# High Frequency Magnetic Loop Antenna Auto-Tuner for Amateur Radio

Gregory Raven KF5N

Currently Being Revised May 2023

# Contents

# Chapter 1

# Introduction

This is the construction manual and "user guide" for a magnetic loop antenna controller which is patterned after a design published in the book "Microcontroller Projects for Amateur Radio" by Jack Purdum W8TEE and Albert Peter AC8GY. The author's firmware for the antenna controller was developed in the Arduino IDE, and the controller uses the STM32 "Blue Pill" module. The book includes a detailed description and construction guide for both the antenna and the controller. The book is available here (click on the link):

Microcontroller Projects for Amateur Radio

This derivative design replaces the STM32 "Blue Pill" controller module with a Raspberry Pi Pico. I was experimenting with the Pi Pico, and I noticed the Pi Pico was very similar in size to the Blue Pill module, and it had the necessary GPIOs and ADC (Analog to Digital Converter) to perform the same function as the Blue Pill board. Another reason to use the Pi Pico is that it has not been affected by "supply chain" problems which have plagued other electronic endeavors. The official vendors are listed in the "Buy now" button, however, the boards can also be purchased at Amazon, eBay, and Digikey.

https://www.raspberrypi.com/products/raspberry-pi-pico/

The firmware was developed in the Visual Studio Code text editor which is provisioned by a script provided by Raspberry Pi. This script installs and configures everything required to develop firmware for the Pi Pico. The C/C++ "Software Development Kit" (SDK) provided by Raspberry Pi has a wealth of functionality to control the peripheral components of the Pi Pico. The documentation for the Pi Pico is available here:

Raspberry Pi Pico Documentation

One of the prime reasons to chose the Pi Pico is that the development kit includes a "debugger", which is a feature the Arduino IDE lacks. However, this may change as the Arduino project migrates to the 2.0 IDE.

As the original project was developed as an Arduino project, a lot of "hacking" of the

code was required to adapt it to the Raspberry Pi SDK. The first experiment was to hack the display library since this is the most complex library in the project. After a few days of experimenting with both the library code and Cmake, the hacked display library was functioning! The other libraries were much easier to translate to the Pi Pico SDK.

In the future, a version of firmware may be provided which uses the Arduino IDE. It should be possible, however, at the moment I can't support it.

Although this project follows the same pattern as the original, there are other significant hardware changes:

- Replaced large external stepper driver with A4988 stepper driver module. The module is much smaller and is integrated into the PCB.
- Replaced MC1458 op amps with LM358. This eliminates the requirement for a negative supply voltage, which in turn eliminates the negative DC supply module.
- A BNC RF connector is added.
- An Ethernet connector is added. This is used to connect the controller to the antenna tuner motor and limit switches.
- A coaxial power connector is added for the +12 volt power required.
- Electronic switches were added to save standby power consumption.
- A +12 Volt relay control output was added.
- The number of pushbuttons required was reduced from seven to three. This was allowed due to simplification of the user interface.
- An audio muting board can be added to mute speaker audio during the auto-tuning process.
- A shunt diode plus fuse style reverse polarity protection was added.

The buttons and menu system are de-bounced.

The PCB is 100mm by 100mm and can be fabricated for approximately US $1 per board (not including shipping cost).

The enclosure is 3D printed and is similar to the design of the original project.

The controller should function with the original project's "double-double" loop antenna. I decided to go a simpler route with a basic single-loop design. The loop is resonated with an eBay sourced butterfly capacitor. The antenna uses a combination of PVC plumbing and 3D printed parts.

The remainder of this document should provide an experienced homebrewer with enough information to duplicate the controller and antenna. The Github repositories containing the project information will be periodically updated as the firmware and components continue to evolve.

