ICARC Fox Transmitter

William Robison
KC0JFQ

September 29, 2020

A proposed article for publishing in QEX.

LaTeX
This is pdfTeX, Version 3.14159265-2.6-1.40.18

kc0jfq@n952.ooguy.com.

Documents webpage:
http://n952.ooguy.com/HamRDF

Contents

List of Figures

Supporting high resolution graphics files:

Schematics 102_73161_25.pdf

Figure 2, Schematic, Sheet 3
Figure 3, Schematic, Sheet 4
Figure 4, Schematic, Sheet 5
Figure 5, Schematic, Sheet 6
Figure 1, Schematic, Sheet 9

Boards 102_73161_25_pwa.pdf

Figure 6, Artwork Top, Page 1
Figure 7, Artwork Bottom, Page 4
Figure 17, Completed FOX Transmitter

Schematics 102_73176_0.pdf

Figure 8, Schematic, Sheet 2
Figure 9, Schematic, Sheet 3
Figure 10, Schematic, Sheet 4
Figure 11, Schematic, Sheet 5
Figure 12, Schematic, Sheet 6

Boards 102_73176_0_pwa.pdf

Figure 13, Artwork Top, Page 1
Figure 14, Artwork Bottom, Page 4
Figure 18, Completed FOX Transmitter

Schematics 102_73161_24.pdf

Figure 15, Schematic, Sheet 1

Boards 102_73161_24_pwa.pdf

Figure 16, Artwork Top, Page 1

William Robison KC0JFQ

1 Genesis of a New Fox Transmitter

The Iowa City Amateur Radio Club resumed foxhunting activities in early in 2018 following a long pause, a pause long enough for useful skills to have been lost to the mists of time. The location where we resumed our hunts is a county park near town. The park provided a lodge that was used as a comfortable base of operations. We started our hunts with a set of three transmitters that all operated at 146.565MHz. These are typical fox transmitters that are time multiplexed so that their transmissions do not occur concurrently.

Although these early hunts were comfortable for our event organizers, the novice fox hunters were having a rather difficult time locating the transmitters. With each transmitter being active for only one minute out of five, obtaining fixes was proving rather difficult. The success rate, that being the ability to find the transmitters, was disappointing.

To address this, I started a design effort to come up with a richly featured fox transmitter that would be able address the problems seen with our hunts. The fox transmitter architecture must be able to reproduce the operating modes used by other fox hunting products as well as satisfy the needs of novice hunters.

A casual set of requirements emerged somewhat as follows:

- Easily configured and synchronized schedules (a USB or WiFi connection to a host)
- Software selected frequency (no jumpers allowed for this!)
- Convenient mechanical packaging (easy to transport, setup and physically find)

Some wag even commented "Why doesn't it talk?". This, of course, was said in jest, but it turned out to be difficult to resist.

Two designs emerged, one using a single chip processor, a System-On-Chip, that draws minimal power. A second design replaces the SOC with a Raspberry PI-Zero that adds a PWM audio (i.e. voice) capability. The board outlines and the external mechanical interface of the two designs are identical.

1.1 A Tale of Two Transmitters

The SOC design provides control and programming access through an on-board USB UART. A simple verb/noun command structure allows the device to be tested and configured through the USB UART.

The Raspberry PI-Zero design can be accessed through the USB port provided by the PI-Zero. Substituting a Raspberry PI-Zero W provides network access through the on-board WiFi hardware. Operation of the PI-Zero hardware is predominantly using features present in the Linux system that runs on the PI-Zero.

1.2 Packaging

The first boards produced for the project were built to fit in a Hammond 1599E enclosure. This enclosure provides room for the circuit board and a battery and may be obtained in several mechanically compatible variations. Subsequent revisions were built on the same circuit board outline, moving the external connectors a bit to improve fit. The final artwork for both the SOC design (figure 17) and the PI-Zero design (figure 18) keep the external connectors in the same locations so holes in the housing match, allowing a single drilling jig to be used on the case.

2 Hardware

The two hardware variants are described in the following paragraphs. The first variant makes use of a ZiLOG zNEO SOC (System on Chip) device with a generous sized program memory (128KB). The second variant makes use of a Raspberry PI-Zero. The PI-Zero uses a 1GHz single core processor with 512MB of RAM. The PI-Zero is available without the 40pin header installed which allows the PI-Zero to be located above the main logic board with the top of the PI-Zero visible and easily accessible.

2.1 DC Power

Power conditioning is identical on both units. A switchmode regulator is used to improve battery life and allow freedom in the selection of the power source. A secondary linear regulator supplies 3.3V to the digital logic on the board.

Nominally power is supplied using a 6 cell AAA alkaline pack. Although a simple 9V radio battery would work, in practice six AAA cells seems to be a bit less expensive with the added benefit of longer run time. Primary cells are used to simplify battery management, simply replacing cells when necessary. A 3 or 4 cell Li-Po pack that will mechanically fit would also work well as the first stage regulator will tolerate input voltages up to 24V.

2.2 Frequency Control

Both units make use of clock synthesizers produced by Integrated Devices Technology. These integrated PLL (Phase Locked Loop) devices provide an integrated oscillator, PD (Phase Detector), CP (Charge Pump), VCO (Voltage Controlled Oscillator), and output scaler. Frequency selection is controlled by the oscillator crystal selected and the values loaded into several divider registers in the device.

The SOC variant uses an ICS525-02 clock synthesizer that is powered from either the 3.3V rail or the 5V rail. The output of the clock synthesizer is fed directly to an amplifier daughter-board

The PI-Zero variant uses an ICS307-02 that is powered from the 3.3V rail. The ICS307 output runs through a level shifter to supply a 5V logic level clock to the amplifier daughter-board.

2.3 RF Amplifier

The RF amplifier, when present, is located on a daughter-board to allow a bit of experimentation without having to alter the main board. The clock synthesizer raw output is, however, sufficient to produce about 20mW of power from 3.3V logic levels. Using 5V logic levels increases the output to about 30mW.

A simple class-D amplifier daughter-board has also been tested that uses a pair of CMOS inverters as the output amplifier that is able to produce about 60mW. This daughterboard is visible in figure 18)

2.4 FM Modulation

The transmitter is modulated by changing the load capacitance on the crystal on the ICS525/ICS307 using a pair of varactor diodes. Care must be exercised when selecting the crystal for the ICS525/ICS307. The load capacitance specification for the selected crystal must be high enough to allow for trim parts to allow the crystal to operate at its designed frequency. Installing a crystal with a poorly chosen load capacitance specification will have the crystal oscillating below its intended frequency when the modulation control is idle/disabled.

