OS and this configuration can be changed as needed.

3. Inline plug-in codecs for Raspbian generic audio driver

   At the moment, the preferred codec access method is a hybrid one using Rasp-

bian's built in "Generic" I2S audio input and output drivers for handling DMA audio and direct access to peripheral registers for all the other control needed by application programs.

In combination with the Generic Audio Drivers, it is also possible to use Raspbian's ALSA and PULSE audio systems to write and use transient codec software inline with audio streams to handle the messy details of DSP modulation and demodulation in an orderly fashion.

Programming notes

1. Raspbian configuration for programming - build essentials

   You will need to use the Raspbian package manager or other tools to install such development tools as your intended use requires. Ordinarily the basic build essentials used with the Pi are adequate for GnuRadio, C, C++ and Python development.

2. Clock/Calendar Option support

   The needed clock/calendar device driver is included with Raspbian and can be loaded at boot time by the following line to the /boot/*config.txt* file.

   You may also wish to modify the Raspbian startup scripts to make better use of the device. PiMoroni and Adafruit tutorials for their clock/calendar products are good sources for more information on the subject.

   NOTE THAT the Raspbian clock system sets the RTC device hardware time for the DS3231MZ device incorrectly by 0.5 second. This offset seems to be deliberate and is there for reasons based on the design of early PC hardware. It also seems nearly impossible to remove this offset as Raspbian's several different patched-together timekeeping systems all seem to enforce it in different places.

   This error appears during boot-up and disappears a few minutes after connection to a network after NTP fully completes its synchronization. If you require more accurate timing when NOT connected to a network, then you will need to find a way to deal with this problem.

   When in the field, I often briefly enable Wifi after boot-up for long enough so that NTP can sync to a network connection from my cellular phone and then disable wifi to eliminate the RF noise generated by the wifi.

3. Changing the ID EEPROM contents

The startup EEPROM on this hat is normally write protected unless TP 1 is connected to ground by a jumper. If you need to learn more about how this device is used, consult the documentation and tools provided by the Raspberry Pi Foundation at https://github.com/raspberrypi/hats

4. Si5351 programming

   The example programs all use versions of the Etherkit Arduino libraries ported to the Pi for access to the chip but the following additional change is needed to the header file for these or other libraries.

   Many libraries limit the lowest frequency that can be generated by the PLLs. This may need to be adjusted downward for proper quadrature operation as low as 3.5 Mhz. Extensive use of this part in other designs has proven this to be reliable way to use the part.

   The Etherkit Arduino library requires the following change in the header file "si5351.h":

   ```
   //#define SI5351_PLL_VCO_MIN 600000000
   #define SI5351_PLL_VCO_MIN 380000000        // mpv change
   ```

5. NXP SGTL5000 startup, driver and register controls

   Refer to the *Radiohat* examples and the extensively documented one in the NXP (formerly Freescale) data sheet.

   The SGTL5000 is normally clocked directly from CLK3 of the Si5351 oscillator chip. If you wish to provide external clocking (as is typical of use with the Teensy micro-controller) then you will need to carefully remove R35, which is a zero ohm resistor and provide your own clock signal via a wire connected to the end of R35 that goes to pin 21 of the SGTL5000.

   If you are not using a Raspberry pi, I suggest terminating that wire on one of J8's unused pins to provide strain relief for the external connection. I normally use pin 11.

   Some external clock sources (like Teensy GPIO pins) are unable to drive this board without adding a 100 ohm series resistor close to the driving source and keeping the wiring very short.

   Note that if you are using an external clock source, the SGTL5000 will not appear as an I2C device or an I2S device until a suitable clock is provided by programming and enabling your clock source.

6. Transmit/Receive switching

   Please refer to the included "pitrans" example.

7. Audio mixer control requirements

   Please refer to the included "pitrans" example.

8. GPIO control - WiringPi vs SysFS and other methods

   The default version of WiringPi installed for the Pi 4 is currently broken badly. According to it's author, it can be updated if you follow these instructions. This is NOT OPTIONAL if you wish to use it:

   ```
   cd /tmp
   wget https://project-downloads.drogon.net/wiringpi-latest.deb
   sudo dpkg -i wiringpi-latest.deb
   ```

9. I2C programming

   To "future-proof" your programs, we suggest using the built-in Raspbian I2C API which is available to C programs that use the following headers and that requires no additional external libraries:

   ```
   #include <linux/i2c-dev.h>
   #include <sys/ioctl.h>
   ```

10. I2S device overlays - Open source example

    The Raspbian operating system provides a built-in generic driver for I2S devices that can be enabled by compiling and installing the following open source device overlay. It provides full duplex 48KHz 24 bit I2S data transfer DMA operation of the SGTL5000 under the name "GenericStereoAu" once it has been initialized by other software.

    This open source overlay is available from GitHub and works well:

    https://github.com/AkiyukiOkayasu/RaspberryPi_I2S_Slave

    It's also enclosed with the support files and to install it, it needs to be copied to the correct location on the boot volume. Here's one way to do that:

    ```
    cd ~/radiohat/RPI as\ SLAVE/ IIS\ setup/
    cd RaspberryPi_I2S_Slave-master/genericstereoaudiocodec.dtbo
    sudo cp genericstereoaudiocodec.dtbo  /boot/overlays/genericstereoaudiocodec.dtbo
    ```

The example program called "pitrans" provided in the support package for Radiohat can be used to initialize the SGTL5000. It has an option to do this and then immediately exit by running it with the "-i" option. This only needs to be done once after each reboot of the OS.

The program provides some other examples of transceiver control and parameter adjustment of the SGTL5000 and is used to operate the VFO and audio mixer in most of the demonstration programs provided.

11. GnuRadio hints

- As noted above, this hat's audio will not function properly until it has been initialized by a separate device overlay, program and/or driver.

- frequency control, transmit/receive, keying etc. need to be handled by an external program or by writing your own Python Block to talk to the hat. Simple python blocks to do these things can be written to be executed in the GnuRadio Companion environment without requiring any external linkage.

- The Gnu Radio ALSA device names for this card's 48KHz 32 bit input and output are currently:

hw:GenericStereoAu,1 for IQ input from QSD Demodulator
hw:GenericStereoAu,0 for IQ output to QSM Modulator

- One way to use the hat with the Jack Audio Connection kit is to configure the hat as the ALSA audio card for jack in the configuration dialog using the above names and then add the following two lines to the end of the configuration file / etc/gnuradio/conf.d/gr-audio-jack.conf :

default_input_device = gr_source
default_output_device = gr_sink

You must also edit the default audio configuration in /etc/gnuradio/conf.d/gr-audio.conf to change the line that says:

audio_module = auto

to say:

audio_module = jack

devices named gr_source and gr_sink will then be available for jack input and output connections. This will disable other audio connection methods in GnuRadio.