## 1.1 Loop Antenna System Block Diagram

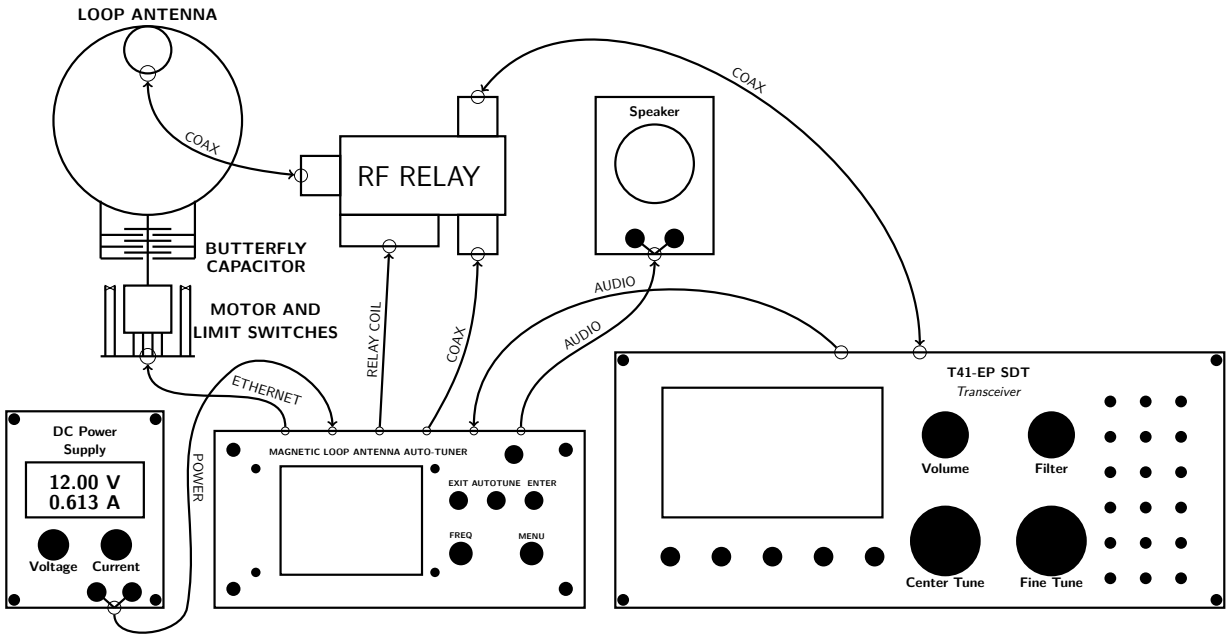


**Figure 1.1: Components of an Amateur Radio Station using the Loop Antenna Controller**

## 1.2 What is the Power Consumption?

The controller requires +12 Volts DC. There are two power consumptions:

1. AutoTune: A current drain of 0.6 Amps which is about 7.2 watts.
2. Normal operation: A current drain of 0.12 amps which is about 1.5 watts.

The AutoTune mode is engaged only when re-tuning to a new frequency. Thus the controller will be in “Normal Operation” the majority of the time during use.

## 1.3 Level of Difficulty

This is not a beginner’s project! You will need a good collection of tools, including a soldering station, and good skills in PCB construction and 3D printing. It will be helpful to have a DVM, oscilloscope, and maybe even a NanoVNA for testing and troubleshooting.

I do not provide detailed construction information for the antenna and tuner assembly. There are photos, and STL files for the 3D printed parts. In my case, I mounted the tuner parts (stepper motor and butterfly capacitor) on a rectangular piece of Plexiglas. There is a lot of measuring and drilling involved to get this just right. Getting

the motor positioned perfectly so that the belt drive works smoothly with the butterfly capacitor is paramount. Take your time and measure carefully before any drilling begins.

It is not mandatory, but I found a small HF receiver capable of SSB/CW to be very useful in monitoring the controller's DDS frequency generator. It doesn't need to be connected to the DDS. It can be on the same bench with the whip antenna extended a little bit. There will be more than enough signal for the receiver to verify the DDS is operating on the correct frequency.

## 1.4 Caveat Emptor

My biggest concern with this project is undiscovered RFI and EMI problems. It needs more on-air testing. The problems could manifest themselves as interference to the receiver from spurious energy from the Pi Pico or other components. Perhaps the transmitter could "jam" the controller and cause erratic operation.

Putting the controller in the 3D printed and unshielded box is a risk for EMI problems. The PCB design uses good EMI design practices wherever possible. However, it is possible the controller really needs to be in a metallic box. It will require more testing "in the wild" to know for sure.

In general, I believe this derivative design will perform much the same as the design it was derived from. However, there are enough changes and additions that new problems could appear.

That should make it clear that this project is a "prototype" and not a proven design. It has no warranties or guarantees of any kind. The best of luck to those who decide to take this project on! If you do have success ()or failure) with this project, I hope you will share your experience on the Software Controlled Amateur Radio group at groups.io.