2.5 Audio

The SOC variant uses a timer channel to produce a square wave for generating the audio tone that modulates the carrier. This, of course, limits the SOC variant to transmitting code. The square wave is buffered and passed through a low-pass filter to reduce harmonics. It is then passed along to the modulation control circuit. The PI-Zero has two PWM channels (stereo), only one of which is used to generate audio. The PWM signal from the PI is buffered in the same manner as the SOC variant. The audio signal is filtered and passed to the modulation control circuit in the same manner as on the SOC board.

On the PI-Zero an audio amplifier is added to drive an on-board speaker or an external speaker. Producing sound from the PI-Zero typically requires an audio file.

Sending code on the PI-Zero variant is handled differently than on the SOC variant. Although a bit more involved, the PI-Zero approach provides some improvements. Code messages must be generated and stored in an audio file. I found a small utility written by Thomas Horsten called "cwwav" that I found on GitHub. A text file is fed into cwwav and a .wav files results that can be moved over to the PI-Zero and sent out through the audio device.

2.6 HT Control

Controlling the RF section of the fox transmitter is, for all practical purposes, identical to controlling a handheld transceiver. The audio and a separate Push-To-Talk control are routed to a header that provides the capability to control an external handheld transceiver. This connector also provides connections to the power subsystem to allow external power to be routed into the board.

2.7 TOY Clock

A Time-Of-Year clock is provided on both variants. This clock is a simple 32 bit counter with a backup battery. The SOC software keeps its system clock in much the same way that the Linux system maintains its system clock, a 32 bit count of seconds from some epoch.

This particular device, a Maxim/Dallas DS1672, also incorporates circuitry to charge the backup battery.

2.8 FRAM

The SOC variant stores all of its variable data, such as callsigns and operating schedules in an external non-volatile memory. The family of FRAM devices that are managed by the software are are made by Cypress Semiconductor. FRAM is non-volatile, of course, but has a speed advantage over EEPROM when writing as it operates at wire speed. The FRAM devices do not require any delays following a write operation.

2.9 Configuration Port, USB

This port appears only on the SOC variant as the Raspberry-PI model provides an alternate means of access. Physical access to the port is available only when the cover is removed (see figure 18). This protects the connector from debris when the fox transmitter is in use.

The USB device is powered through the USB port (rather than from the battery) so it does not draw unneeded power from the host system. Nominally the bit rate is set to 19,200 bits/second. Although this bit rate somewhat limits the download speed of the system, it is in line with the speeds that the CI-V port is capable of operating at. By operating the USB UART and the CI-V port at the same speed, commands may be injected through the CI-V port.

2.10 CI-V Port

The SOC device has 2 UART ports, both of which are used in the design. One, mentioned earlier, provides a host connection through the USB UART to program and test the system. A second serial port connects to a network port that emulates an ICOM CI-V port. The port is presented through a stereo 3.5mm jack that may be jumpered to operate half duplex or full duplex. The PI-Zero variant also implements this CI-V port in the same manner using its serial channel. The CI-V port is present to allow time synchronization between multiple devices in the field should the precision of the TOY clock prove inadequate.

2.11 B-mon

Both the SOC and the PI-Zero variants implement a battery voltage monitor. The SOC variant has an A/D as one of the peripherals in the device, one channel of which is used to sense the raw battery voltage. The PI-Zero variant, lacking on-board A/D capability, makes use of an external 4-channel A/D device (AD7991) that is accessed through the I2C bus. This A/D device is configured to use one of the four channels as a voltage reference, leaving 3 analog input channels available for use, one of which is used to sense the raw battery voltage.

2.12 I-mon

The PI-Zero variant adds a battery current monitor. A Zetex ZXCT1009 is located between the battery and the first regulator. To minimize loss, a 50 mOhm sense resistor is used to measure battery current. The ZXCT1009 converts the voltage drop across the sense resistor to a current that is placed across a 10K Ohm resistor and fed to the A/D. In normal operation less than 1mW is consumed in the battery current sense circuit.

3 Software

The software design aims to eliminate the need for field configuration. We really want to be able to simply turn the device on as it is placed in its hiding spot in the field.

Time synchronization is achieved using a battery backed TOY (Time of Year) clock with the operating configuration stored in non volatile memory. When powered on, the time is read from the TOY clock and the operating schedule is retrieved from non volatile memory.

The SOC fox transmitter can commence operation in less than a minute of being powered on. The PI-Zero fox transmitter, being a Linux based system, takes a few minutes to load.

3.1 Manual Control

The PI-Zero variant may be accessed in all the ways that a Raspberry-PI can be accessed. The SOC variant has a USB port that emulates a UART so that no special software or driver is required. Although the author makes use of a custom tailored terminal emulator that deals with the details of accessing the USB port at the correct speed, any simple terminal emulator should be sufficient.

In the case of the PI-Zero unit, with WiFi enabled you can use SSH to access the PI-Zero. The PI-Zero also has HDMI and USB ports to connect monitor , keyboard, and mouse.

3.2 Scheduling

The heart of the fox systems presented here is the scheduling architecture. This scheduling methodology is based on instruments built at the University of Iowa Department of Physics and Astronomy; the Cassini/RPWS instrument (launched in 1997 and orbited Saturn from 2004 through 2017) and the Juno/Waves instrument (launched in 2011 and currently orbiting Jupiter).

This scheduling architecture allows cyclic operation to be described using two parameters. One parameter is the cyclic period and the other is an offset into this period.

The basic idea of the scheduling method is to define a scheduling period that is synchronous with the system clock, and then call out an offset into that period for some activity to begin. Multiple activities can be coordinated with the only common knowledge in the system being the system time. The system time being expressed as seconds from an arbitrary epoch.

As an example, consider several units operating on a 5 minute cycle, where unit 0 starts at +0 minute offset, unit 1 starts at +1 minute, and unit 3 starts at +3 minutes. In this example, the 5 minute cycle starts at the top of the hour and repeats, eventually lining back up with the top of the next hour. Assuming these 3 units each spend less than one minute on the air, we want to see unit

1 start at 00:00:00, unit 2 at 00:01:00, and unit 3 at 00:03:00. The cycle will repeat at 00:05:00, 00:10:00 and so on.