Don't forget to restore the original line if you want to go back to connecting to the board in other ways.

12. WSJT-X hints

While the WSJT-X programs appears to provide support for Jack IO, that support is broken and cannot be used because WSJT-X expects audio in a format Jack cannot provide. The WSJT-X developers currently have stated they have no plans to fix this as they see it as being a problem in the QT and PortAudio libraries and not something they are responsible for.

One way to get around this problem is to employ the Pulse audio module that can be installed using:

sudo apt-get install pulse-audio-module-jack

If you then start jack with the audio devices for input and output being those provided by the hat, the commands

pacmd load-module module-jack-source
pacmd load-module module-jack-sink

will create a pair of jack devices that appear in the WSJT-X Audio configuration dialog as *jackin* and *jackout*.

With simple jackd patches, this allows testing WSJT-X receiving and transmitting double sideband signals (preferably transmitting to a dummy load). With suitable codecs in line, single sideband signal can be produced and received.

A USB headset can be added to the patch by using:

```
alsa_in -d hw:1,0 &        # device number may vary
alsa _out -d hw:1,0        # device number may vary
```

Very simple rig control (T/R switching only) is possible by telling WSJT-X to use the raspberry Pi Serial Port/s DTR signal to key the transmitter and then monitoring the appropriate GPIO pin in the Hat control software. One problem with this method is that the default state for the pin will key the transmitter until the pin and WSJT-X are initialized. You must be careful of how you sequence start-up.

Obviously a better rig control method is needed and many approaches are possible, but this will work for simple tests as long as you keep the transmitter above 1500Hz, since WSJT-X relies on being able to shift the transmitter frequency to keep unwanted audio harmonics from appearing in the transmitted

signal for tones below that frequency.

Step-by-step instructions for doing this are beyond the scope of this document. Refer to "*Getting Started with Radiohat Digital Modes*" for more detailed instructions.

13. Rig Control

    To enable the serial port:

    Start raspi-config: sudo raspi-config.
    Select option 3 - Interface Options.
    Select option P6 - Serial Port
    At the prompt Would you like a login shell to be accessible over serial?

        answer 'No'

    At the prompt Would you like the serial port hardware to be enabled?

        answer 'Yes'

    Leave raspi-config and reboot the Pi for changes to take effect. You will also need to properly initialize the protocol signal GPIO lines to make use them.

4. Example and test software

- These additional files are provided in the support folder download just to get you started. They are simplified examples of the things that are possible using the *RadioHat*. None of these fragments are intended to be complete programs. Remember that *RadioHat* is not an end-user product - it's only a component in the system you build.

  - pitrans.cpp

    This is a sprawling C++ program that is useful in testing the board and can be useful as example code for controlling the *RadiHat* devices. It also can b run with the command line option *-i* to initialize the devices after boot up.

    The VFO is controlled by incrementing and decrementing the VFO frequency with up and down cursor keys and by changing the weighting of those adjustments by moving the cursor left and right between digits. Most mouse wheels and touch pad devices can also vary these values when the cursor is positioned over them.

    Other internal values the program uses to control the mixer gains are shown in a row at the top of the display area and can be incremented or decremented using the keys shown. Upper case numbers increment values, lower case ones decrement them.

    pitrans depends on the *ncurses5* and *WiringPi* libraries (which must be located and downloaded) and generally needs recompilation before use to adapt to the environment it is running in.

    If a file named "CALFACTOR.txt" is present in the folder containing the binary file and if that file contains a single line with a decimal long value in ascii on it that value will be used by the si5351 library as a crystal correction factor in parts per billion.

    If your board has a sticker with such a number on it attached to the top side of the headset jack that number is a good starting point to use as it was determined during final test of the board.

    Some of the enclosed shell scripts use pitrans under different names such as "pitrans1" or "pitrans2". This was done deliberately to allow variant versions to be used by the shell scripts. If you have not created any modified versions of pitrans then it is best to make copies of (or links to) your compiled binary (usually created by the compiler with the name "a.out") and name them pitrans, pitrans1 and pitrans2 to ensure that they are found by such scripts.

  - I2S audio device overlay

    As discussed earlier, *Radiohat* presently uses generic I2S drivers that are built in to Raspbian, but to enable them a Device Tree Overlay file is required. This open source

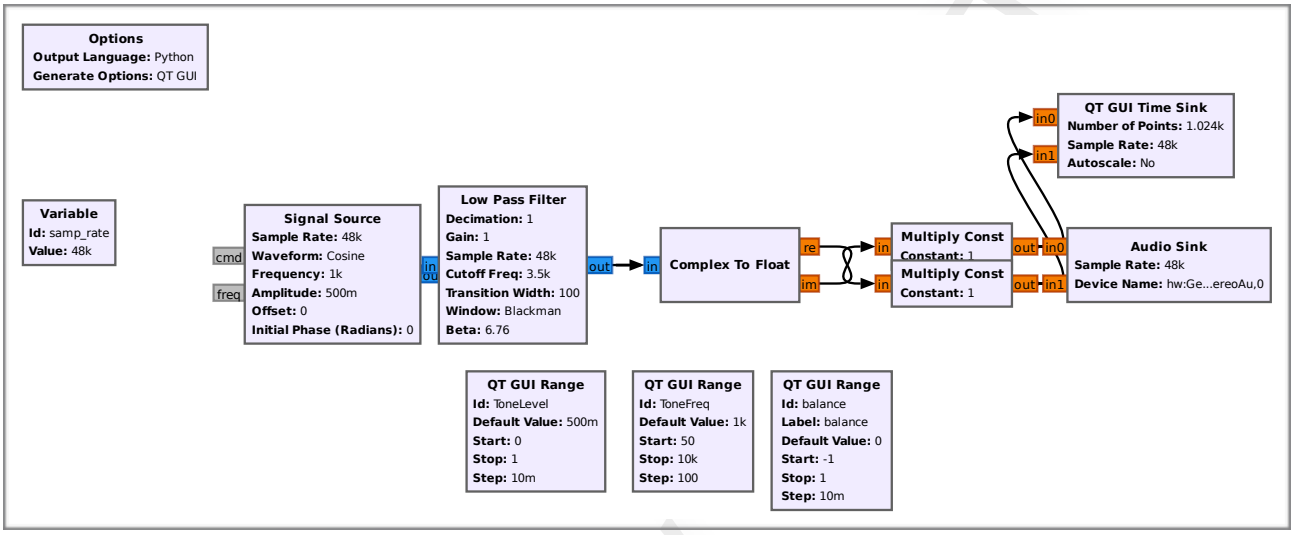
overlay is available from GitHub and works well:

https://github.com/AkiyukiOkayasu/RaspberryPi_I2S_Slave

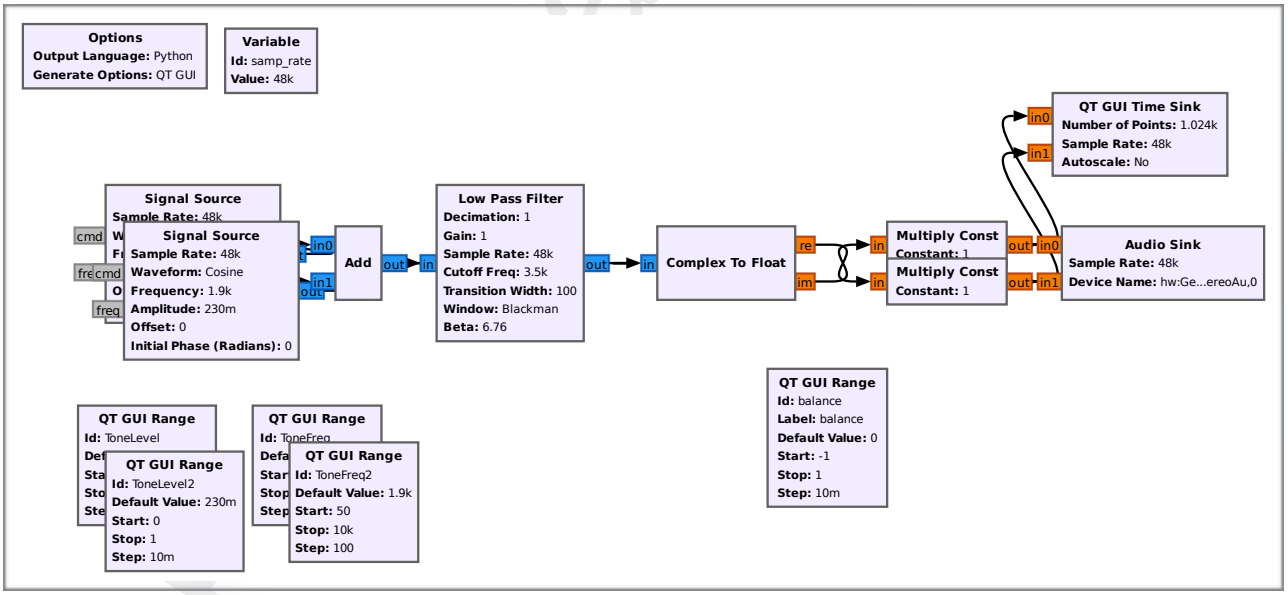
It's currently enclosed with the support file download and to install it, it needs to be copied to the correct location on the boot volume. Here's one way to do that:

```
cd ~/radiohat/RPI as\ SLAVE/ IIS\ setup/
cd RaspberryPi_I2S_Slave-master/genericstereoaudiocodec.dtbo
sudo cp genericstereoaudiocodec.dtbo  /boot/overlays/genericstereoaudiocodec.dtbo
```

- These are two Gnu Radio models for Quadrature test oscillators that can be used to perform single tone and two tone testing of the modulator and subsequent power amplifiers. They are enclosed in the support folder. They both require the use of the pitrans program to provide VFO and Audio Routing and T/R switching.