# Chapter 2

# Controller Hardware

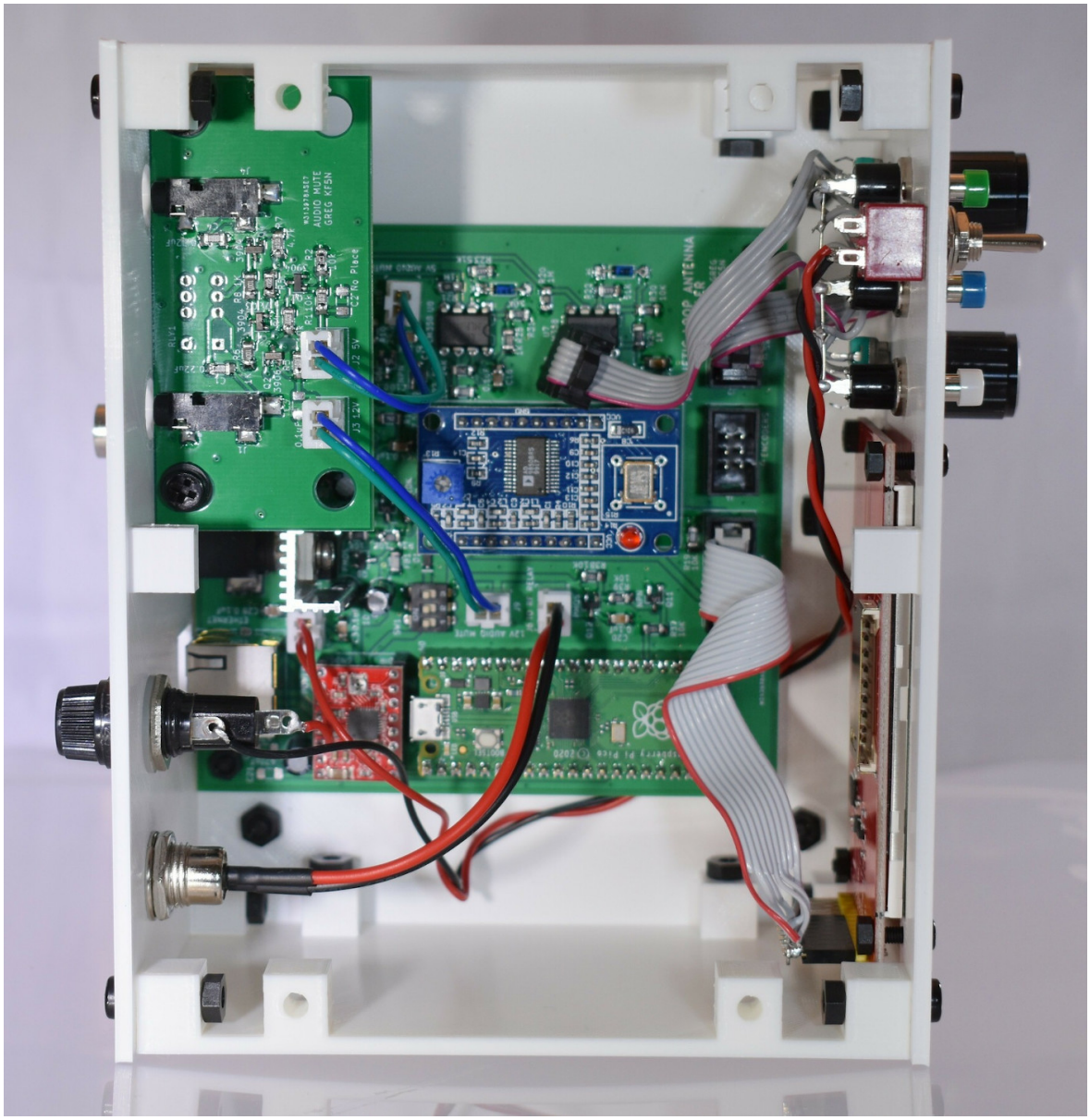


**Figure 2.1: Top view of the fully assembled loop antenna controller.**

## 2.1 Microcontroller

The original design used an STM32-based board commonly known as the ”Blue Pill”. This is a single-core ARM microcontroller with 64kb of Flash memory, and 20kb of RAM. It has an Analog to Digital Converter (ADC) which is required by the SWR measuring function of the project. The board is supported in the Arduino IDE and has a large range of supporting libraries.

A drawback of the “Blue Pill” in the Arduino IDE is that there is more than one source for the supporting libraries. These libraries are not necessarily interchangeable. Configuration of the IDE and libraries can be confusing.

Compounding the ”Blue Pill” problem is the numerous variants of the board with microcontrollers sourced from various vendors. There are geniune ST Micro devices, unauthorized clones, and authorized clones. The officially supported Arduino support software will reject the unauthorized clone devices. Alternative support libraries may or may not work with the clones.

When you buy a “Blue Pill”, you may not be able to determine exactly what you are getting.

In early 2021 Raspberry Pi made a surprise announcement of a new microcontroller product called the “Pi Pico”. The microcontroller is a dual-core ARM system-on-chip with an ADC, and enough general purpose input-output (GPIO) to support the magnetic loop controller. Flash memory is a much larger 2MB with 260kB of RAM.

