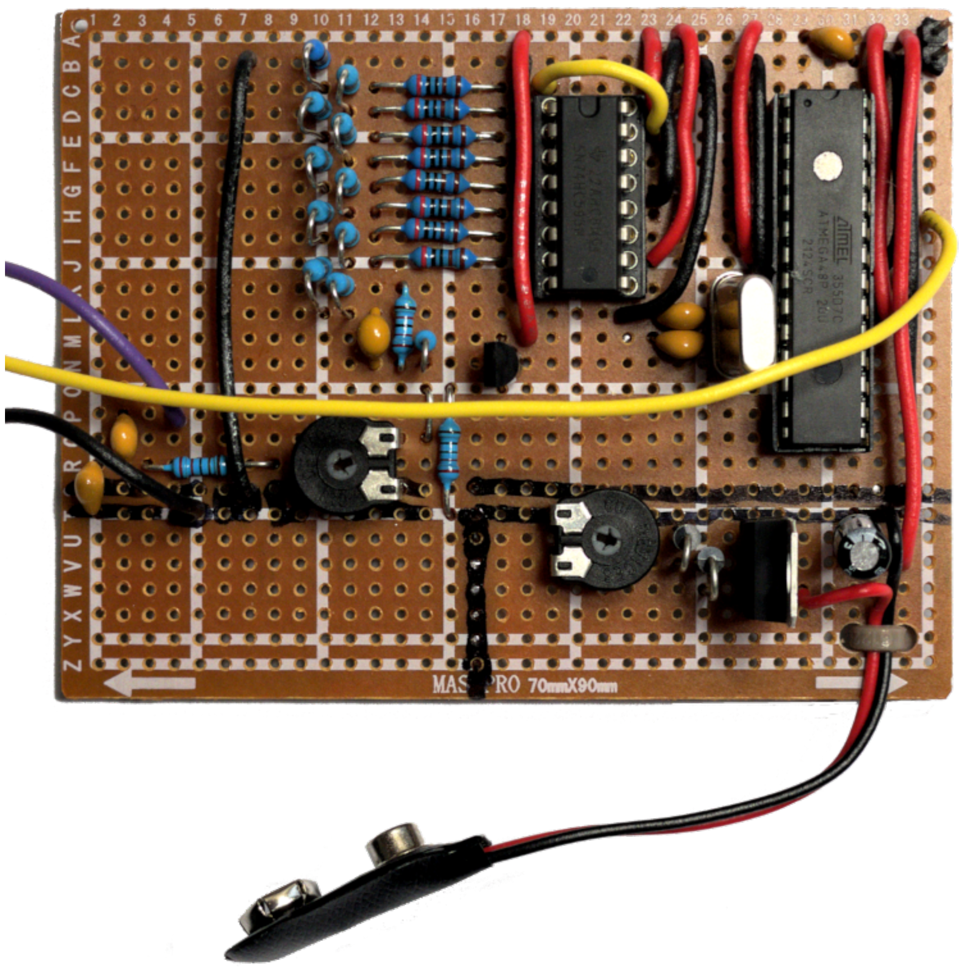
The `jeffpc` Amateur Radio Fox



Revision A

Please address any comments or questions to Josef 'Jeff' Sipek <jeffpc@josefsipek.net>.

Built from SCM commit `560:3c325d1402c9` from 2022-12-27 21:14Z

Contents

Part I

Design

Chapter 1

Goals

There is already a number of different fox hunting designs out there—both commercial and hobbyist built. Therefore, there is no *practical* reason to make another design. The design described in this document came about because of a combination of three factors: (1) desire to experiment with microcontrollers and electronics in general, (2) desire to deploy a fox transmitter for others to find, and (3) the inspiration I got from Eliot Mayer's W1MJ 5-Watt Fox V1.0 writeup.

This chapter documents the various goals.

While the ultimate goal was to make a low-power fox, I decided that a good "version 1.0" milestone would be to focus on the audio portion of the circuit and use an existing radio for the RF portion. I hope to revisit this simplification in future revisions.

Physical size. The fox should be reasonably small to allow for discreet deployment in a wide variety of locations. A box no bigger than 20 cm by 12 cm by 10 cm should be sufficient to house all the electronics.

Power consumption. The fox should use a reasonably small amount of power to not require high capacity or high current batteries. A set of AA batteries should last at least a day and ideally would last at least 36+ hours to cover a weekend-long hunt.

Modes of operation. Many low-power foxes produce a programmable tone pattern and a Morse code identification to comply with the regulations. There is no technical reason the fox firmware could not generate complex tone patterns based on mathematical expressions. This apporach can create very complex patterns with a very small footprint in the firmware or configuration. The firmware should include all the possible modes of operation (e.g., Morse code ident only and various tone patterns) and the exact mode used should be configurable.

Extensible. The hardware portion of the design should be generic enough that major behavior changes require only firmware updates.

Audio output. The generated audio should be high enough fidelity to be able to convey speech and data in addition to Morse code. Being able to produce signals up to 8 kHz requires generating samples at 16 kHz rate. An 8-bit sample size should provide enough resolution.

Push-To-Talk. The fox should generate a signal that can be used to key a radio.

Avoid exotic components. The fox should use use commonly available components. Specialized components can simplify the design, however they increase the cost, are harder to source[1], and make the design more likely to become obsolete sooner.

[1] Especially during the semiconductor supply chain issues in 2022.

Chapter 2

Electrical

This chapter describes the hardware design and presents the justifications and calculations used for component selection. Throughout this chapter portions of the schematic will be shown. The full schematic can be found in appendix B. The operating limitations are summarized in appendix A.

2.1 Theory of Operation

Please refer to the schematic in appendix B.

The ATmega48P microcontroller U1 runs at approximately 8 MHz and generates 8-bit samples 16384 times per second. Each sample is output on the D and D_CLK pins used as SPI data and clock pins, respectively. Some time after the sample is shifted out, the D_LATCH pin goes high indicating that the most recent value should be transmitted. This latch signal delays the “liveness” of the data to make it appear at the transmitter at the right cadence—about every 61 microseconds (1/16384 seconds).

The data and the latch signal are fed into the 74HC595 shift register U2. After the 8 bits of the sample are shifted in, the latch signal’s positive edge latches the shifted out value to the shift register’s parallel output.

The 8-bit value on the shift register’s output is fed into an R2R DAC (R1–16), then attenuated by the R17 and R18 voltage divider, buffered by the Q1 emitter-follower, attenuated further by the RV2 trim-pot, filtered by the R22 and C6 low-pass filter, and finally AC coupled to the radio via C7.

U3, an LM317, provides a stable 3.3V voltage for the rest of the circuit.

When using the Baofeng interface circuit, the PTT signal output by the microcontroller is fed into the Q2 MOSFET's gate to ground the PTT pin on the radio's 3.5 mm connector. The ground, PTT, and audio connections to the radio use toroids L1–3 to suppress any RF coming back from the radio.