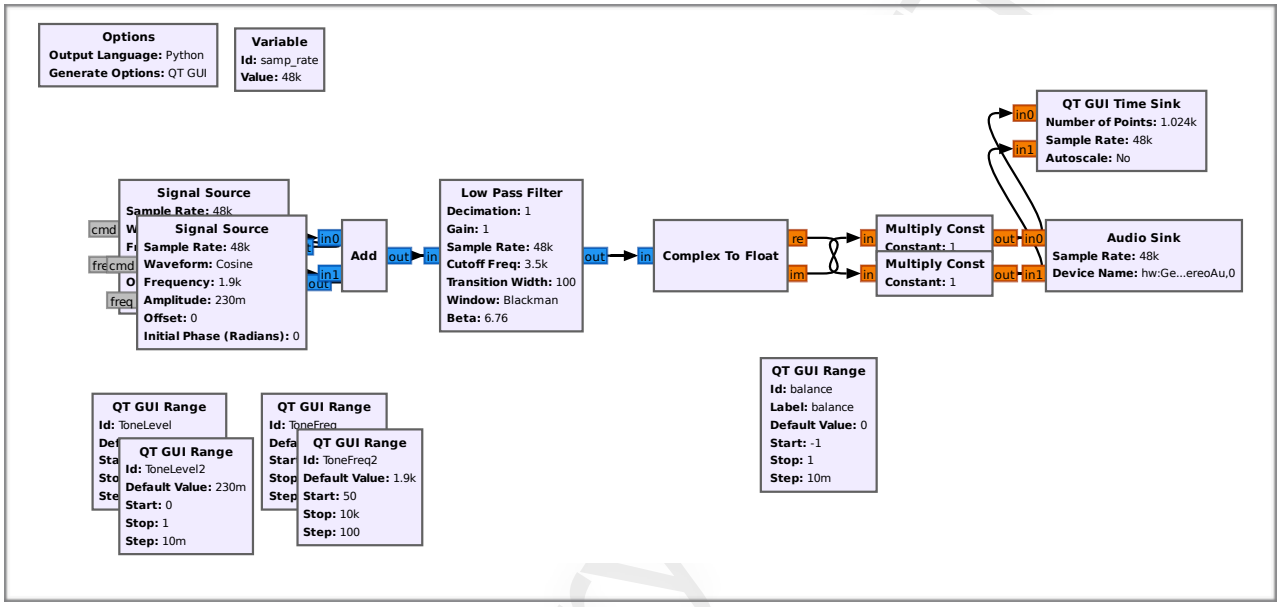


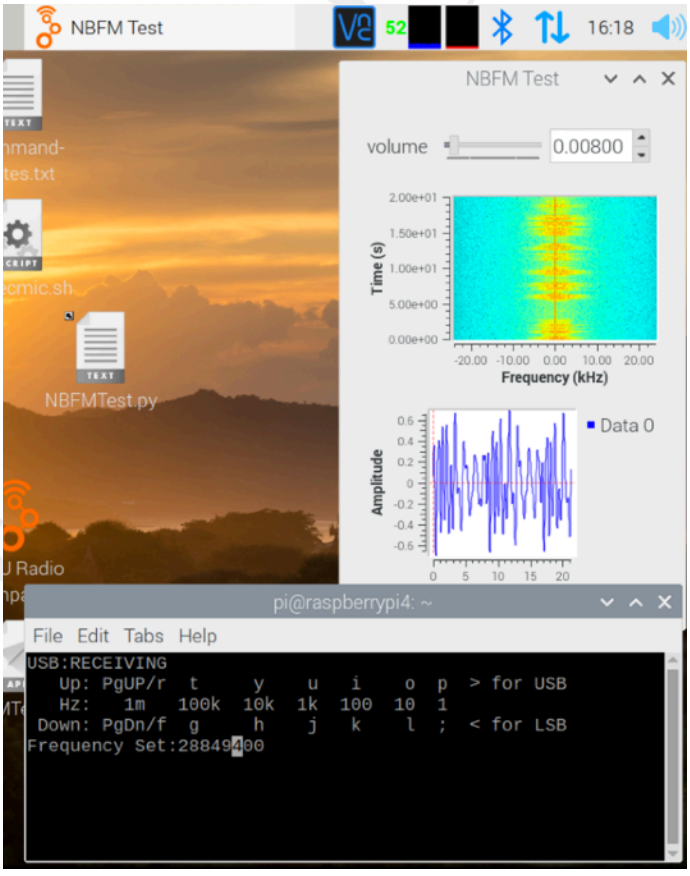
osc.grc



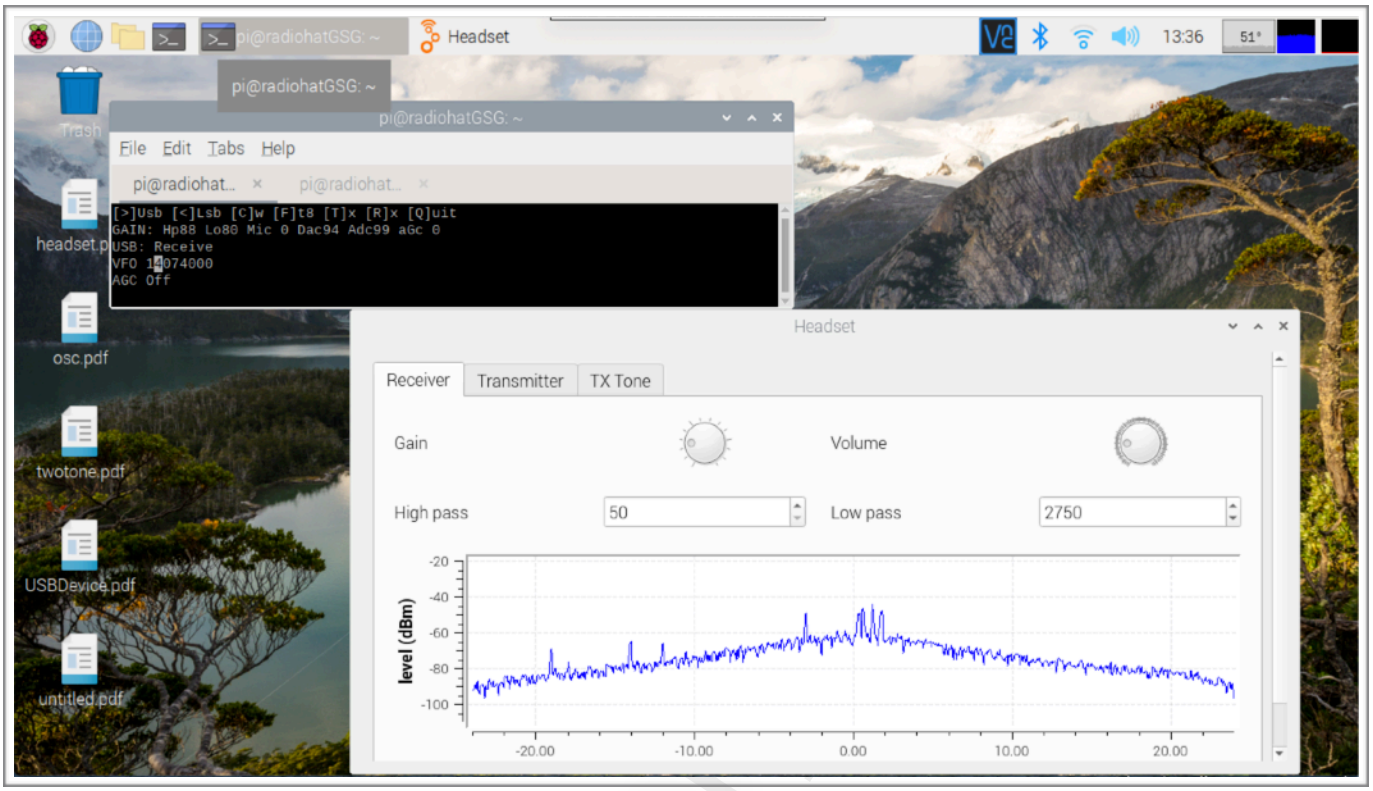
twotone.grc

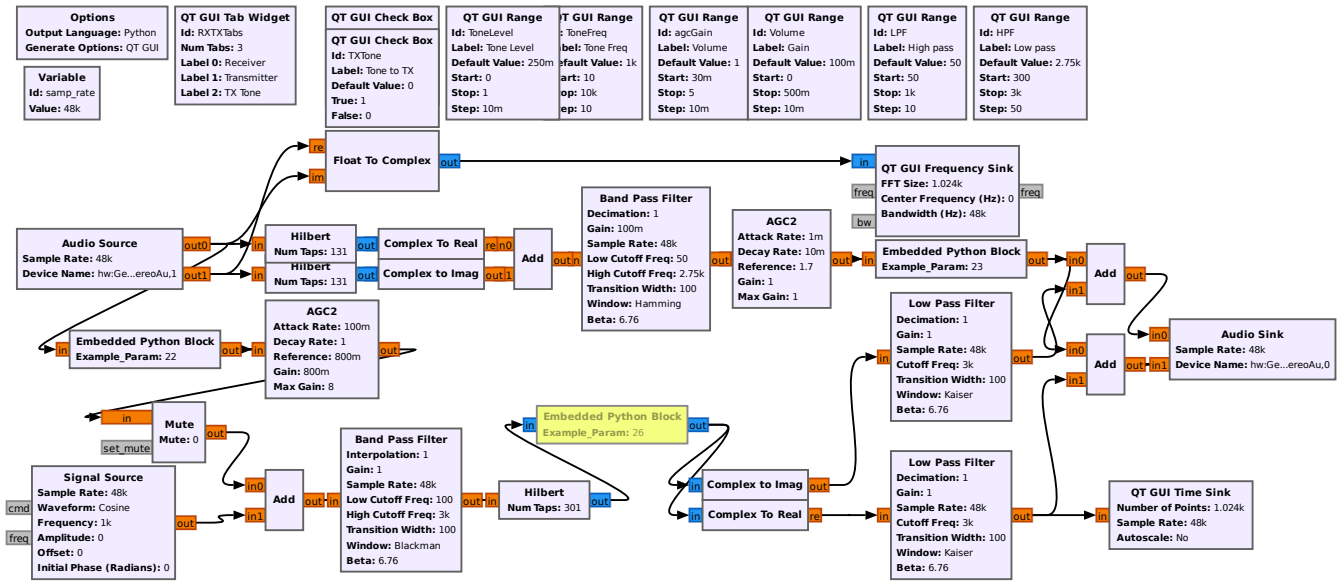
- This example uses pitrans and a simple Gnu Radio model to listen to local repeaters via an inexpensive transverter.