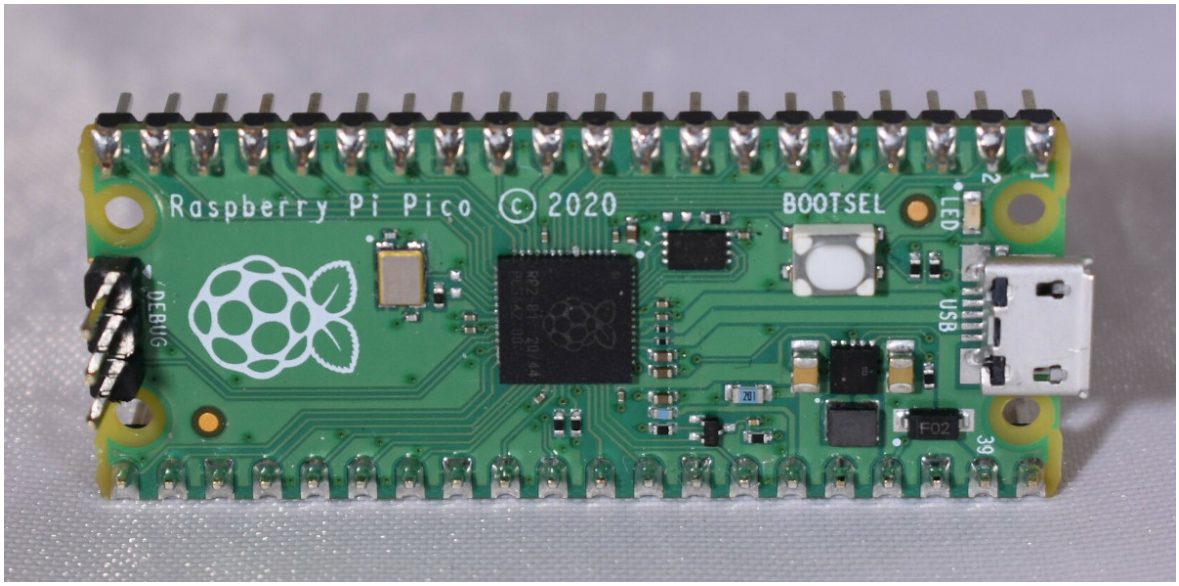


**Figure 2.2: The Raspberry Pi Pico board. Note the three-pin debug header on the left side of the board.**

The Pi Pico form factor is similar to the Blue Pill. The price is also similar, with an advertised cost of US $4 (not including shipping). Pi Pico boards have been

consistently available in spite of the semiconductor supply chain issues seen in recent months.

Making the Pi Pico greatly attractive is the expansive and well-supported Software Development Kit (SDK) along with numerous examples. The downside of this new device is that there are not many libraries available, although the ecosystem is continuing to expand.

Arduino support for t Pi Pico is available, however, this author has not explored this option yet.

## 2.2 DDS Module, RF Amplifier and SWR Bridge

The DDS Module, RF Amplifier, and SWR bridge circuits are direct copies of the original project.

The DDS Module is based on the AD9850 integrated circuit. The module can be ordered from Amazon or eBay.

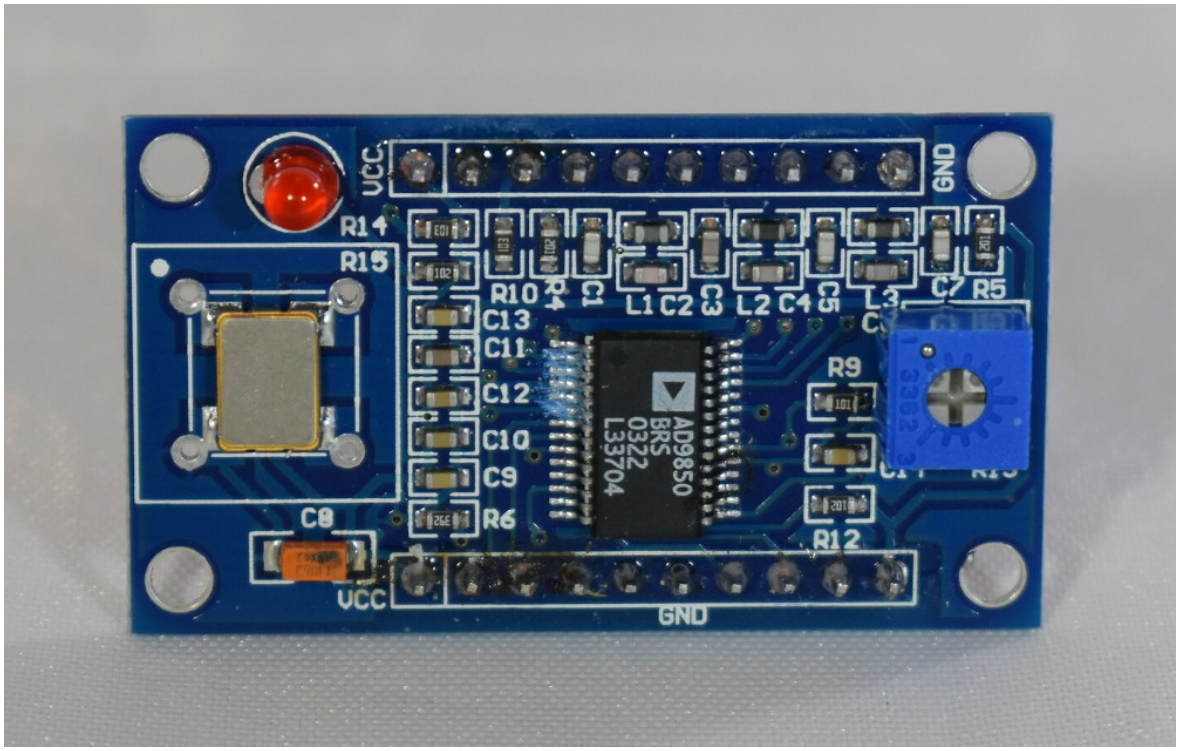


**Figure 2.3: AD9850 DDS Module**

The RF amplifier consumes significant current, so it was decided to add a PMOS switch connected to a GPO so that the amplifier can be deactivated when not required.