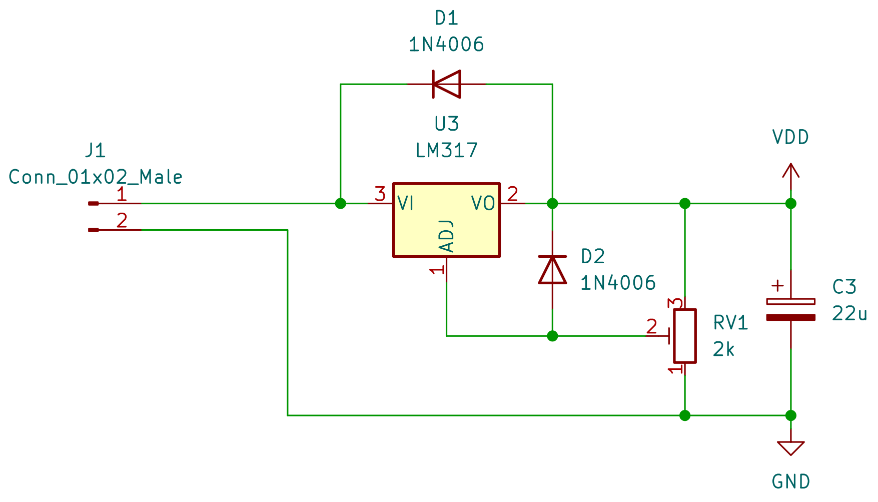
2.2 Power Supply

The entire circuit runs on 3.3V. This is to allow future expansion (e.g., adding 3.3V SPI flash with PCM data) and to maximize the range of usable battery voltages.

At a relatively light load (e.g., 20mA), the LM317 has a dropout voltage of about 1.5V. Therefore, we should be supplying it with at least 4.8V (3.3V+1.5V), but 5.5V is a safer choice to allow for variation in manufacturing and operating conditions. The choice of 5.5V is plenty for the expected power source—6 AA NiMH rechargeable batteries.

Each NiMH cell can reach about 1.4V when fully charged and goes down to about 1V when just about fully discharged. Therefore, a pack of 6 cells in series produces voltage between 8.4V and 6V, which is well above the selected 5.5V minimum.

The LM317 has a minimum load current requirement to maintain regulation ($I_{O(min)}$) which can be as much as 10mA. This design uses 15-20mA and therefore trivially meets this requirement.



The power supply circuit is a simplified version of the circuit found in the LM317 datasheet. Notably, because the load is relatively constant, it removes most of the filtering capacitors. The datasheet recommends 1N4001 diodes to protect against input and output short circuits. I had 1N4006 on hand and so I used those instead. The lower rated 1N4001 should be sufficient.

2.3 Microcontroller

Somewhat arbitrarily I've selected the ATmega48 as the microcontroller. It is relatively cheap, much less popular than the ATmega328 powering Arduino-based designs and therefore more available even during the semiconductor supply chain issues in 2022.

The chip can easily operate at 3.3V and draws only a few milliamps when active. There is a handful of different variants of the chip (e.g., A, P, and PA suffixes). Any version capable of operating at 3.3V at the needed 8.4MHz should work. I've used ATmega48P-20PU and will use ATmega48A-PU as well.

The USART is used in SPI mode to shift out the sample data. Future revisions of this design will possibly make use of the SPI and I2C hardware as well to provide additional features.

2.4 Timing

There are two major timing requirements:

1. The microcontroller needs to run fast enough to have enough time to calculate and output the next sample.
2. The microcontroller needs to output the samples at as close to the correct sample rate as possible.

To be able to produce audio up to 8kHz, we must output samples at at least 16kHz. The sample computation duration depends heavily on the kind of audio signal that's being produced (e.g., Morse code takes less computation than a complex digital waveform) but can easily exceed 150 cycles[1].

Therefore, the microcontroller should run at at least 2.4MHz ($150 \cdot 16k$).

[1] AVR microcontrollers execute most instructions in 1 cycle.

The ATmega48 microcontroller has timers that can fire an interrupt at a pre-set rate[2]. The timers work using a scaled version of the system clock and all of the available scaling factors are powers of two.

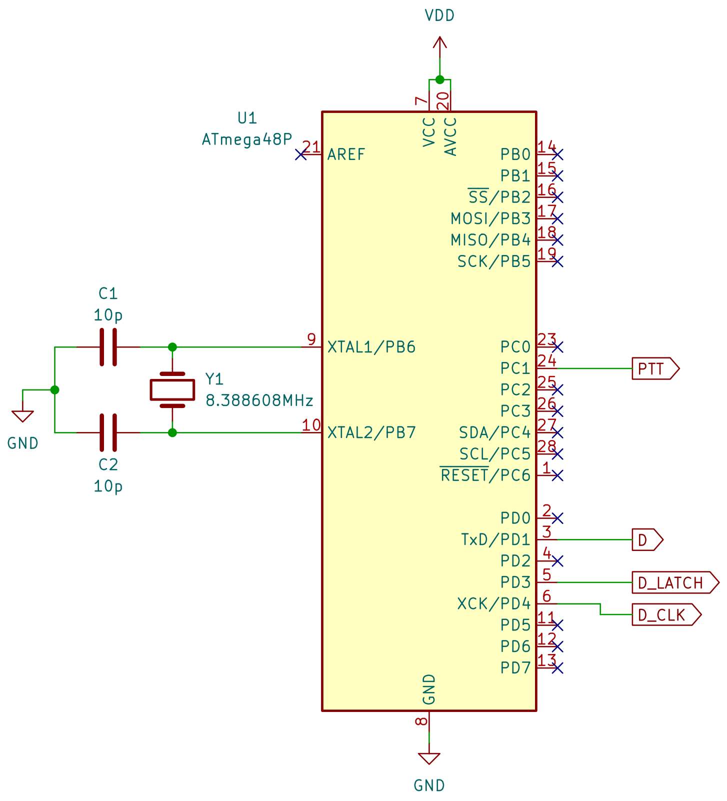
Since the timer prescalar uses factors of powers of two and the actual sample calculation is *significantly* simpler if the sample rate is also a power of two (i.e., multiplication and division by the sample rate become simple left and right bit shifts), it makes sense to make everything a power of two—including the system clock. Additionally, because the scaling is exact, a major source of jitter in the interrupt inter-arrival time is eliminated.

The closest power of two to 16kHz is 16384Hz.

The available powers of two greater than 2.4MHz (see above) and less than 20MHz (the maximum frequency from the ATmega48 datasheets) are 4, 8, and 16 (binary) MHz.

To have ample room for future expansion, I went with 8MHz crystal—or more accurately 8.388608MHz.

This (2^{23}Hz) divides nicely by the 16384Hz sample rate, providing 512 cycles per sample to compute and output the sample via SPI.

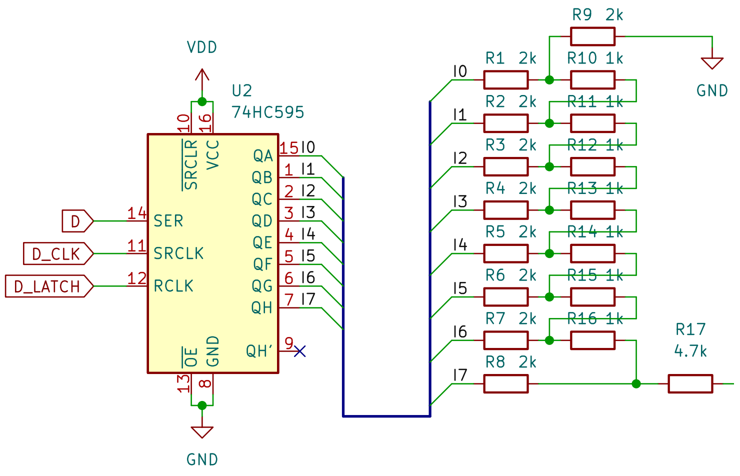


[2]More accurately, they fire at a present counter value and the counter value, which is a period rather than frequency.

The two 10pF capacitors are intended to make the crystal operate at its designed frequency. The crystal datasheet indicates a 30pF load capacitance, while the ATmega48P datasheet recommends 12-22pF for each of the capactors. Somewhat accidentally, I tried using 10pF and kept them since they worked.