- This is a pitrans and GRC based C++ SSB transceiver.





- Other software that is useful with *RadioHat*:
  - Gnu Radio version 3.7 or higher
  - LiquidDSP (and other DSP Libraries).
  - Jack Audio Connection Kit

5. Hardware Configuration

In some cases, it may be desirable to replace the supplied components with alternatives.

T1 and T2 RF transformers

These SMD broadband transformers generally provide good performance, but their turns ratio and core materials may not be optimal for all uses.

If you wish to experiment with the some other form of transformers, such as hand-wound toroids, you will need to remove the supplied SMD transformers using a hot air desoldering tool and then install your toroids by drawing their pigtail leads down tight to the top of the board and soldering the wires at the back. It's probably best to also secure your coils from strain and vibration by using a small drop of hot-melt glue where they meet the board to fasten them. The plated-through holes are limited to AWG 26 or smaller wires. Do not attempt to enlarge them or you will damage the plating.

Demodulator Bandwidth Limiters

The receiver front end was designed to try to minimize the received bandwidth to only that needed for the actual communication. Some transceiver builders prefer that this bandwidth be widened to allow flatter response for a waterfall or spectrum display. C2, C3, C4 and C5 control the switched filter behavior. They are currently set to 430 Nf. C6 and C7 control the frequency at which the receiver preamp begins to roll off. They are currently set to 4.7 Nf.

For use with a wideband spectrum display you may wish to change those components to values like 100Nf and 1Nf. If you make the values of C6 and C7 too small, the extremely wide band performance of modern op amps may result in problems from AM broadcast signals (despite the built-in bandpass filter).

Demodulator Gain Feedback Resistors

The Demodulator gain was adjusted to attempt to suit most use without additional preamplification required. Increasing the gain will result in more overloading from strong signals at night. It is primarily controlled by R29, R32, R33 and R34. They are currently set to 50 ohms. Reducing their value will increase the gain. Values between 0 and 200 ohms have been used in other switching demodulator designs of this type. Increase them to decrease the gain. Ensure that they are well matched for best single sideband suppression.

One can also reduce the front end gain by adding a pad ahead of the receiver input, or by reducing the value of R11 and R12 which set the op-amp gain. If you change R11 and R12, you will also be affecting the bandwidth of the receiver and may need to re-

set the values of C6 and C7 as well.

SMA connectors

The optional SMA connectors may interfere with certain mounting configurations where the edge of the Raspberry Pi is close to the enclosure side.

In such cases you may wish to solder small diameter coaxial cable pigtails directly to the SMA connector pads or connect to the test point locations next to the SMA pads using female "Dupont" connectors in two pin shells.

If you wish to install one or more of the SMA connectors, place one in position at the edge of the board over the pads with the center pin on top of the board. Lightly solder only the center pin. Then carefully check the connector's position before soldering any more pins. Reheat the center pin as needed to move the connector around until it is properly aligned with the board, then solder one ground pin to the top side of the board.

If all goes well, you can now go back to the center pin and add more solder for a secure connection. Repeat for the other connector if you are installing it.

If you think you may ever want to remove the connector, it's best not to solder the rest of the ground pins unless they are absolutely needed for mechanical strength.

40 pin connector

If you need to increase clearance between the hat and the Pi main board for cooling, access to the other connectors on top of the Pi, or because you have a case with clearance problems, this can be achieved by installing by plugging a 40 pin dual row connector with long solder pins into it for use as a pass-through extender. These connectors are available from Pi accessory suppliers as this is a common problem with certain case designs.

STEMMA-4 connector

While the JST connector for STEMMA-4 cabling is pre-installed, there's usually not a problem with it obstructing clearances when the hat is installed inside a cabinet. However, access to that connector may be impossible if the Pi is installed too close to a cabinet side.

If you will be using the board this way, do not try to replace it with a vertical SMD connector! Installing a vertical JST connector to a horizontal connector footprint will reverse the connector pinout!

Instead, simply use female "Dupont" style pins or other .1 inch spaced connectors to access the pi I2C pins 1,3,5 and 9 directly on the 40 pin connecter, since these are the same wiring nodes that are available on the STEMMA connector except that the 3.3V power will now be supplied from the Raspberry Pi.

Bi-pin power jumper

A 2 pin .1 inch jumper block is used to set the boards 5V power configuration.

As noted elsewhere, the 5V power from the Pi is convenient to use, but noisy enough to seriously disrupt receiver operation and might even create noise in the transmitted signal. This is especially true if you are using the Pi's Wifi adapter for networking as it seems to create huge power transients at irregular intervals, especially when in use with 5 GHz networks.

If at all possible, provide external DC power from a clean source of between 8 and 15 volts to the designated pins 6(+) and 8 (-)on J5 and reset the jumpers to use the on board analog regulator to reduce. The current needed from this supply is typically less than 100 MA (depending on whether you are powering other accessories like antenna relays, or auxiliary circuit boards from the hat's 5V power).

If you do use the external power input, remember that it may require filtering or transient protection for connection. For example: to connect it to an automobile, additional filtering and fusing will certainly be required. Also note that most "buck" regulators generator enormous amounts of RFI unless thoroughly filtered and shielded or unless they use carefully selected switching frequencies.

The heat sinking provided by the board is usually adequate for a 13 Volt power source, but you may need to add more heat sinking, external pre-regulation or a dropping resistor, especially if you intend to supply a lot of 5V power from the board for your own accessories.

Decide what you want to do and then plug the jumper block in across one pair of pins as shown on the silk screen: pins 7 and 9 to use 5 Volt power from the on-board regulator, or pins 9 and 10 to take 5 Volt power from the Pi.

Host computer selection

RadioHat 1.0 has been tested with:

- Raspberry Pi Zero 2
- Raspberry Pi 3B+
- Raspberry Pi 4B 2, 4 and 8 Gb

All worked well for basic transmit and receive operations with GnuRadio and pitrans. The Pi Zero 2 had some difficulty with memory shortages when attempts were made to add JackD and WSJT-X for digital modes.

Limited testing of receive capabilities has also been performed using slightly modified code from members of the discussion group at https://groups.io/g/keithsdr :

- Teensy® 4.0
- Teensy® 4.1

Other micro-controller boards with full speed I2S duplex support will probably work as well if adequate DSP library support is available.