One quirk of the SWR bridge circuit is that it is sensitive to the forward voltage drop of the 1N5711 diodes used. This will not affect the ability of the controller to tune to

minimum SWR, but it will affect the accuracy of the SWR reported on the display. I observed improved accuracy when the forward voltage drop of the 1N5711 diodes is lower. Diodes ordered from Amazon by “Chanzon” performed the best.

## 2.3 Stepper Motor

The stepper motor is the most important component in determining the tuning resolution of the system.

I will describe the tuning algorithm implemented in the controller’s firmware. Understanding this will make clear why the stepper motor is critical in determining the tuning resolution.

A magnetic loop antenna behaves like an inductor-capacitor (LC) resonant circuit. The ”inductor” in this case is the large metallic loop, which is also the radiating structure of the antenna.

The loop is designed for low-loss, and therefore a high “Q factor”, which means the resonance is going to be very sharp. This means the antenna is also a narrow bandpass filter.

Now looking at the mechanics of the tuning capacitor, this means that the resonant frequency is going to have a high rate-of-change as the capacitor is rotated. This can be quantified in ”Hertz per degree”.

The problem here is that it may be required to rotate the capacitor by a fraction of a degree in order to tune to the target frequency selected by the user.

How good can we do with an off-the-shelf stepper motor? The most common motors specify 200 steps per rotation. Dividing 360 degrees by 200 and the result is 1.8 degrees per step. That seems a bit coarse!

However, it is recommended to use a belt drive to obtain a ”drive ratio”. Common components used in 3D printers are readily available. Toothed pulleys and belts make an excellent no-slip drive system. A common pulley used on the motor shaft has 20 teeth, and combining this with a 60 teeth pulley on the capacitor results in a 3:1 drive. So now the 1.8 degrees per step is divided by 3 for 0.6 degrees per step. Better!

However, working against us is the butterfly capacitor. Unlike a regular variable capacitor, the butterfly capacitor goes from maximum to minimum capacitance in only 90 degrees of rotation. Thus using a butterfly style capacitor means better resolution is required. The advantages of the butterfly capacitor outweigh this one downside.

An option to improve resolution by a factor of two is to buy a more expensive stepper motor. It is possible to buy a stepper with 400 steps per resolution, or 0.9 degrees per step. I have done this and it does help to improve the tuning accuracy of the system.

The cost is about double for a 0.9 degree stepper motor, however, I believe this is a good investment and will pay for itself in improved tuning accuracy.

## 2.4 Stepper Motor Driver Version 1

The original project used a large external stepper motor driver called the TB6600. This is a robust device with DIP switches to control step size and current drive. It is capable of 1/32 microsteps. The main problem with this device is that it is larger than the entire PCB and needs to stand up vertically. So the project's case must be much larger to accommodate this stepper driver.

Meanwhile, I had taken the cover off the controller of my 3D printer, and discovered that it used very small plug-in stepper driver boards. These driver boards are about the size of a postage stamp! They are based on the A4988 driver device. However, these drivers are limited to 1/16 microsteps. This requires an external DIP switch or resistors to select. The current is adjusted by a very tiny potentiometer on the board. The authors of the original project stated they used 1/16 microsteps, so perhaps this small board would be adequate. So far it has! The size reduction of the finished device in the case is substantial. The A4988 cost is about $2, versus $12 for the TB6600, so it is a nice cost reduction as well. The modules can be purchased from Amazon or eBay.