This scheduling model provides exactly what we need to coordinate a group of fox transmitters that are to operate on a common frequency while preventing concurrent transmissions.

In practice this is accomplished by describing the schedule cycle in terms of seconds (which is nominally our scheduling granularity). Only two values are then needed to describe a schedule, the first being the starting point of the cycle, the period, and the second being the offset into the cycle. The system clock in all of the transmitters having been synchronized using their battery backed TOY clocks.

To find the start of a scheduling period, once each second the control loop calculates the current offset by dividing the system time by the scheduling period and comparing the remainder of the division with the scheduled offset. When the remainder and offset match, we are at the start of this scheduling point and the targeted activity can begin. This calculation occurs for multiple schedules that are being managed by the control software.

In practice, this allows the fox transmitter the ability to send several unique messages. Multiple unique messages may be transmitted before the cycle repeats.

4 A sample SOC setup

Operation of the SOC fox transmitter is entirely controlled by the commands stored in the FRAM. The FRAM is managed as a set of fixed length records, each one holding 32 bytes. Here is a sample setup.

The first group is the initialization commands. The operating software scans the FRAM looking for initialization records (those that begin with ”INI=”). These commands are executed in the order that they are encountered.

```
INI=TSET
INI=XTAL 24.576
INI=FREQ 144.335
INI=CALL W0JV/4
INI=CONF REV_12
INI=MODS S0 10:00 01:34
INI=MODS S1 10:00 03:30
```

4.1 TSET

The TSET command loads the system time from the TOY clock. This is how we get all the transmitters operating together.

The TOY clock chosen, the DS1672, stores time as an unsigned 32 bit number. The clock contents are used directly, without conversion, by the operating software.

4.2 XTAL

The XTAL command defines the ICS525 crystal. The ICS525 configuration tables allow five or six different crystal frequencies to be used. The parameter selects a section of the configuration table that corresponds with the selected crystal frequency.

4.3 FREQ

The FREQ command defines the transmitter operating frequency. The operating software will pick the nearest frequency at or above that supplied in the first parameter.

4.4 CALL

The CALL command defines our callsign. Noting the presence of the slash, we encode the callsign in much the same way we would for an APRS station to keep all the callsigns unique. This callsign will be sent at the beginning and end of every message.

4.5 CONF

The CONF command defines the hardware configuration. This command accommodates the 3 artwork revisions for the SOC designs. This example is taken from a setup for the revision 12 board. It selects the correct polarity for the various control lines.

4.6 MODS

The MODS commands define two command sequences to run. They are both cycle every 10 minutes with the S0 sequence starting one minute and 34 seconds into the period and the S1 sequence starting 3 minutes and 30 seconds into the period.

The software in the SOC scans the schedule table every second looking for the next sequence that needs to be executed. When a scheduling points for "S0" or "S1" are reached, the commands for that sequence are executed.

The SOC implementation is able to deal with up to 10 schedules. The PI-Zero is limited only by the available space on its micro-SD card.

We will now move on to look at an example command sequence that causes the fox transmitter to actually transmit a message. The "S1" in this example corresponds with the "S1" in the MODS command detailed earlier. These commands are issued whenever the S1 scheduling point occurs.

```
S1=TONE 1.4
S1=CWPM 20,-1,-1,-1,-1
S1=BEGN
S1=CWPM 25,1,9,15,15
S1=BATT 7.2
S1=CWPM 15,1,3,5,7
S1=WAIT 3
S1=CODE IOWA CITY
S1=CODE AMATEUR RADIO
S1=CODE CLUB FOX HUNT
S1=CWPM 20,-1,-1,-1,-1
S1=DONE
```

4.7 TONE

Starting with the TONE command, which sets the audio tone generator to 1.4KHz. The range of the audio generated here is limited by the filters between the driver and the modulator. In practice, the working range is about 300Hz to 2300Hz.

Take note that the audio frequency can be altered at any time.

4.8 CWPM

The CWPM command, which appears several times, sets the code generation parameters. The first parameter is the word rate. The following 4 parameters assign weights to the gaps in the generated code. The 15WPM setup has weights that should be familiar.

The -1 arguments select normal default values.

Take note that the code parameters (i.e. words/minute) can be changed at any time (as shown in the example).

4.9 BEGN

This command will enable the transmitter and send a CQ message with our callsign. This command enables the RF section and then sends a fixed format message that includes our callsign in order to comply with the rules.

4.10 DONE

At the end of the message, the DONE command sends a final message that contains our callsign, and then disables the transmitter. Our callsign, is sent at the end so that we remain in compliance with the rules.

4.11 BATT

The BATT command is a management command that reports on the current battery condition by sending a short message with the numerical battery voltage. The supplied argument is the low battery voltage point. When the measured battery voltage is above this trip point the battery message will contain "BATC HI HI HI HI". When the measured battery voltage is below the trip point the battery message will contain "BATC SOS BATC SOS ".

4.12 WAIT

The WAIT command pauses for the number of seconds specified in the argument. The carrier remains on and unmodulated during this pause.

4.13 CODE

And finally, message traffic is sent using the CODE command. The chipping rate (or WPM rate) is controlled, of course, by the CWPM command. The message text is case insensitive as we do not have a notion of case when sending code.

Bearing in mind the small record size in the FRAM, code traffic must be broken up into small enough chunks to fit within the 32 byte record limit. This doesn't noticeably affect the cadence with which the code is sent as the FRAM file system is reasonably fast at returning the next command.

5 PI-Zero setup

Much of the SOC operating code could be compiled on the PI-Zero, although the SOC software is a simple single threaded loop. There are limitations imposed by the small RAM (4KB) available on the SOC that make this a poor choice at this point in time.

Running on the PI-Zero gives us access to a much richer environment, In addition, many existing tools are available for the Raspberry-PI.

We can take advantage of cron jobs and "systemctl" services to have the PI-Zero automatically start running a bash script controlling the transmitter following power on.

A small number of utilities (in C) were written to deal with the unique hardware of the fox transmitter. They are detailed below:

5.1 AD7991

This routine reads the three analog channels of the AD7991 A/D device located on the I2C bus at address 0x28. The output may be tailored to drive the "vocalizer" routine mentioned later. This provides a vocal announcement of the status of the power system.