Heat and case considerations

Some Pi cases do not have proper clearance for mounting "hats" as specified by the Raspberry pi foundation. Oversize heat sinks and/or fans are usually the culprits. A dual row square pin header can usually be used as an "extender" to get around this problem.

Metal cases that mount the hat outside the case generally work better because they add more shielding between the *RadioHat* and noise sources in the Pi. Some metal cases introduce difficulty with WiFi operation, but use of WiFi is usually not a good idea for this kind of application.

Even the most heavily shielded cases seem to provide adequate WiFi access for performing software updates and time syncing if they are brought within a few feet of a router.

PI GPIO

Here are some of the pins I'm using in my reference transceiver, but this is just one way of doing things - it's far cleaner to use I2C for everything except the two pins that enable the transmit and receive mixers:

```
//      GPIOs I'm not using (I've used up the SPI pins for GPIO)
//                4

//      These are still free but try to avoid them for future use:
//        5,6   Pi built-in shutdown detection
//    4       is used by Pimoroni power switch shim
//
//          UART - 14: TXD, 15: RXD, 16: RTS, 17: CTS
//

//   THESE ARE UNIQUE TO MY TRANSCEIVER  CABINET
```

```
//    The QRP labs filter boards requires the 10 meter filter to be always
//    enabled in addition to any other filters in use
#define GPIO_B1Filter          7 //      10 meters LP filter (Active High)
#define GPIO_B2Filter          8 //      15 meters
#define GPIO_B3Filter          9 //      20 meters
#define GPIO_B4Filter         10 //      40 meters
#define GPIO_B5Filter         11 //      80 meters
#define GPIO_TXRXRelay    12    //      High sends TX->Filters else Filters->RX
#define GPIO_PWRAMP_ON      13      //      High enables Power amp Bias

//    THESE ARE PART OF THE HAT
#define GPIO_notTX            22      //      a low enables the HAT tx mixer
#define GPIO_notRX            23      //      a low enables the HAT rx mixer

//    THESE ARE RESERVED FOR THE CONTROL PROGRAM'S USE
#define GPIO_KeyerDIT       24          //      keying and ptt inputs
#define GPIO_KeyerDAH  25
#define GPIO_SKeyIn           26
#define GPIO_PTTIn            27
```

I2C auxiliary control

The preferred way to to I2C networking is to connect additional 3.3Volt (only) I2C devices to the 4 Pin STEMMA style connector. The mating cable is a

JST PH 2mm 4-Pin Female Socket

One source of obtain information about the use this type of bus, connectors, assembled cables, and compatible modules is the Adafruit web site at https://www.adafruit.com/.

6. VNC Networking

RNDIS Gadget

This method of IP networking over USB works well with Apple devices including IOS devices connected with the Apple "Camera Connection Kit" USB dongle - it uses Zeroconfig and a self assigned IP address.

add to the bottom of the `config.txt` file:

```
dtoverlay=dwc2
```

Insert after `rootwait in cmdline.txt:`

```
modules-load=dwc2,g_ether
```

Similar configurations work with android devices that use rndis for tethering, but I haven't had a chance to document any of them - the setup is different for almost every Android device and operating system version.

Wired Ethernet

This works well enough for most applications, but it produces a little bit of digital noise. To network this way you will need to configure the host computer as required for adding an additional network. Many modern computers (like Apple Macintosh and IOS devices and possibly others) can easily create an "ad-hoc" networking connection if configured properly. If this is not possible static addressing and DNS service will need to be configured on both ends.

Wifi (not recommended)

This usually works as expected if you configure the pi to talk to your router, but the Pi's Wifi chip is EXTREMELY noisy and makes most HF operation difficult - especially if you are trying to back power the board from the Pi. The level of noise on the power supply lines varies greatly with distance to your router and frequency band in use.

With some operating systems you may need to install Bonjour, Avahi, or Zero-conf support to locate the Pi by it's hostname on your network. These services are often pre-installed, or easily installed for use with printers.

If none of them can be made to work you will need to configure the Pi using static addressing (preferably one configured by a DHCP reservation in your router), You will then have to manually type that static address into your client software.

7. Reference

J5 Power connector

These pins provide power supply input, output and option control for the Hat in various system configurations.

1, 2: 5.1 VDC 3 AMP external power INPUT to supply the Raspberry PI
3, 4: Dedicated Ground pins for use with 5.1VDc @ 3 amp external power
5: 5V BOARD power bus OUTPUT to run auxiliary devices like relays
6: 12V external power INPUT for the on board LM317 5V regulator
7: 5VDc OUTPUT from the LM317 regulator for connection with a bi-pin jumper to the RadioHat's BOARD power bus input pin (9)
8: Dedicated Ground pin for use with 12V external power input cabling
9: 5 Volt BOARD power bus INPUT.
10: Raspberry Pi's 5V bus power via a Schottky isolation diode

Use only the following jumpers to select a 5 Volt source for *RadioHat*

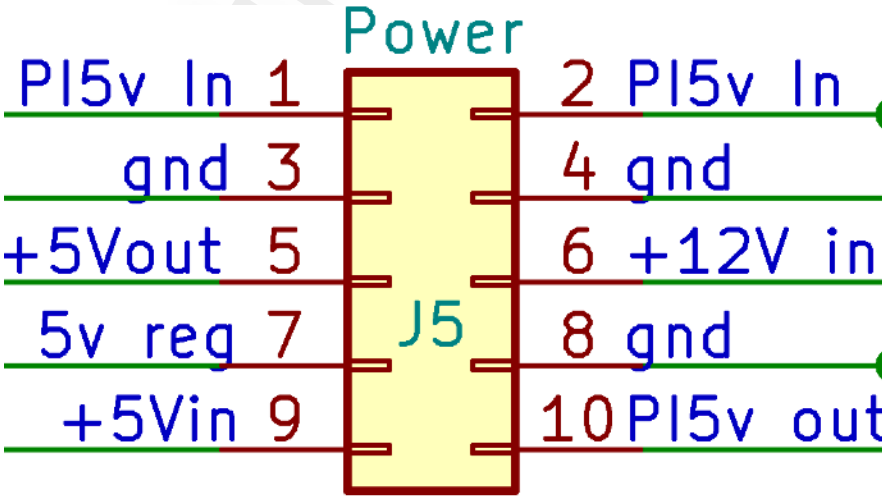
To power the *Radiohat* from an external source regulated by the LM317:

jumper pin 9 to 7

To power the *Radiohat* from the Raspberry PI's 5 volt power:

jumper pin 9 to 10

(Top View - Pin 1 is nearest the RF outputs)