2.5 DAC

The DAC is made of a 74HC595 shift register and a R2R circuit that generates one of 256 equally spaced voltage levels between 0 and 3.3V.



Recall that the microcontroller runs at approximately 8MHz and produces 16384 samples per second. This gives it 512 clock cycles per sample.

In the fastest configuration, the SPI hardware in the microprocessor can output data at half the clock speed, so if there were zero processing and it just shifted out samples as fast as it could, it would be limited to 32 bytes/sample.

Some time is needed to generate the sample, so if I allot 75% to computation and 25% to output, I can only do 8 bytes/period. A typical SPI DAC IC uses something like 2-3 bytes to output a sample. This is uncomfortably close to not having enough time to calculate the sample.

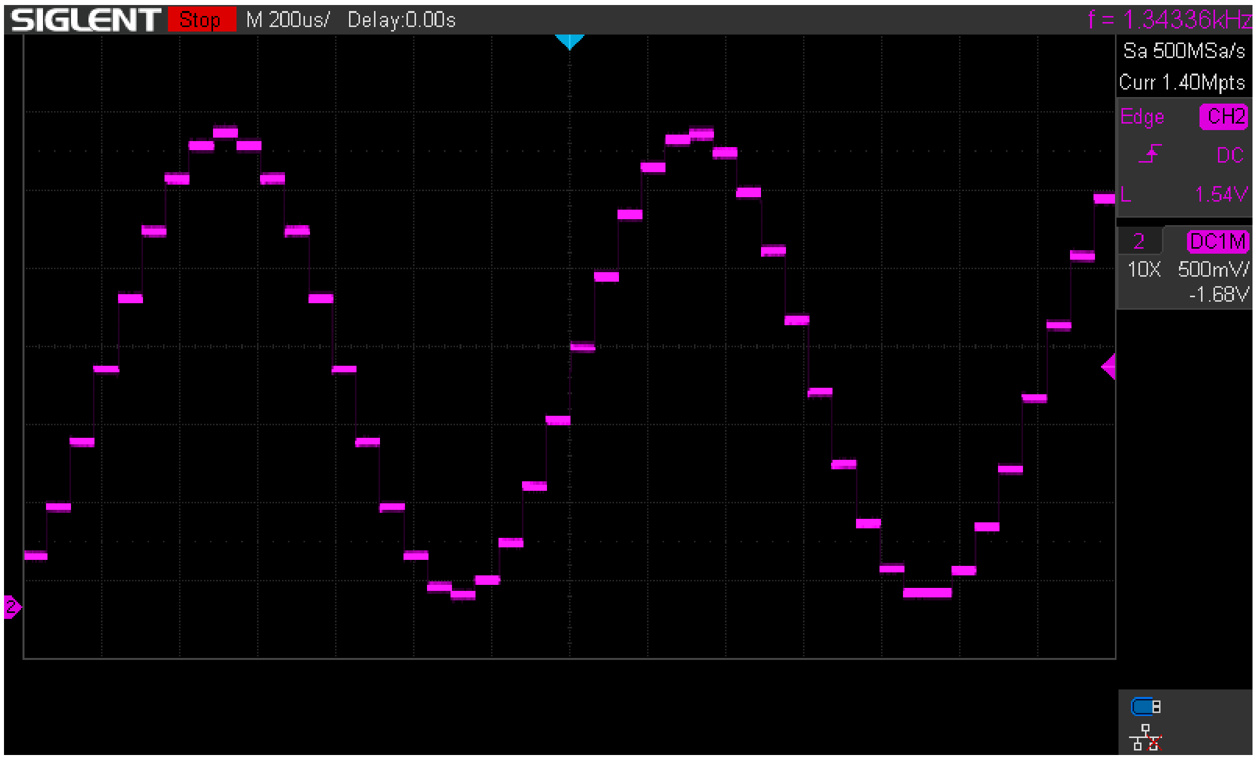
Running the chip at twice the frequency (i.e., approx. 16MHz) would require increasing the supply voltage to at least 3.8V. This would make future integration with 3.3V devices more difficult, consume more energy, and get closer to the edge of the envelope for the microcontroller.

Another option is I2S. There are ICs that essentially forgo the command and just output the values as they show up. This is good, but they are more specialized, cost a couple of dollars each, and many of the common ones are currently (2022) back-ordered.

Sufficient performance can be achieved with a shift register and a R2R network of resistors, which are common, dirt cheap, and available. As a bonus, the design is actually more extensible as well. That is, it is possible to wire up a second shift register in series and extend the output from 8 bits to up to 16 bits trivially[3].

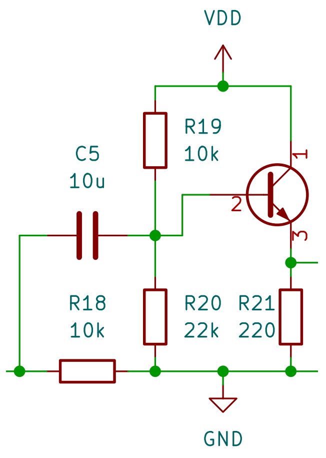
The only downside to using a shift register with a R2R network is that the output can't drive anything without a buffer on its output.

Below is an example of the DAC's output when generating a sine wave. It is apparent that the limiting factor in the output's smoothness is the sample rate and not the DAC resolution.



[3]Component tolerances will likely make it far from trivial to get the 15.3μV steps required for a 16-bit DAC, and therefore overall it is unlikely to be a trivial design. However, 10-bit DAC should be quite doable.

2.6 Amplifier



The output of the DAC is fed into a emitter-follower amplifier acting as a buffer. A discrete transistor is used instead of an opamp because commonly available opamps do not have the necessary range at 3.3V or do not operate at all at such a low supply voltage. Specialty rail-to-rail opamps exist, but they are harder to source and command a higher price.

For example, both the TL074 and the LM324 fail to meet our requirements. The TL074 recommends a $\pm$5V supply which is far greater than the 3.3V we can provide. The LM324 works at 3.3V but has severe limitations on the output range. It is guaranteed to output up to 1.6V below V_{CC} and 1V above ground. In other words, the range is between 1V and 1.7V.

2.6.1 Emitter-follower biasing

$$I_{CQ} = 7.5mA \tag{2.1}$$

$$V_E = \frac{V_{CC}}{2} = \frac{3.3V}{2} = 1.65V \tag{2.2}$$

$$R_E = \frac{V_E}{I_{CQ}} = \frac{1.65V}{7.5mA} = 220\Omega \tag{2.3}$$

$$V_B = V_E + 0.7V = 1.65V + 0.7V = 2.35V \tag{2.4}$$

$$I_B < 50\mu A \tag{2.5}$$

The datasheet specifies the base current (I_B) as less than $50\mu A$ with $2.35V$ collector-emitter voltage and $7.5mA$ collector current.

Let's make the bottom bias resistor (R_b) 100 times the size of the emitter resistor (R_E):

$$R_b = 100R_E = 100 \cdot 220\Omega = 22k\Omega \tag{2.6}$$

The bias voltage divider then becomes:

$$
\begin{aligned}
V_{CC} \cdot \frac{R_b}{R_a + R_b} &= V_B && (2.7)\\
\frac{V_{CC}}{V_B} \cdot R_b &= R_a + R_b && (2.8)\\
\frac{V_{CC}}{V_B} \cdot R_b - R_b &= R_a && (2.9)\\
\frac{3.3V}{2.35V} \cdot 22k\Omega - 22k\Omega &= R_a && (2.10)\\
8.894k\Omega &= R_a && (2.11)
\end{aligned}
$$

Let's use $10k\Omega$ for R_a. This will make the current through the bias approximately $100\mu A$ which is at least twice as much as the $I_B < 50\mu A$ obtained from the datasheet.