5.2 ICS307

This routine deals with controlling the RF and audio sections of the design. Various command line switches allow all of the bits in the ICS307 control word to be set or cleared. Although the ICS307 lives on the SPI bus, the handling of device select is non-standard and is handled in this utility. Other control lines, primarily power control pins, are also handled here.

5.3 Readsw

There are two jumpers present on the fox transmitter labeled MASTER and TEST. These jumpers are present on both board variants. This routine returns the status of these two jumpers. A third channel, in the form of a simple buffered switch closure that appears on the HT connector, may also be read using this utility.

5.4 vocalizer

This is a simple routine that is used to vocalize the battery voltage and current. There are sound clips (.wav files) that are required for each of the utterances supported by this utility.

5.5 Radio

This is where some useful support routines are located. Two, in particular, are of interest. First is the "halo_term" utility that I use to simplify access to USB ports.

The second is an implementation of the scheduling algorithm discussed in the scheduling section (**??**)), above. This program, when called with the same scheduling arguments as shown in the MODS discussion, will return control at the same time as the MODS schedule above starts.

We can schedule multiple activities by simply spawning of additional processes.

5.6 PIGPIO

The PI GPIO library is used to access the PI-Zero hardware. This library may be found at http://abyz.me.uk/rpi/pigpio/

The previously mentioned routines are written in C and make use of this library to deal with the details of the SPI and I2C blocks. PIGPIO also provides a daemon to overcome some issues and the above routines will use the daemon when it is loaded.

6 Performance

Battery life was of particular concern during design and prototyping of the SOC transmitter. Throughout the design several steps were taken to reduce the power requirements to allow the unit to operate for a reasonable time on battery.

Battery life for the PI-Zero variant is limited due to the use of the Raspberry-PI as the control element. The PI-Zero, when idle, requires about 1/2 watt. Power consumption when active rises to about 3/4 watt. Although this is almost 20 times the SOC power draw, a set of 6 AAA batteries should provide four to eight hours of operation.

Figure 1 is the power measurements taken from the first few units produced. The FOX1 through FOX4 units are built from early revisions of the artwork that lack the switchmode supply.

The current SOC revision, referred to as the ”-25” revision in the figure, has several power saving updates from earlier versions. In particular the the ”-25” revision changed to a switch-mode regulator to reduce battery voltage from nominally 9V to 5V for use by the amplifier and the ICS525. The discrete oscillators X1 in Figure 4 and X2 in Figure 2 were not used, being replaced with crystals to reduce cost and power consumption.

Operational cost is further reduced by using a six cell ”AAA” pack (or an external pack) rather than a 6LR61 battery. Cost of six ”AAA” is usually less than a single 6LR61. Operational life is also extended considerably.

The Estimated Run Time columns break down the best case times for the 6LR61 having a capacity of 550mAH and ”AAA” cells that range from 850mAH to 1200mAH capacity. Estimated run times, even for the 6LR61 battery, should be sufficient to handle most hunts.

The ”AAA” six cell pack fits in the target case while a ”AA” pack is too large to allow the cover to be installed. If extended times are needed, an external pack or lithium chemistry batteries may be employed to extend run time. The primary voltage regulator will tolerate input voltages up to 24V to allow a multi-cell pack to be employed.

7 Conclusion

The first hunt where these new units were deployed resulted in a dramatic increase in the number of transmitters that were actually located by the hunters. Managing the transmitters by the event organizers is rather trivial, simply turn the unit on as it is dropped in its hiding place and move on. The brightly colored antenna provides an easy to find visual for the hunter as well as for the event organizer at the end of the hunt. No synchronization procedure is required on the day of the hunt.

For the hunters, we move away from punch cards to record a find and simply have them record a random ID number that is taped to each unit. This method, of course, requires a bit of effort to update the ID number on each unit prior to the hunt but eliminates the need to station additional hardware at the fox transmitter.

7.1 Authors email

kc0jfq@n952.ooguy.com

7.2 Board Availability

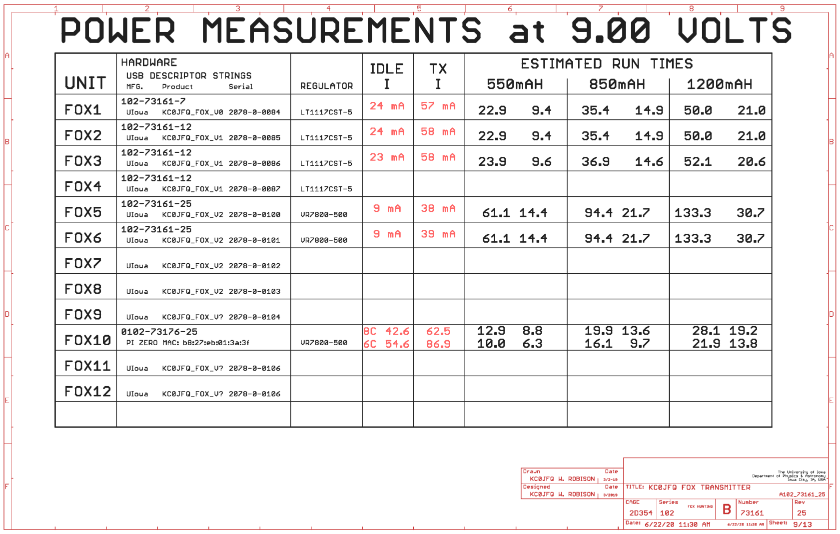
Transmitter boards may be obtained from the author for $10 each.

7.3 Additional Documents

The authors web site for the Fox Transmitters:

http://n952.ooguy.com/HamRDF

The board is predominantly surface mount. Assembly requires some skill with an iron and a static safe work area. Build documents are available on the web site.