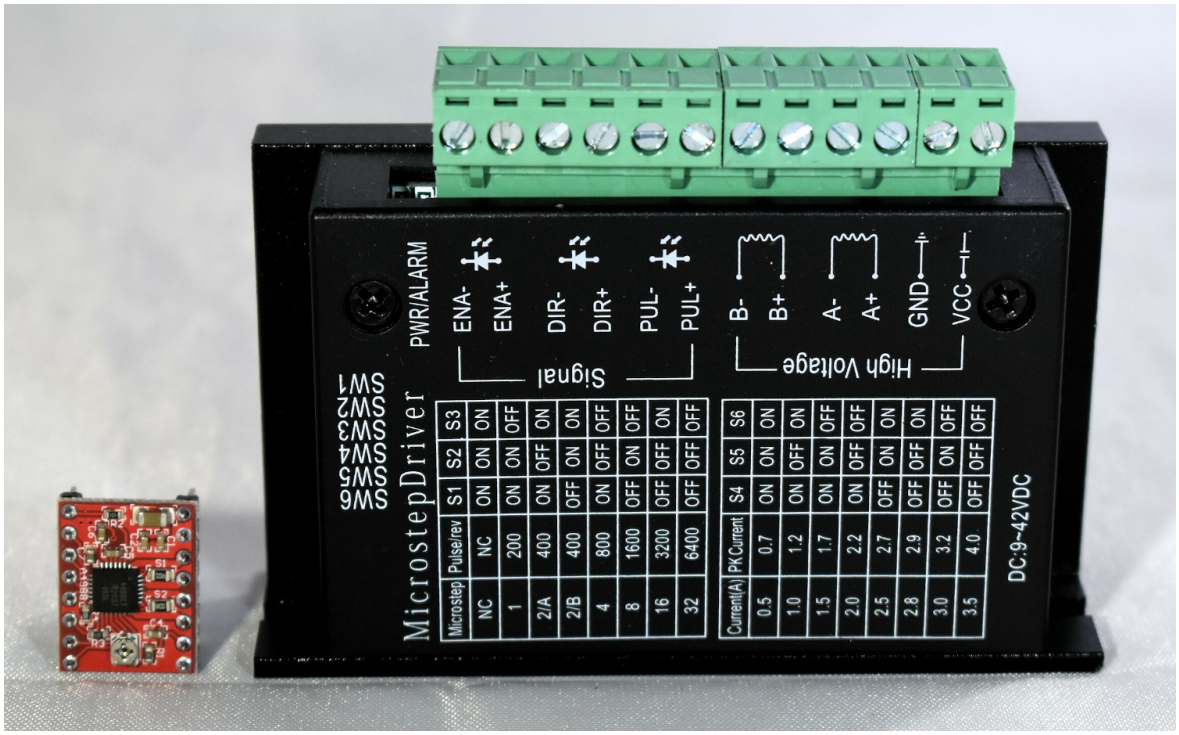


**Figure 2.4: Size comparison: A4988 Module on the left, and the TB6600 driver on the right**

## 2.5 Stepper Motor Driver Version 2

A further investigation of stepper drivers revealed the so-called “silent-stepper stick”. This is another stepper driver board in the same format as the A4988 (and others).

The integrated circuit used on these silent-stepper sticks are from the company Trinamic:

https://www.trinamic.com/products/integrated-circuits/details/tmc2225-sa/

The above link is to Trinamic’s TMC2225 stepper driver IC. “Stepper sticks” using this device are readily available from Amazon:

https://www.amazon.com/gp/product/B09NCYZKQC

The advantages of this sophisticated stepper driver are significant, and this stepper driver is the primary reason for Version 2 of the loop controller.

The TMC stepper driver has important features which are utilized to improve tuning accuracy. The TMC driver has features which allow the stepper motor to be very gently turned off. Next, I will attempt to explain why this is important.

### 2.5.1 The Tuning Algorithm

To design the system for repeatable and accurate tuning requires understanding of what happens during the tuning process.

The following assumes that the initial tuning calibration has been completed successfully. Tuning calibration uses a variant of this same tuning algorithm. Tuning calibration finds the “end points” or the stepper position for the upper and lower band limits for each band. The stepper position is relative to stepper position zero, which is the position at which the capacitor is fully meshed. Stepper zeroing is automatically executed whenever the controller is powered on, or at the beginning of the initial calibration. Zeroing can also be executed manually from the Calibration menu.

I will describe what happens when an operator is using a particular frequency, and then decides to move to another frequency within the same band.

1. Using the pushbuttons and encoders, a new frequency is entered and the user presses the button “Autotune”. The controller algorithm proceeds as follows.

2. The controller uses the stepper position band limits to estimate the stepper position for the newly entered frequency. This is done using the mathematical process of interpolation.

3. The controller commands the stepper motor to move to the estimated position, minus a fixed offset (“Coarse Tuning”). This way, the position of the capacitor is such that the resonant frequency of the antenna will be lower than the desired target frequency. The reason for this is that the tuning process will always

approach the desired frequency from below. See the explanation of "backlash" below.

4. Now the controller will begin commanding the stepper to move upwards. After each stepper movement, the SWR is measured. During the first few steps the SWR will be very high, typically 10:1 or more. While the SWR is high, the stepper will be commanded to move in "Coarse Steps", the value of which can be changed by the user in the Hardware Settings menu.

5. The process will continue until the SWR falls below 3:1. The controller switches to single-steps in order to get as close to resonance as possible. The SWR will be recorded at each step, and the lowest SWR step will be saved. The sweep will continue until either sweeping past the minimum SWR step plus coarse steps, or the SWR increases above 3:1. The antenna will now be tuned above the desired target frequency.

6. The controller has recorded the step at which SWR minimum occurs, however, due to backlash, it cannot simply command the stepper to go to that step. It must command the stepper to move to below the SWR minimum step, then move back upwards to the SWR minimum, thus always approaching from below.

7. Now the stepper has been commanded to the SWR minimum step, and the target frequency has been found. Now the power to the stepper must be released. This is critical! The stepper position must not be disturbed as the power is removed if possible.