2.7 Low-pass filter

The output from the amplifier is fed through a simple RC low-pass filter to remove higher harmonics from the stair-step shaped output of the DAC. Note that this filter is very simple and therefore far from ideal and therefore it relies on the radio to further filter the signal before transmission.

Since we want to be able to produce signals up to 8 kHz, we want the filter cutoff frequency right around there. Using a 2.2kΩ resistor and a 10nF capacitor gets us close:

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 2.2k\Omega \cdot 10nF} = 7.234 \text{ kHz} \tag{2.12}$$

Being almost 800 Hz below the ideal cutoff is acceptable in this case because RC filters have a very slow rolloff. Therefore, the attenuation at 8 kHz isn't anywhere near enough to worry about it—especially since the vast majority of the signal will be at frequencies much lower than this extreme.

To be more explicit about the attenuation, the RC filter forms a voltage divider. The impedance of the capacitor at 8 kHz is:

$$X_C = \frac{1}{2\pi f C} = \frac{1}{2\pi \cdot 8\text{ kHz} \cdot 10nF} = 1.989k\Omega \tag{2.13}$$

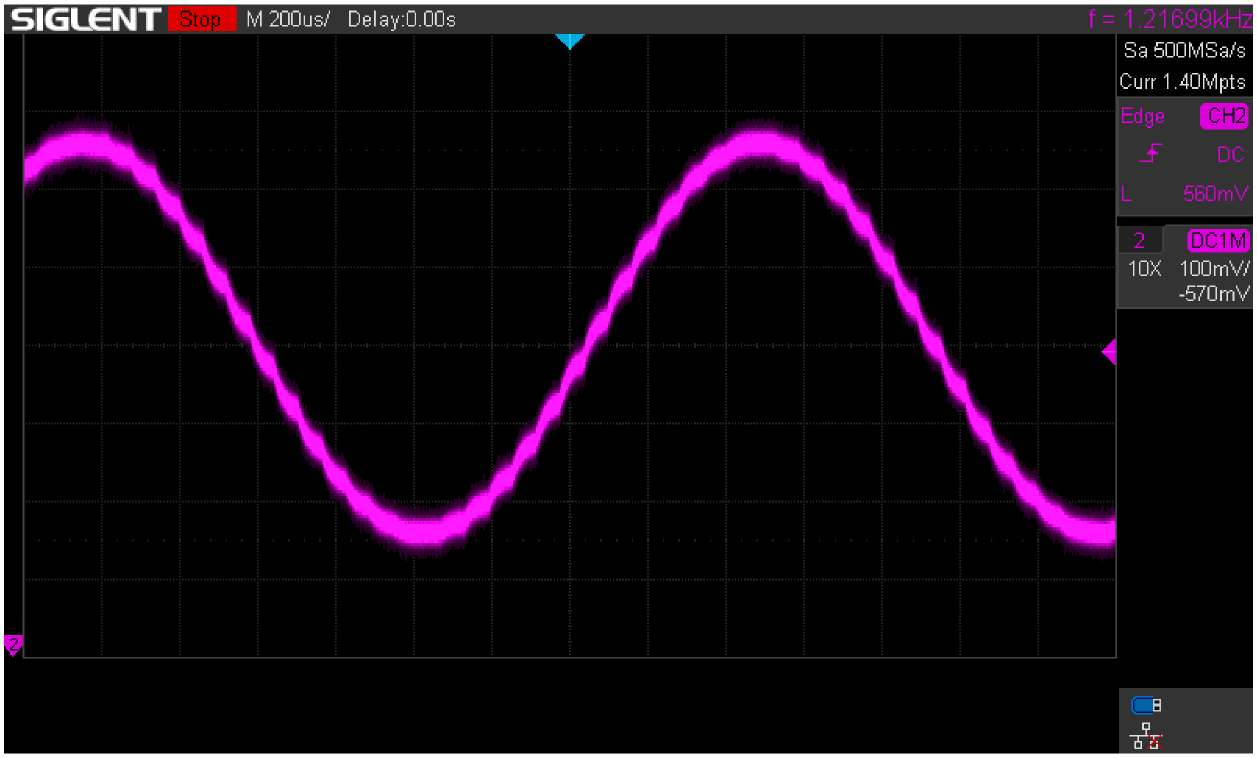
Yielding attenuation of:

$$\frac{V_{out}}{V_{in}} = \frac{X_C}{X_C + R} = \frac{1.989k\Omega}{1.989k\Omega + 2.2k\Omega} = 0.473 \tag{2.14}$$

This is very close to the 0.5 that occurs at the cutoff frequency.

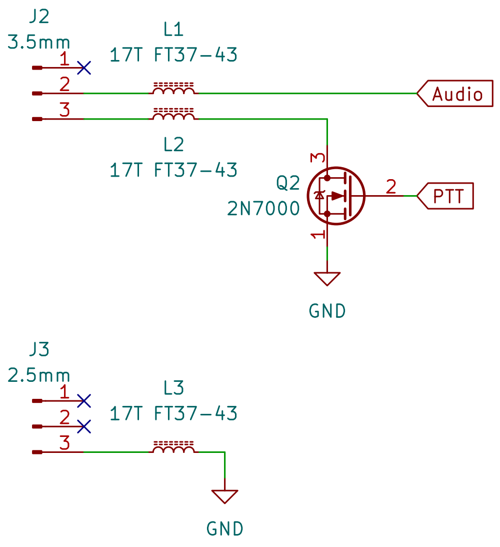
Moving the cutoff frequency higher results in insufficient attenuation of higher harmonics.

Below is an example of the filter's output when generating a sine wave. The previously (see example screenshot in section 2.5) very visible stair-step approximation of a sine wave is now much smoother, but still with noticeable high-frequency components.



2.8 Baofeng Interface

The main circuit generates audio and PTT signals. The (already AC-coupled) audio can be fed directly to the correct pin of the radio. The PTT signal can be used to control a transistor to key the radio by grounding its PTT pin.



A 2N7000 MOSFET is cheap and easy to obtain and therefore an obvious choice for the PTT transistor. The only potential issue with it is its $V_{GS(th)}$. According to the datasheet, the typical value is 2.1V and the maximum is 3V. This is a bit close when using 3.3V logic.

Worst case, one can always measure the threshold voltage of a particular transistor during assembly to ensure its performance is acceptable.

Finally, each of the three connections to the radio (audio, PTT, ground) uses a ferrite choke to prevent RF coming from the radio from entering the fox circuit.

Chapter 3

Software

This chapter talks about some of the major ideas behind the software design without going into too many implementation details to avoid duplicating the information in the code and comments. The source code for the firmware is licensed under the MIT license (see appendix E for a copy) and can be found in a Mercurial repository at https://hg.sr.ht/∼jeffpc/fox.

3.1 Toolchain

Most AVR-based projects created these days use the Arduino ecosystem of libraries. Since one of my personal goals for this project was to learn how to use the ATmega48P chip, I opted to use the avr-gcc and binutils for the tool chain and to not rely on *any* libraries. This led to a longer development time, but also a better understanding of how the various on-chip components interact and about their limitations.

3.2 Execution Flow

Overall, the firmware is very interrupt-centric. After the post-poweron startup code initializes the microcontroller as needed, the execution stops by using the `sleep` instruction. From that point on, all code that executes is in the interrupt handler function.