POWER MEASUREMENTS at 9.00 VOLTS

UNIT	HARDWARE USB DESCRIPTOR STRINGS MFG. Product Serial	REGULATOR	IDLE I	TX I	ESTIMATED RUN TIMES 550mAH	850mAH	1200mAH
FOX1	102-73161-7 UIowa KC0JFQ_FOX_V0 2078-0-0084	LT1117CST-5	24 mA	57 mA	22.9 9.4	35.4 14.9	50.0 21.0
FOX2	102-73161-12 UIowa KC0JFQ_FOX_V1 2078-0-0085	LT1117CST-5	24 mA	58 mA	22.9 9.4	35.4 14.9	50.0 21.0
FOX3	102-73161-12 UIowa KC0JFQ_FOX_V1 2078-0-0086	LT1117CST-5	23 mA	58 mA	23.9 9.6	36.9 14.6	52.1 20.6
FOX4	102-73161-12 UIowa KC0JFQ_FOX_V1 2078-0-0087	LT1117CST-5					
FOX5	102-73161-25 UIowa KC0JFQ_FOX_V2 2078-0-0100	VR7800-500	9 mA	38 mA	61.1 14.4	94.4 21.7	133.3 30.7
FOX6	102-73161-25 UIowa KC0JFQ_FOX_V2 2078-0-0101	VR7800-500	9 mA	39 mA	61.1 14.4	94.4 21.7	133.3 30.7
FOX7	UIowa KC0JFQ_FOX_V2 2078-0-0102						
FOX8	UIowa KC0JFQ_FOX_V2 2078-0-0103						
FOX9	UIowa KC0JFQ_FOX_V? 2078-0-0104						
FOX10	0102-73176-25 PI ZERO MAC: b8:27:eb:01:3a:3f	VR7800-500	8C 42.6 6C 54.6	62.5 86.9	12.9 8.8 10.0 6.3	19.9 13.6 16.1 9.7	28.1 19.2 21.9 13.8
FOX11	UIowa KC0JFQ_FOX_V? 2078-0-0106						
FOX12	UIowa KC0JFQ_FOX_V? 2078-0-0106						

Drawn KC0JFQ W. ROBISON Date 3/2-19
Designed KC0JFQ W. ROBISON Date 3/2019
TITLE: KC0JFQ FOX TRANSMITTER
CAGE 2D354 Series 102 FOX HUNTING B Number 73161 Rev 25
Date: 6/22/20 11:30 AM Sheet: 9/13