8. This is where the TMC stepper comes into play. The TMC stepper has a feature which automatically powers down the stepper after a certain amount of idle time. Powering down the stepper means decreasing the current in the two coils. Also, the rate at which the current is decreased is adjustable. The current needs to be ramped down slowly in order to not move it's position away from minimum SWR. So after a short wait time, the current is ramped towards zero. But there is more! Next the TMC stepper shorts the stepper coils. This is called "passive braking". This is a characteristic of stepper motors, whereby when the coils are shorted the torque required to rotate the rotor is increased. Finally, the driver circuit is deactivated (This is necessary due to RF noise from the driver.). Hopefully, this process should disturb the position of the stepper motor in the least amount which is physically possible.

So why would the stepper motor have a tendency to move when the drivers are powered off? This has to do with the physical construction of a hybrid stepper motor, and the fact that the motor uses permanent magnets. If you pick up a stepper motor, and turn the rotor with your fingers, you can feel a series of detents or notches as the rotor is turned. See the Wikipedia article for "cogging torque":

https://en.wikipedia.org/wiki/Cogging_torque

The 0.9 degree stepper appears to have lower cogging torque which is another advantage, and another good reason to spend the extra money on this component.

### 2.5.2 Backlash

"Backlash" is an old term used to describe the play or "sloppiness" in a tuning mechanism. Almost any type of drive system employing gears or belts is going to exhibit the backlash phenomenon. An analogous phenomenon in electrical circuits is called hysterisis.

Backlash means the positioning is not going to be repeatable, especially when a tuning point is approached from the opposite direction. This is because there is always going to be some slack or stretch in the case of a belt-drive, or imperfectly meshed gears in the case of gear drive.

Thus a way to mitigate backlash is to always approach the tuning point in the same direction.

The physical design of the drive is obviously important. The drive mechanism should be as rigid as possible, belts should be made of a non-stretching material, and in the case of gears they should be precisely machined. Capacitor bearings should be high-quality and have close-fitting bearings. Support structures for the stepper motor and capacitor should be rigid, and fasteners should be torqued sufficiently to prevent movement.

## 2.6 External RF Relay

In order for the controller to tune the antenna to resonance and minimize SWR, it must be directly connected to the antenna. After tuning, the antenna must be switched over to the RF transceiver. An electromagnetic relay is required. A PMOS switch is used in the controller to switch +12 Volts to a coaxial power connector on the rear of the controller. This should be connected to the coil of the RF relay. The coil is de-energized during normal operation of the transceiver. The coil is energized only when the controller is auto-tuning. Thus when controller power is off, the transceiver is connected to the antenna.