The timer1 comparator A is configured to fire 16384 times per second (see section 2.4) and each time it fires, the interrupt handler:

1. Outputs the previous sample

2. Generates the next sample and stores it for the next interrupt

This one sample delay has two major benefits: (1) it allows the output of the sample via SPI to occur with exactly the same latency since the start of the interrupt reducing jitter, and (2) it allows more complete overlap of SPI output with computation enabling more complex calculations.

3.3 Wave Generation

The firmware contains a 256-entry lookup table of a sine wave. If the code were to step through this table one entry per sample, it would generate a 64Hz sine wave ($\frac{16384}{256}$). To get higher frequency tones, the code steps through the sine wave lookup table at a faster rate. For example, a 128Hz tone can generated by skipping every other entry in the table.

While trivial to implement, this scheme has a serious downside—only frequencies that are an integer multiple of 64Hz can be generated. This is fine with modes such as the Morse code ident, but will likely make melodic tone patterns sound out of tune[1].

3.4 Configuration

The firmware has three major modes of operation. In addition to the mode, the microcontroller's EEPROM contains a number of parameters dictating the exact behavior.

The parameters are:

Station callsign

Mode of operation The modes of operation are discussed below in section 3.4.1.

Number of tones in pattern to play Certain modes could in theory play indefinitely. Their actual duration is specified as the number of tones in the pattern to play before stopping.

Key-on delay Some radios have a considerable delay between PTT getting triggered and audio starting to be transmitted. This parameter specifies how many samples[2] to delay after actuating the PTT before playing the tone pattern. It can be any integer in the range $[0,65536)$, corresponding to $[0,4)$ seconds.

[1]The plan is to address this limitation in revision B.

[2]There are 16384 samples/second.

Key-off delay Similar to the key-on delay, but specifies the delay between end of the last tone and PTT getting released.

Stride delay This specifies the inter-transmission time in seconds. It can be any multiple of 4 in the range [4,1020]. This gives the range from 4 seconds to 17 minutes.

The inter-transmission time is the time between the beginnings of two consecutive transmissions. Note, however, that if the transmission itself takes more than the inter-transmission time, the fox will wait until the next multiple of the inter-transmission time.

For example, if set to 100 seconds with a 20 second transmission, then the fox will transmit for 20 seconds, stay off-air for 80 seconds, and the start the transmit cycle again. However, if set to 30 seconds with a 35 second transmission, then the fox will transmit for 35 seconds, wait 25 seconds off-air, and then start the transmit cycle again. This is the same as if the inter-transmission time was set to 60 seconds.

The contents of the EEPROM can be generated by using the `gen-eeprom.py` script.

3.4.1 Modes

The modes are:

Fixed Pattern A fixed pattern of 10 tones, roughly corresponding to: C6, C5, D6, D5, E6, E5, G6, G5, A6, and A5. Each tone is played for 0.25 seconds. This mode plays the tones in sequence until the configured number of tones was generated (see section 3.4).

LFSR Pattern A pattern generated by using a 4-bit linear feedback shift register (LFSR) mapped to the tones: E4, G4, A4, C5, D5, E5, G5, A5, C6, D6, E6, G6, A6, C7, and D7. Each tone is played for 0.25 seconds. This mode advances the LFSR state until the configured number of tones was generated (see section 3.4).

Morse Code Ident The station callsign is transmitted *once* in Morse code at approximately 14.4 words per minute (using the PARIS standard)[3]. The station callsign is transmitted using the D5 tone, which corresponds to 587 Hz.

Note that the exact frequencies may sound somewhat out of tune due to the wave generation limitation described in section 3.3.

[3]
$$\left(50\frac{dits}{word}\right)^{-1} \cdot \left(\frac{1365\,samples/dit}{16384\,samples/s}\right)^{-1} \cdot 60\frac{s}{min} = 14.40\frac{words}{min}$$

3.5 State Machine

The description of the various configuration parameters in section 3.4 implied a number of states that the firmware automatically sequences through. The following table enumerates them and provides information about the PTT line, the audio output, as well as a description of what the state is for.

State	PTT	Audio	Description
STOPPED	off	none	Waiting for the beginning of next inter-transmission time (see “stride delay” in section 3.4)
KEY-ON-DELAY	on	none	A delay to give the transmitter time to power-on (see “key-on delay” in section 3.4)
KEY-OFF-DELAY	on	none	A delay after the end of transmission before PTT is released to make it sound less abrupt (see “key-off delay” in section 3.4)
PLAY-FIXED PLAY-LFSR	on	on	Tone pattern is produced (see “number of tones in pattern to play” in section 3.4, and section 3.4.1 for the pattern descriptions)
PLAY-IDENT	on	on	Morse code station callsign is produced (see “station callsign” is section 3.4)
PATTERN-PAUSE	on	none	A 0.5 second delay between a tone pattern and the Morse code station callsign

On power-on, the firmware transitions to the STOPPED state.

There it waits for the stride delay to end. When the delay ends, the firmware enables the PTT and moves to the KEY-ON-DELAY state.

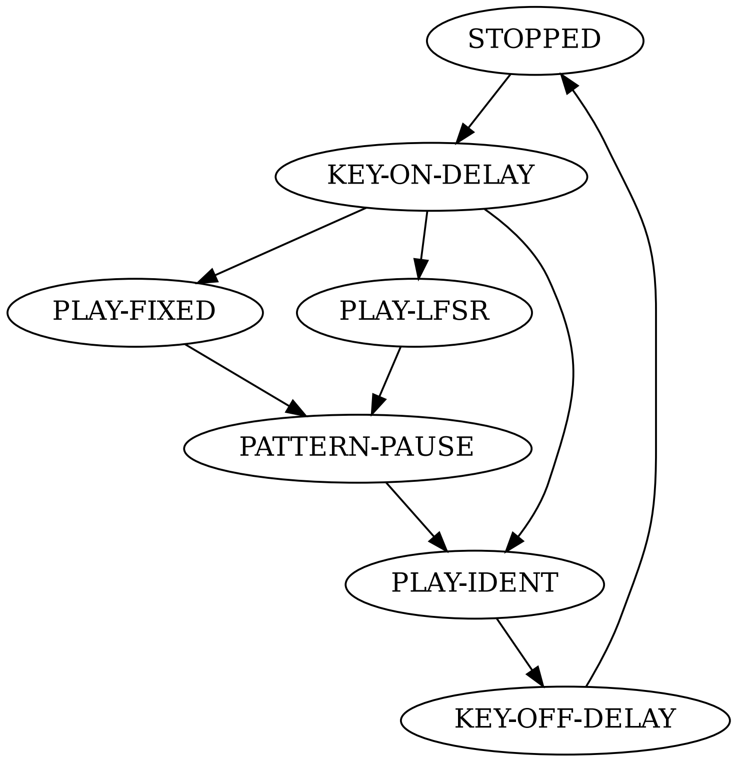
After the configured delay (the “key-on delay” in section 3.4), it moves to one of the three play states (PLAY-FIXED, PLAY-LFSR, and PLAY-IDENT) and begins playing the corresponding tone pattern or the Morse code station callsign.

When a tone pattern completes (PLAY-FIXED or PLAY-LFSR), there is a fixed pause of 0.5 seconds (the PATTERN-PAUSE state) before the firmware moves to the PLAY-IDENT state to transmit the station callsign in Morse code.

When the callsign transmission is complete (regardless of whether it was played after a pattern or as the only output), the PTT stays on for the configured time (the “key-off delay” in section 3.4) making the unkeying of the transmitter less abrupt.

Finally, after the key-off delay finishes, the firmware transitions back to the STOPPED state and the sequence begins again.