Figure 1: 102-73161 Power Measurements

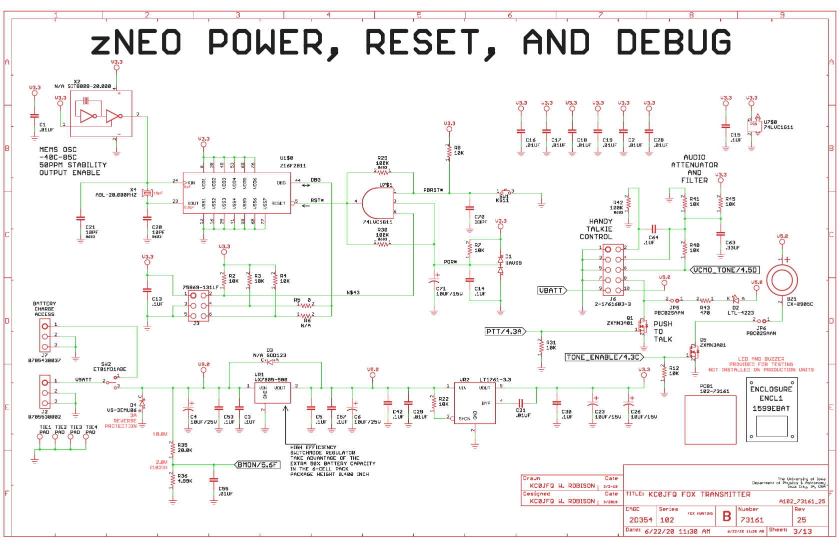


Figure 2: 102-73161-25 Regulators, Reset, HT connector

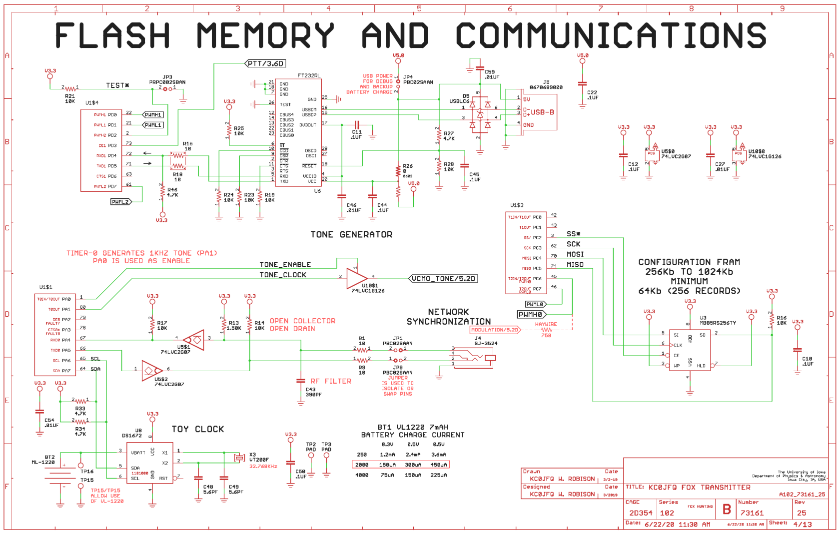


Figure 3: 102-73161-25 Peripherals

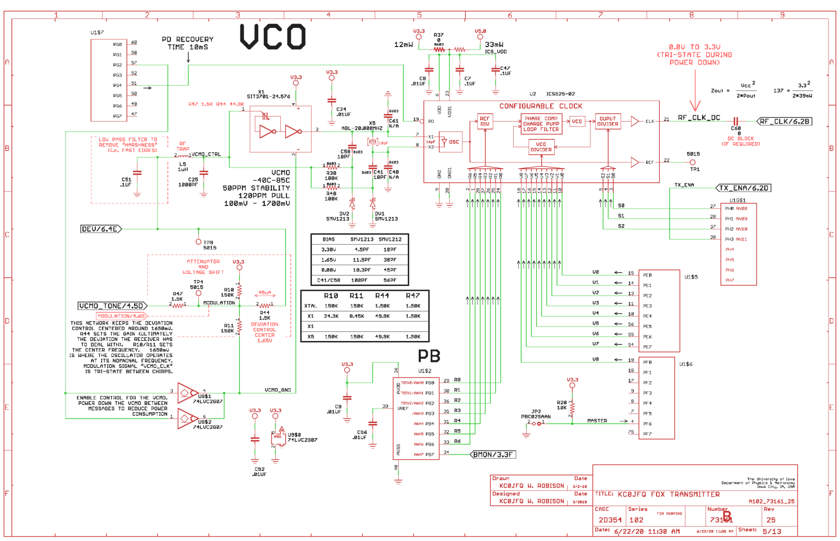


Figure 4: 102-73161-25 Modulation and VCO

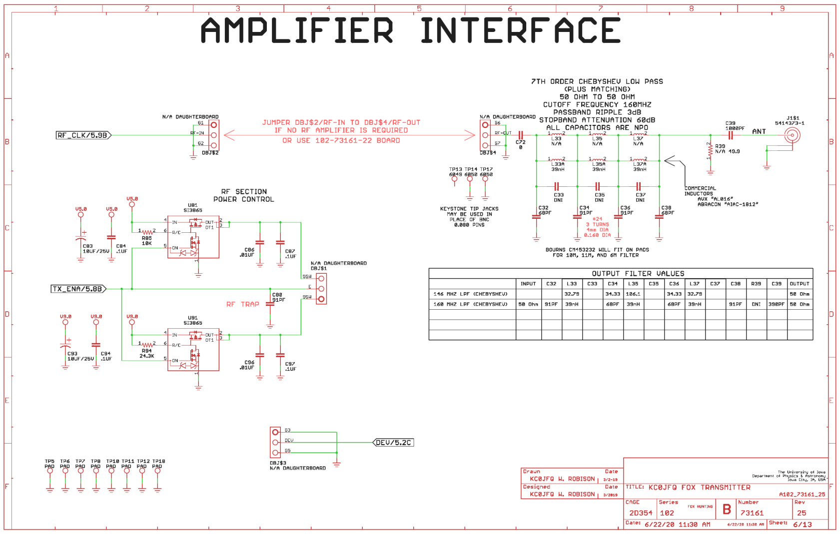


Figure 5: 102-73161-25 Amplifier Interface and Output Filter

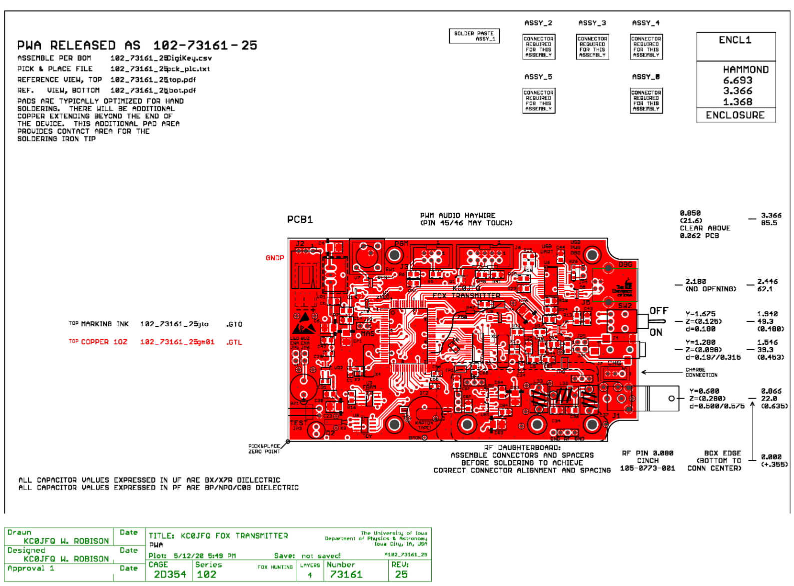


Figure 6: 102-73161-25 Artwork, TOP