J4 IQ Audio and Headset connector

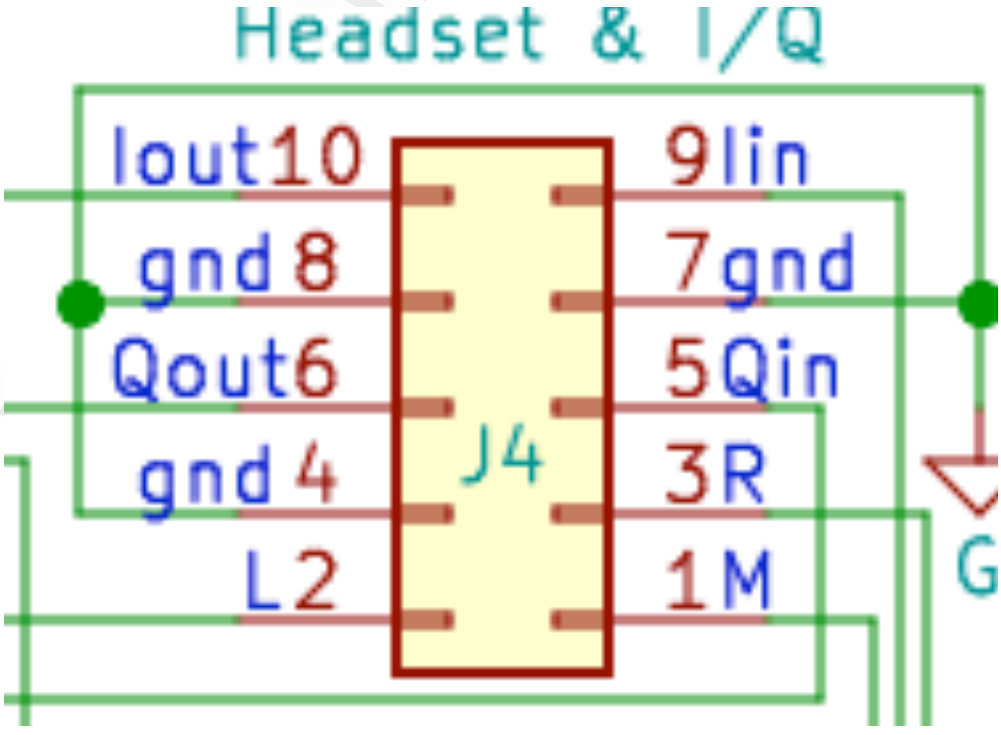
The Line Level IQ pins are arranged to allow separate 3 pin, single row, .1 inch female connectors with their center pins grounded to be used for input and output. The Headset pins are arranged for a 4 pin dual row .1 inch female connector.

It's usually a good idea to isolate any off-board audio IO connectors tied to this header from the chassis ground and sometimes their cabling will require shielding.

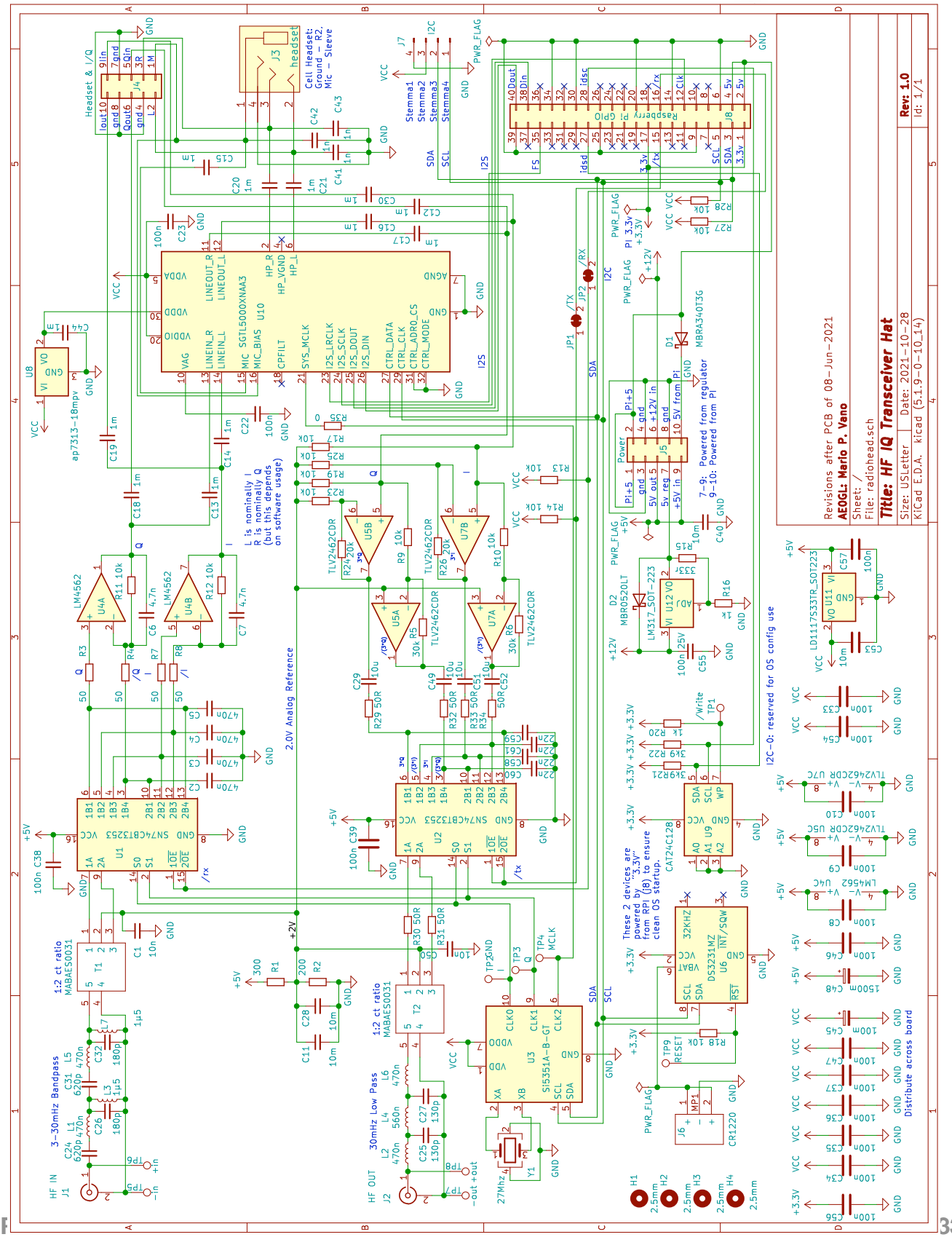
This is the pinout - as there are no real standards for I and Q you may find that need to swap them for compatibility with your software:

1: Headset microphone
2: Headset right channel
3: Headset left channel
4: Headset Connector Ground
5: Line Level "Q" (Quadrature) audio input to Modulator
6: Line Level "Q" (Quadrature) audio output from Demodulator
7: Modulator Input Connector Ground
8: Modulator Output Connector Ground
9: Line Level "I" (In-phase) audio input to Modulator
10: Line Level "I" (In-phase) audio output from Demodulator

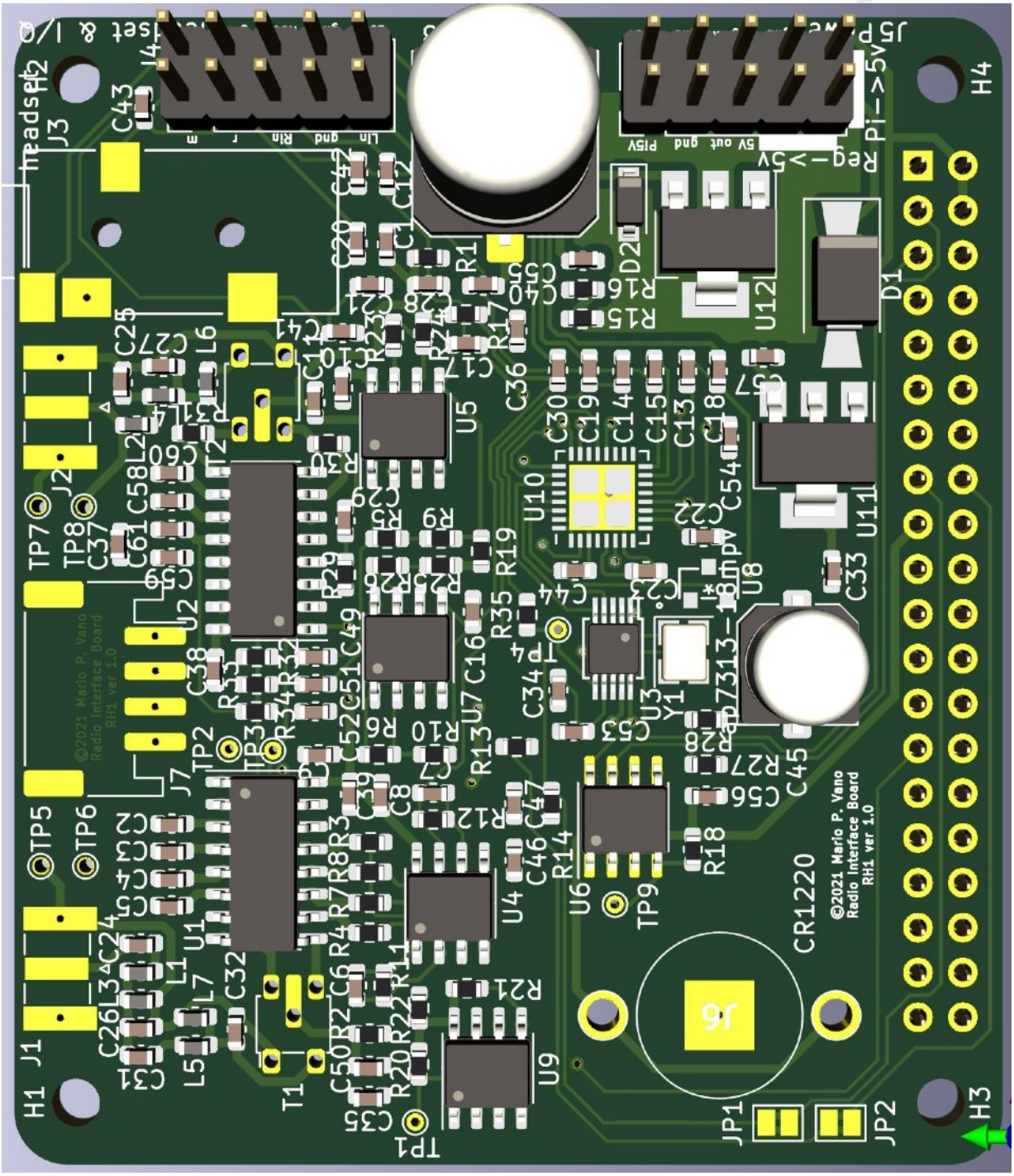
(Top View - Pin 1 is the pin nearest the RF outputs - this diagram needs to be redrawn rotated 180 degrees to make this less confusing)



8. Schematic diagram



9. PWB Rendering



10. FAQ

Q: Why are Low Pass Filters needed between *RadioHat* and a Power Amplifier?

A:

Quadrature Switched Modulators generate a great deal of harmonic output at multiples of their (suppressed) carrier frequency. In most cases the presence of these harmonics degrades the Intermodulation Distortion performance of any following amplifiers and can cause problems with legal compliance.

Another important consideration is that even if the following broadband amplifier was perfectly linear, the amplified harmonics at its output would still be strong enough to cause considerable heating and voltage breakdown problems in subsequent low pass filters.

Q: Why are Filters needed between an antenna and the *RadioHat* receiver input?

A:

Quadrature Switched Demodulators respond extremely strongly to signals at harmonics of the local oscillator frequency. Because this implementation uses a dual mixer, it is particularly responsive at odd multiples of the local oscillator frequency.

While the receiver can be operated for casual testing with no additional filtering, it generally requires some form of low pass or band pass filtering around the listening frequency for acceptable results.

Q: Why is it so hard to set up audio redirection for digital software under Linux?

A:

The audio system in Linux continues to be extremely unstable. Nearly every major release makes drastic changes to it. Keeping up with the changes is difficult if not impossible.

The current "Pulse" audio implementation tampers with audio routing and buffering behind the scenes in ways that are beyond the control of end users and programmers. Unfortunately, it is built in to the operating system in such a way that it is very hard to bypass. Hopefully the situation will stabilize someday.

11. Licensing (Modified MIT License)

This document, the example software, the Gerber files for PWB production and the Schematic and Bill of Materials are all being made available for whatever use you'd like to make of them. Here is the License for them:

The Kicad files and remaining documents will be made available at a later time, possibly under different licensing terms.

12. Links, Credits, references, etc.

Although these links are found elsewhere in the document, here is a collection of them in one spot. They are great resources and each of the individuals, organizations and companies are worthy of your support:

1. The Raspberry Pi Foundation home page: *https://www.raspberrypi.org*
2. Adafruit home page: *https: //www.adafruit.com*
3. PJRC Electronic Projects home page: *https://www.pjrc.com*
4. QRP Labs: *https://www.qrp-labs.com*
5. GnuRadio home page: *https: //www.gnuradio.org*
6. LiquidSDR (Liquid-DSP) home page: *https://liquidsdr.org*
7. Tayloe Mixer information: *http://www.norcalqrp.org/files/Tayloe_mixer_x3a.pdf*