Visually, the states and transitions are:



Part II

Construction & Bring-up

Chapter 4

Construction

I used a stripboard for the circuit (see photo on the front cover page and the layout in appendix D). The steps I took are similar to the typical suggestions for how to solder components (i.e., start with resistors), but I chose to build everything in sections. This allowed me to check that everything constructed still worked before moving onto more components.

I started with the main circuit first, then the Baofeng interface board, and finally the power supply section. The main circuit and the power supply are on the same board, and the Baofeng interface is on a separate protoboard. I did this to allow for some flexibility in driving different radios with the same source.

See appendix C for the bill of materials.

Chapter 5

Step-by-step Instructions

This section includes a detailed list of steps I took while soldering the components on the stripboard. The steps are a bit tedious by checking that everything works in small increments. These small steps make it easier to find faults sooner and minimize the risk of damage to any of the components.

Once fully assembled, the expected voltages are as follows. The $\pm$ values show the expected voltage range while the firmware is producing audio. When there is no audio produced, the voltages should be in the middle of each range.

Signal	**Voltage range**	**Ref**
Vcc	3.3V	—
SPI data & clock	3.3V digital signals	—
Latch & PTT	3.3V digital signals	—
R8/R16 node	0–3.1V	2.5
Q1 base	2.35$\pm$0.75V	2.6.1
Q1 emitter	1.65$\pm$0.75V	2.6.1
Audio	$\pm$0.32V	A

5.1 Main Circuit

1. Cut traces on stripboard as required
2. Install U1 and U2 sockets
3. Install Y1, C1, C2, and C4
4. Install Vcc and GND wires from U1 socket to top supply rails

5. Install top supply rail header[1]
6. Check for shorts: apply 3.3V power to supply rail header, verify that there is no current draw, then remove power
7. Program U1 using an external programmer
8. Insert U1 into socket, apply 3.3V power to supply rail, verify the following, then remove power
    (a) Vcc is 3.3V, current draw is less than 25mA
    (b) U1 pin 3 has SPI data
    (c) U1 pin 5 has latch signal
    (d) U1 pin 6 has SPI clock
    (e) U1 pin 24 has PTT signal
9. Remove U1 from socket
10. Install Vcc and GND wires from U2 to top supply rails
11. Check for shorts: apply 3.3V power to supply rail header, verify that there is no current draw, then remove power
12. Insert U1 & U2 into socket, apply 3.3V power to supply rail, verify the following, then remove power
    (a) Vcc is 3.3V, current draw is less than 25mA
    (b) U2 pin 11 has SPI clock
    (c) U2 pin 12 has latch signal
    (d) U2 pin 14 has SPI data
    (e) U2 pins 15 and 1–7 have data
13. Remove U1 & U2 from socket
14. Install R1–16
15. Insert U1 & U2 into socket, apply 3.3V power to supply rail, verify the following, then remove power
    (a) Vcc is 3.3V, current draw is less than 25mA
    (b) The R8/R16 node has approximated sine output going between 0 and 3.3V (see section 2.5 for oscilloscope screenshot of what to expect)
16. Install R17–20, C5, Vcc and GND wires to R19 & R20

[1]This header is useful during bringup and other testing. It doesn't serve any purpose in the completed circuit and therefore it isn't shown on the schematic.

17. Apply 3.3V power to supply rail, verify the following, then remove power
    (a) Vcc is 3.3V, current draw is less than 25mA
    (b) The R17/R18 node has the attenuated DAC output signal
    (c) The R19/R20 node has the attenuated DAC output shifted to be centered around approximately 2.35V
18. Install Q1, R21, RV2
19. Apply 3.3V power to supply rail, verify the following, then remove power
    (a) Vcc is 3.3V, current draw is less than 25mA
    (b) The Q1 emitter has the attenuated DAC output signal
    (c) The RV2 wiper pin has an attenuated version of the Q1 emitter
20. Install R22, C6, and C7
21. Apply 3.3V power to supply rail, verify the following, then remove power
    (a) Vcc is 3.3V, current draw is less than 25mA
    (b) The audio output (C7) has the correct level, adjust RV2 as necessary

5.2 Baofeng Interface Board

1. Wind 17 turns of wire on L1–3 toroids
2. Install L1–3, Q2, 2.5mm and 3.5mm plugs, and wires for audio, PTT, and ground
3. Interface board with main circuit
4. Apply 3.3V power to supply rail, verify the following, then remove power
    (a) 2.5mm sleeve is grounded
    (b) 3.5mm ring has audio data
    (c) 3.5mm sleeve is open circuit when PTT is low, and grounded when PTT is high
5. Connect 2.5mm and 3.5mm plugs to Baofeng hand-held
6. Connect Baofeng to a dummy load
7. Apply 3.3V power to supply rail, verify the following, then remove power
    (a) The Baofeng keys & transmits the audio

5.3 Power Supply

1. Install U3, D1, D2, RV1, C3
2. Connect between 5.5V and 9V to the U3 input
3. Adjust RV1 until output produces 3.3V
4. Turn off power
5. Install Vcc wire from power supply output to the top Vcc rail
6. Connect between 5.5V and 9V to the U3 input
7. Check that the power supply input is drawing between 14 and 21mA.
8. Check that audio output and PTT signals are within expected ranges

Part III

Appendices

This part contains the following appendices:

Appendix A includes the datasheet.

Appendix B includes the schematic.

Appendix C lists the bill of materials.

Appendix D includes the stripboard layout.

Appendix E includes a copy of the MIT License under which the firmware and documentation is licensed.

Appendix F includes a copy of the CERN Open Hardware Licence Version 2 - Permissive under which the hardware is licensed.

Appendix A

Datasheet

Parameter	**Min**	**Typical**	**Max**	**Unit**	**Section**
Supply voltage (recommended)	5.5	—	9.0[1]	V	2.2
Supply voltage (tested)	5.0	—	10.0[1]	V	—
Supply current[2]	14	16.4	21	mA	—
Audio output voltage	0	±0.32	±0.73	V	—
PTT output voltage[3]	0	—	3.3	V	—
Transmitter PTT voltage (PTT-off)[4]	—	—	10	V	—
Transmitter PTT current (PTT-on)[4]	—	—	50	mA	—
Output sample rate	—	16384	—	Hz	2.4
Output sample bitwidth	—	8	—	bits	1 & 2.1

[1] The maximum is theoretically the same as LM317's maximum input voltage, however linear voltage regulators aren't particularly efficient when the input-output voltage difference is large.