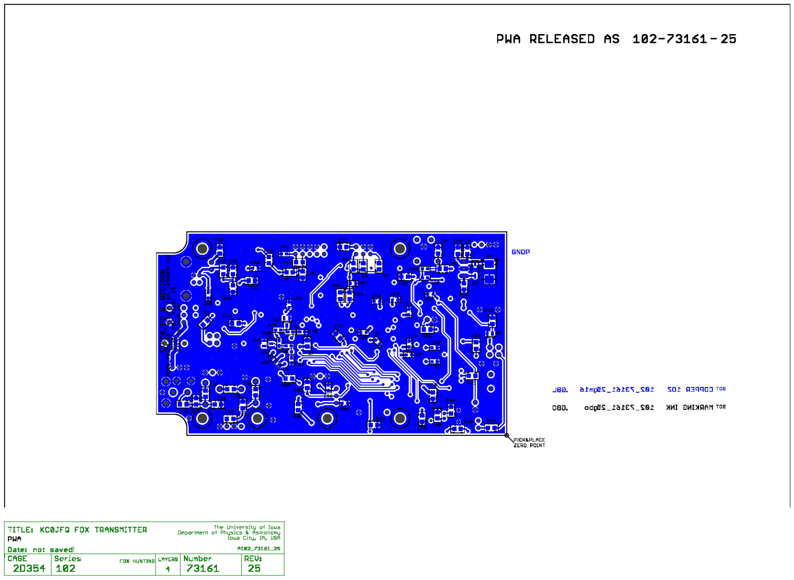


Figure 7: 102-73161-25 Artwork, BOT

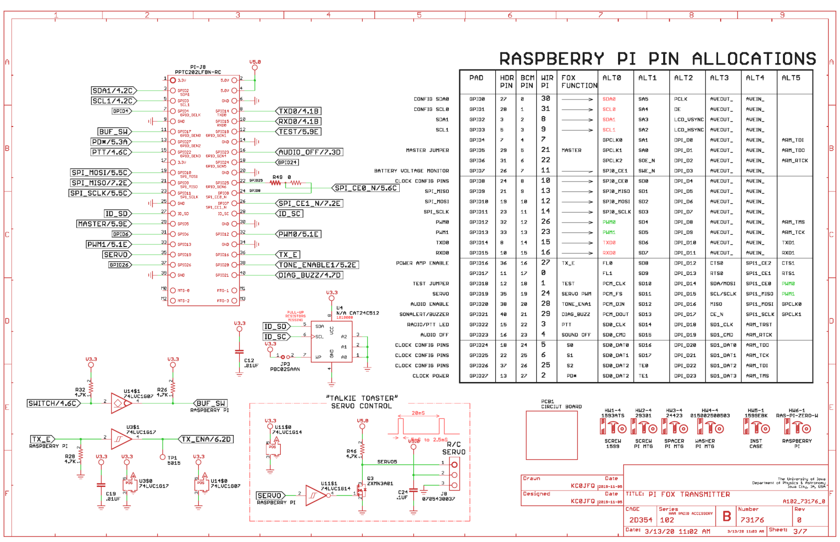


Figure 8: 102-73176-0 PI-Zero connector

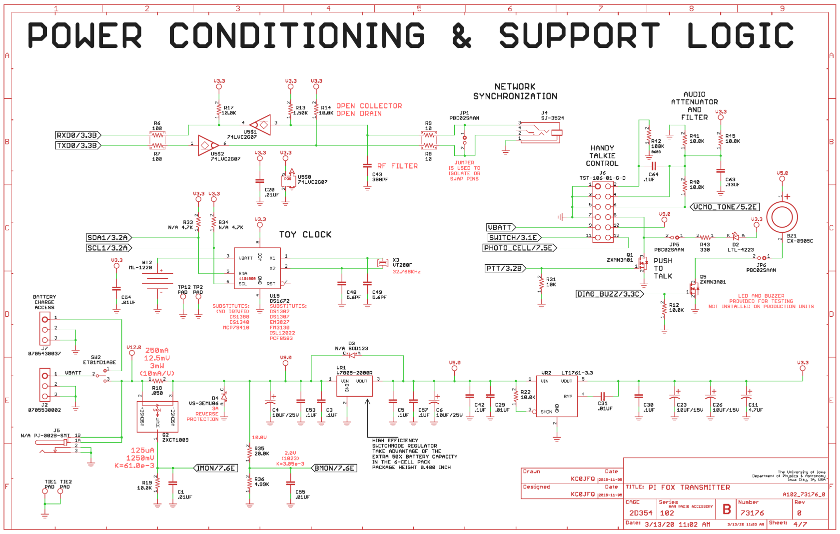


Figure 9: 102-73176-0 Regulators, TOY, CI-V

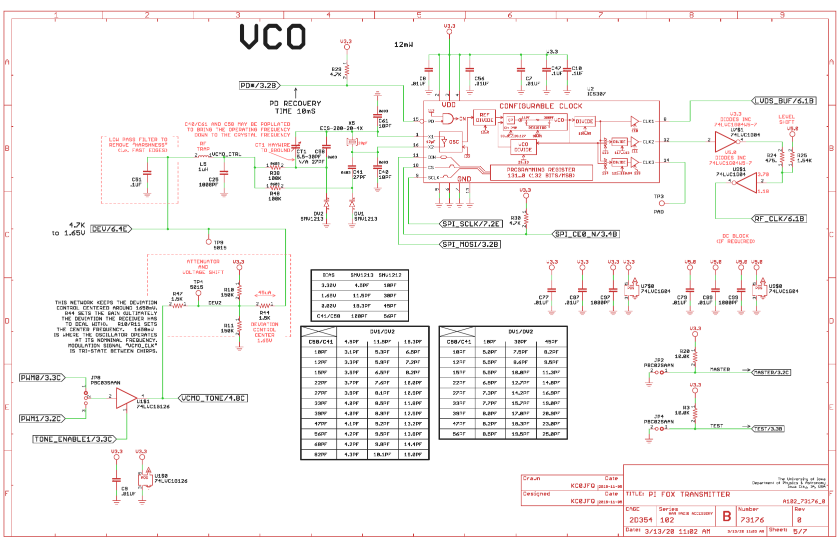


Figure 10: 102-73176-0 Audio filter, VCO

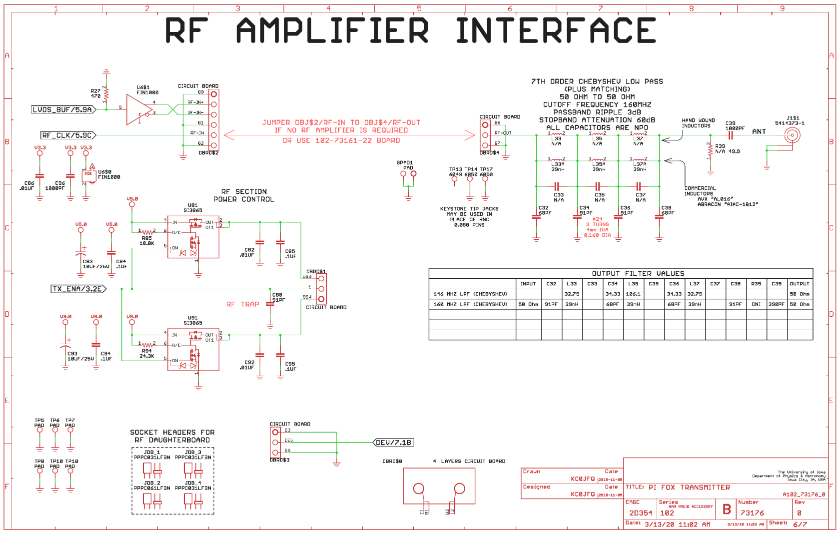


Figure 11: 102-73176-0 Amplifier Interface

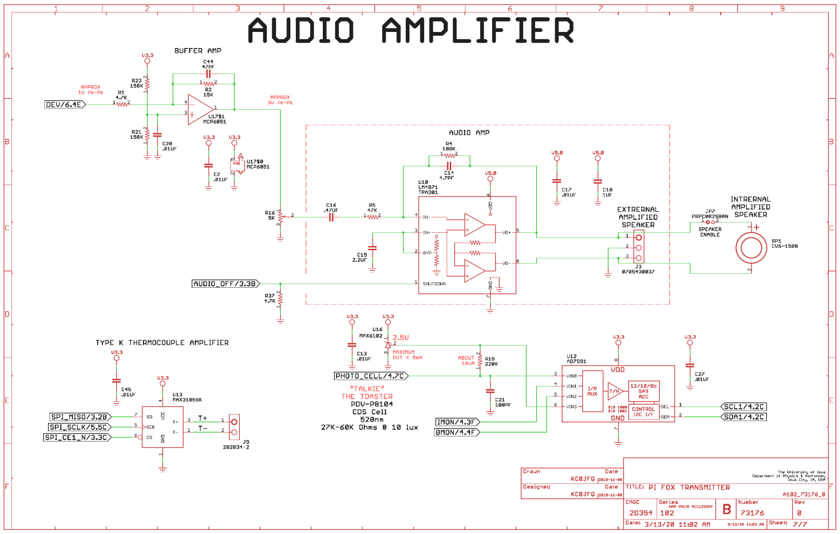


Figure 12: 102-73176-0 Audio Amplifier