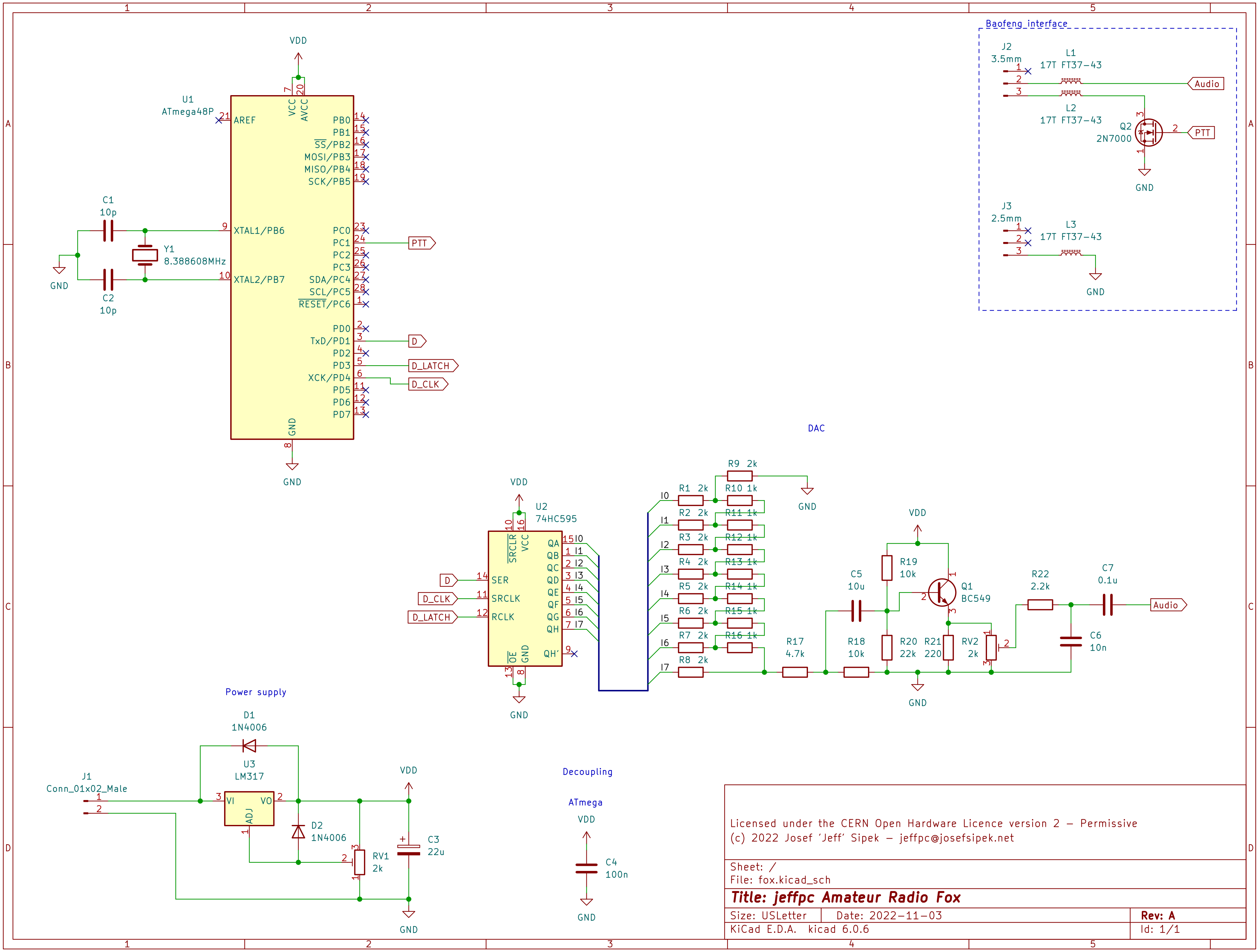
[2] Measured on LM317's input powered with 5.0V lab supply.

[3] This is the voltage going to the Baofeng interface's 2N7000 gate.

[4] This is a conservative value. For the actual maximum, refer to the 2N7000 datasheet with a gate voltage of 0V (PTT-off) and 3.3V (PTT-on).

Appendix B

Schematic



Licensed under the CERN Open Hardware Licence version 2 – Permissive
(c) 2022 Josef 'Jeff' Sipek – jeffpc@josefsipek.net

Sheet: /
File: fox.kicad_sch

Title: jeffpc Amateur Radio Fox

Size: USLetter	Date: 2022-11-03	**Rev: A**
KiCad E.D.A. kicad 6.0.6		Id: 1/1

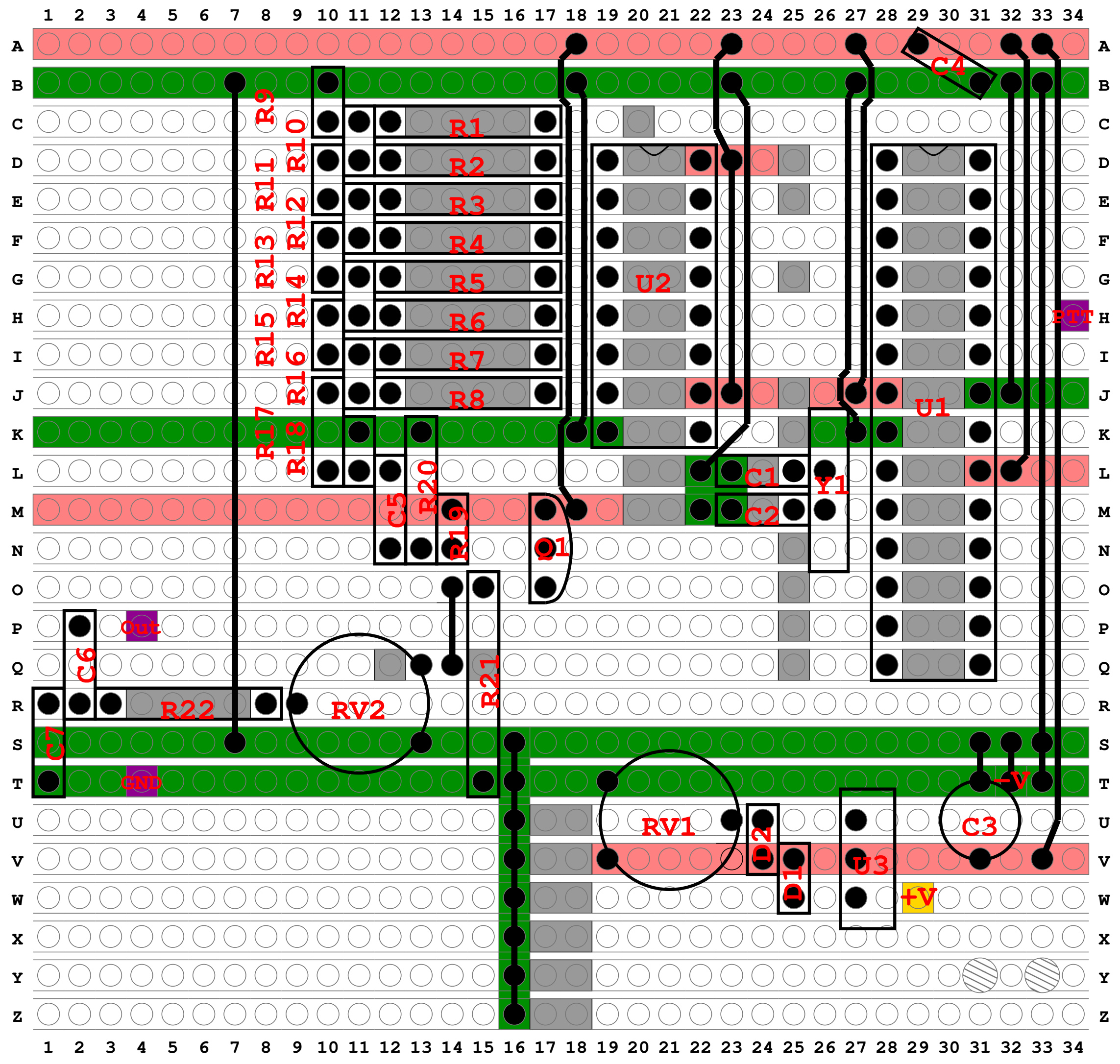
Appendix C

Bill of Materials

Values	References
10p	C1, C2
22u	C3
100n	C4
10u	C5
10n	C6
0.1u	C7
1N4006	D1, D2
Conn_01x02_Male	J1
3.5mm	J2
2.5mm	J3
17T FT37-43	L1, L2, L3
BC549	Q1
2N7000	Q2
2k	R1, R2, R3, R4, R5, R6, R7, R8, R9
1k	R10, R11, R12, R13, R14, R15, R16
4.7k	R17
10k	R18, R19
22k	R20
220	R21
2.2k	R22
2k	RV1, RV2
ATmega48P	U1
74HC595	U2
LM317	U3
8.388608MHz	Y1

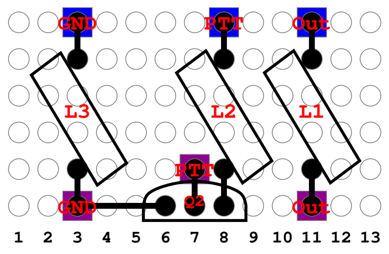
Appendix D

Board Layout



The above image shows the layout and all connections on the single-sided 7cm by 9cm stripboard with 26 by 34 holes I used for the main circuit and power supply.

I put the Baofeng interface on a separate protoboard to make it more future-proof. Note that unlike the stripboard, the protoboard does *not* have any of the holes connected and all the connections have to be created with jumpers.



○	Unused hole	PTT	PTT
●	Used hole/solder connection	Out	Audio
●━●	Jumper wire	GND	Ground
◙	Removed hole	PTT	Baofeng PTT
🟩	Ground	Out	Baofeng Audio
🟥	+3.3V	GND	Baofeng Ground
+V	Battery +V connection	Q1	Component label
−V	Battery −V connection	▨	Drilled out hole

Appendix E

MIT License

Appendix F

CERN Open Hardware Licence Version 2—Permissive

Preamble

CERN has developed this licence to promote collaboration among hardware designers and to provide a legal tool which supports the freedom to use, study, modify, share and distribute hardware designs and products based on those designs. Version 2 of the CERN Open Hardware Licence comes in three variants: this licence, CERN-OHL-P (permissive); and two reciprocal licences: CERN-OHL-W (weakly reciprocal) and CERN-OHL-S (strongly reciprocal).

The CERN-OHL-P is copyright CERN 2020. Anyone is welcome to use it, in unmodified form only.

Use of this Licence does not imply any endorsement by CERN of any Licensor or their designs nor does it imply any involvement by CERN in their development.

1 Definitions

1.1 'Licence' means this CERN-OHL-P.

1.2 'Source' means information such as design materials or digital code which can be applied to Make or test a Product or to prepare a Product for use, Conveyance or sale, regardless of its medium or how it is expressed. It may include Notices.

1.3 'Covered Source' means Source that is explicitly made available under this Licence.

1.4 ’Product’ means any device, component, work or physical object, whether in finished or intermediate form, arising from the use, application or processing of Covered Source.

1.5 ’Make’ means to create or configure something, whether by manufacture, assembly, compiling, loading or applying Covered Source or another Product or otherwise.

1.6 ’Notice’ means copyright, acknowledgement and trademark notices, references to the location of any Notices, modification notices (subsection 3.3(b)) and all notices that refer to this Licence and to the disclaimer of warranties that are included in the Covered Source.

1.7 ’Licensee’ or ’You’ means any person exercising rights under this Licence.

1.8 ’Licensor’ means a person who creates Source or modifies Covered Source and subsequently Conveys the resulting Covered Source under the terms and conditions of this Licence. A person may be a Licensee and a Licensor at the same time.

1.9 ’Convey’ means to communicate to the public or distribute.

2 Applicability

2.1 This Licence governs the use, copying, modification, Conveying of Covered Source and Products, and the Making of Products. By exercising any right granted under this Licence, You irrevocably accept these terms and conditions.

2.2 This Licence is granted by the Licensor directly to You, and shall apply worldwide and without limitation in time.

2.3 You shall not attempt to restrict by contract or otherwise the rights granted under this Licence to other Licensees.

2.4 This Licence is not intended to restrict fair use, fair dealing, or any other similar right.

3 Copying, Modifying and Conveying Covered Source

3.1 You may copy and Convey verbatim copies of Covered Source, in any medium, provided You retain all Notices.

3.2 You may modify Covered Source, other than Notices.

You may only delete Notices if they are no longer applicable to the corresponding Covered Source as modified by You and You may add additional Notices applicable to Your modifications.

3.3 You may Convey modified Covered Source (with the effect that You shall also become a Licensor) provided that You:

a) retain Notices as required in subsection 3.2; and

b) add a Notice to the modified Covered Source stating that You have modified it, with the date and brief description of how You have modified it.

3.4 You may Convey Covered Source or modified Covered Source under licence terms which differ from the terms of this Licence provided that You:

a) comply at all times with subsection 3.3; and

b) provide a copy of this Licence to anyone to whom You Convey Covered Source or modified Covered Source.

4 Making and Conveying Products

You may Make Products, and/or Convey them, provided that You ensure that the recipient of the Product has access to any Notices applicable to the Product.

5 DISCLAIMER AND LIABILITY

5.1 DISCLAIMER OF WARRANTY – The Covered Source and any Products are provided 'as is' and any express or implied warranties, including, but not limited to, implied warranties of merchantability, of satisfactory quality, non-infringement of third party rights, and fitness for a particular purpose or use are disclaimed in respect of any Source or Product to the maximum extent permitted by law. The Licensor makes no representation that any Source or Product does not or will not infringe any patent, copyright, trade secret or other proprietary right. The entire risk as to the use, quality, and performance of any Source or Product shall be with You and not the Licensor. This disclaimer of warranty is an essential part of this Licence and a condition for the grant of any rights granted under this Licence.

5.2 EXCLUSION AND LIMITATION OF LIABILITY – The Licensor shall, to the maximum extent permitted by law, have no liability for direct, indirect, special, incidental, consequential, exemplary, punitive or other damages of any character including, without limitation, procurement of substitute goods or services, loss of use, data or profits, or business interruption, however caused and on any theory of contract, warranty, tort (including negligence), product liability or otherwise, arising in any way in relation to the Covered Source, modified Covered Source and/or the Making or Conveyance of a Product, even if advised of the possibility of such damages, and You shall hold the Licensor(s) free and harmless from any liability, costs, damages, fees and expenses, including claims by third parties, in relation to such use.

6 Patents

6.1 Subject to the terms and conditions of this Licence, each Licensor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section 6, or where terminated by the Licensor for cause) patent licence to Make, have Made, use, offer to sell, sell, import, and otherwise transfer the Covered Source and Products, where such licence applies only to those patent claims licensable by such Licensor that are necessarily infringed by exercising rights under the Covered Source as Conveyed by that Licensor.

6.2 If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Covered Source or a Product constitutes direct or contributory patent infringement, or You seek any declaration that a patent licensed to You under this Licence is invalid or unenforceable then any rights granted to You under this Licence shall terminate as of the date such process is initiated.

7 General

7.1 If any provisions of this Licence are or subsequently become invalid or unenforceable for any reason, the remaining provisions shall remain effective.

7.2 You shall not use any of the name (including acronyms and abbreviations), image, or logo by which the Licensor or CERN is known, except where needed to comply with section 3, or where the use is otherwise allowed by law. Any such permitted use shall be factual and shall not be made so as to suggest any kind of endorsement or implication of involvement by the Licensor or its personnel.

7.3 CERN may publish updated versions and variants of this Licence which it considers to be in the spirit of this version, but may differ in detail to address new problems or concerns. New versions will be published with a unique version number and a variant identifier specifying the variant. If the Licensor has specified that a given variant applies to the Covered Source without specifying a version, You may treat that Covered Source as being released under any version of the CERN-OHL with that variant. If no variant is specified, the Covered Source shall be treated as being released under CERN-OHL-S. The Licensor may also specify that the Covered Source is subject to a specific version of the CERN-OHL or any later version in which case You may apply this or any later version of CERN-OHL with the same variant identifier published by CERN.

7.4 This Licence shall not be enforceable except by a Licensor acting as such, and third party beneficiary rights are specifically excluded.