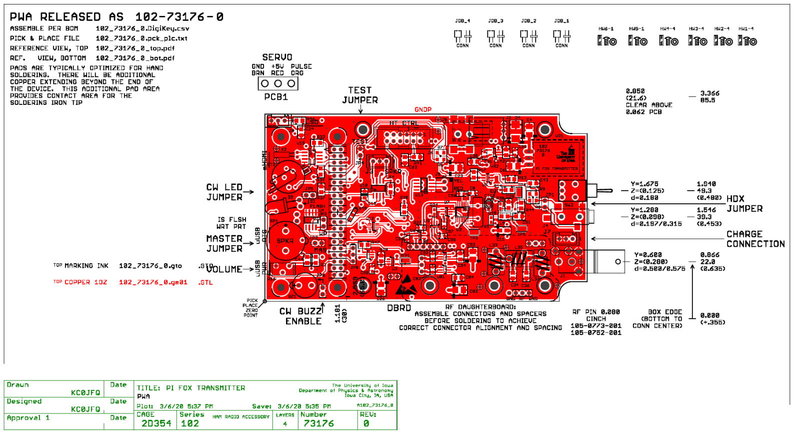


Figure 13: 102-73176-0 Artwork, TOP

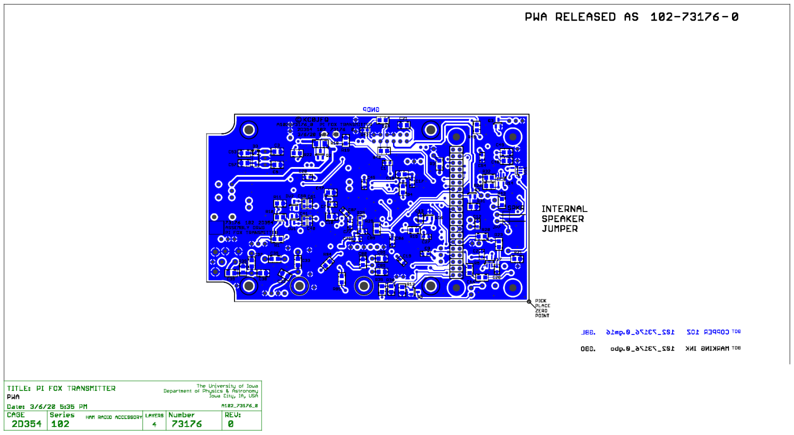


Figure 14: 102-73176-0 Artwork, BOT

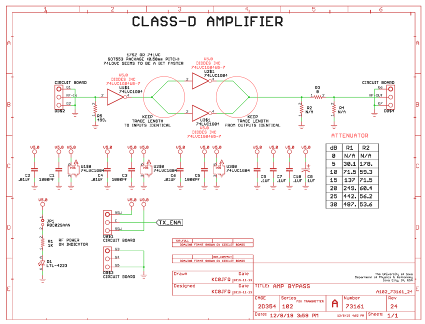


Figure 15: 102-73161-24 Class-D Amplifier

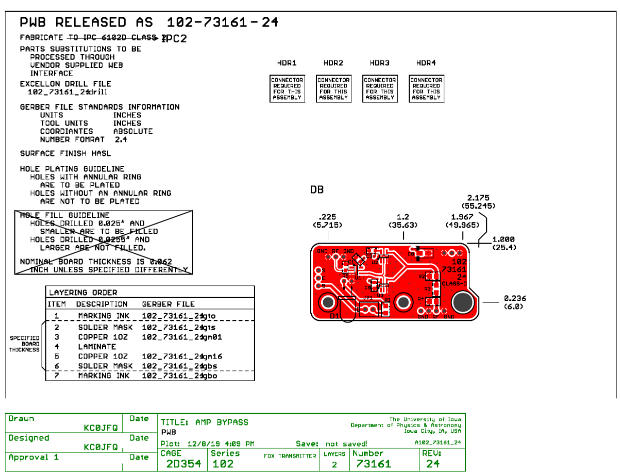


Figure 16: 102-73161-24 Artwork

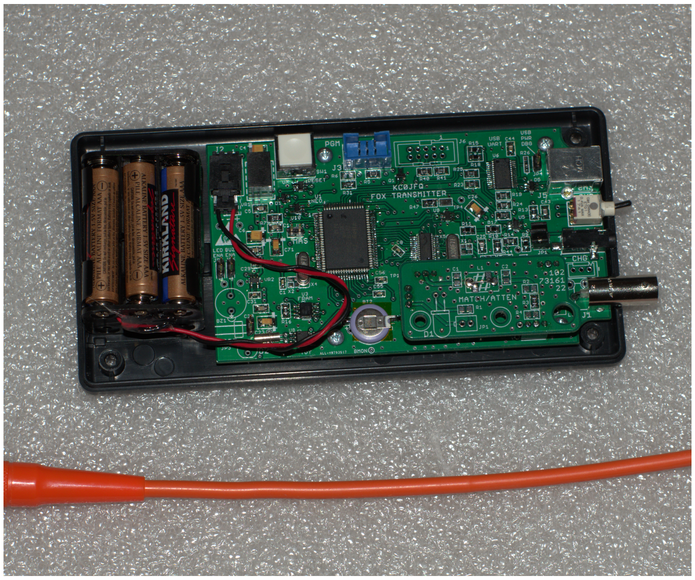


Figure 17: 102-73161-25 Board Image

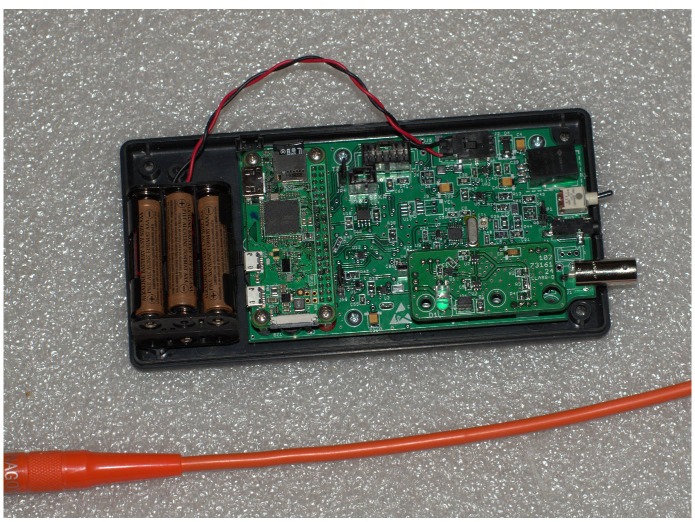


Figure 18: 102-73176-0 Board Image