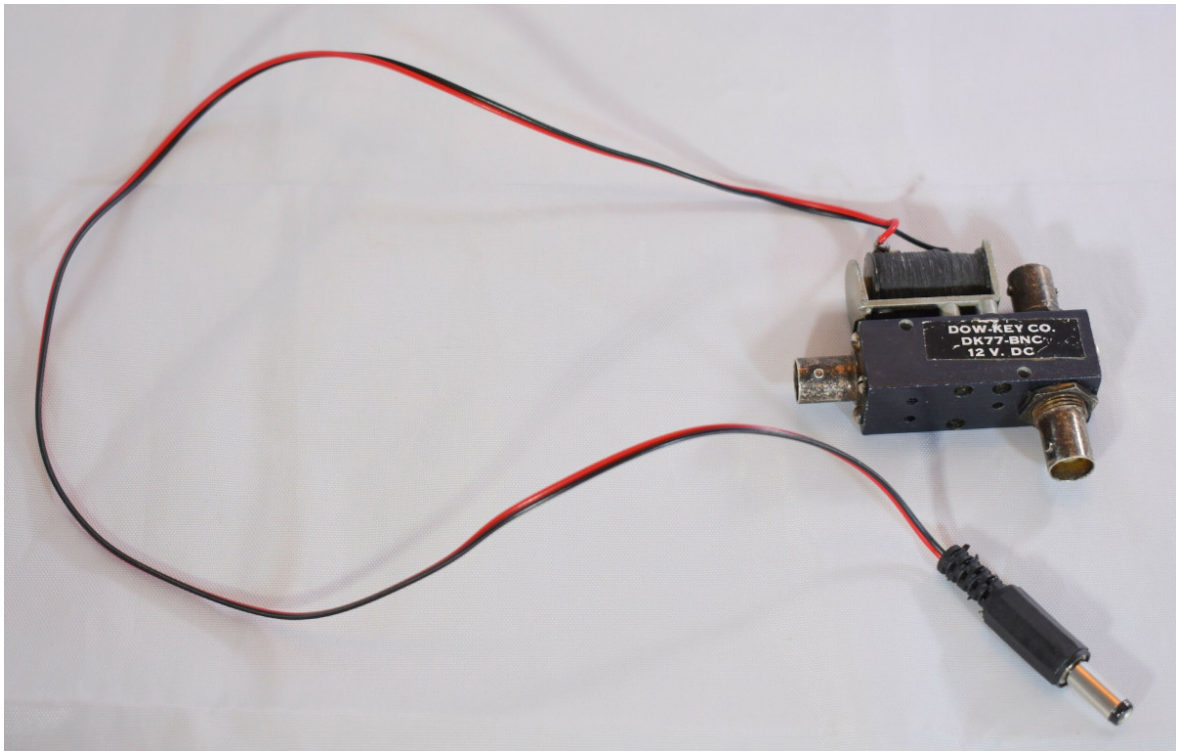


**Figure 2.5: A "Dow Key" type external RF relay with coaxial DC connector**

## 2.7 ILI3941 Display

The ILI3941 is a 320x200 pixel color display module. The module mounts to the front panel, and it is connected to the controller board using a ribbon cable and an 8 pin IDC style connector. An 8-pin header soldered to the ribbon cable plugs into the display module.

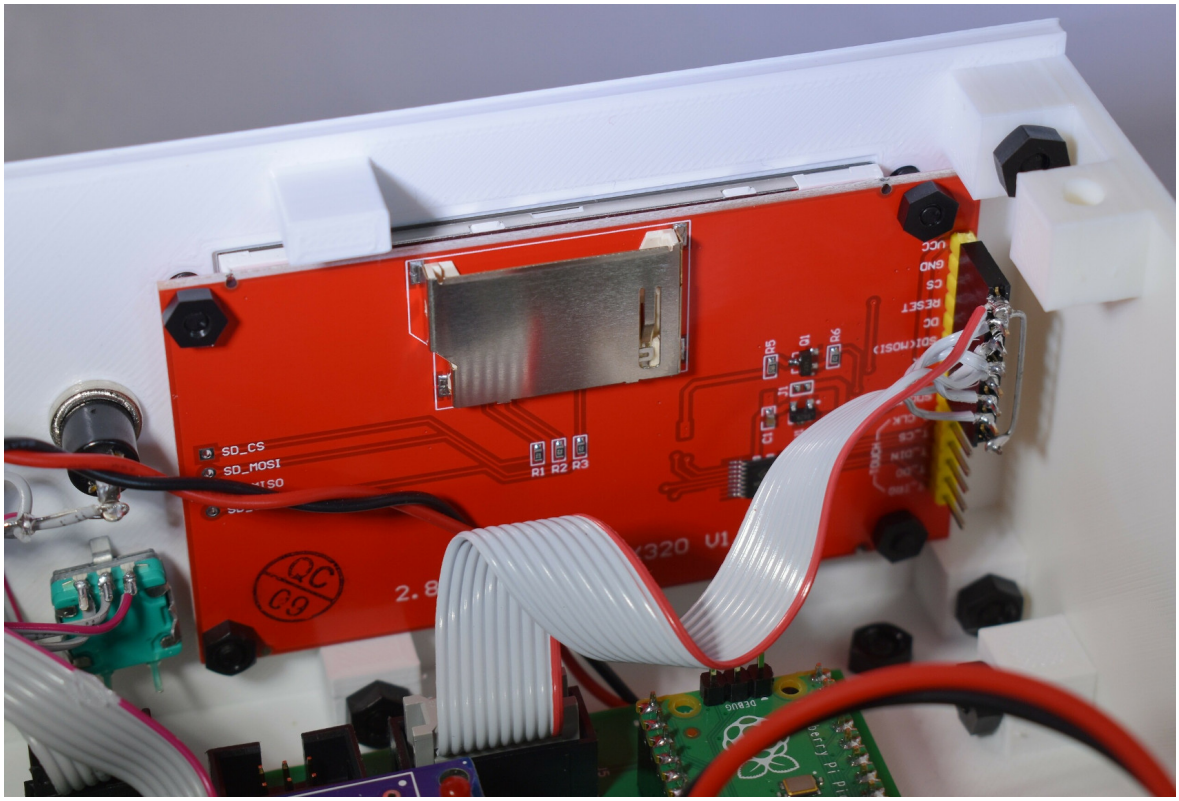


**Figure 2.6: The ILI9341 display mounted to the back side of the front panel.**

## 2.8 Butterfly Capacitor

The "butterfly capacitor" is a critical component! The capacitor determines the frequency range of the antenna. It also determines the power handling capability via the spacing of the capacitor plates.

The capacitor can be ordered from eBay from the TA1LSX store:

https://www.ebay.com/str/ta1lsxstore?_trksid=p2047675.m3561.l2563

There is one downside to the Butterfly versus conventional capacitors. Butterfly capacitors go from minimal to maximal capacitance in only 90 degrees of rotation versus the conventional capacitor's 180 degrees. This increases frequency change per step as the stepper rotates which means less resolution. However, the pulley-belt drive can be changed to compensate by getting a larger capacitor-to-motor pulley ratio. My original design uses a 60:20 ratio. I intend to change to a larger ratio by ordering a larger capacitor pulley and matching belt.

## 2.9 Audio Muting

Please refer to the chapter on the audio muting board.

# Chapter 3

# Controller Schematic

The RF circuitry from the DDS to the output of the SWR bridge was a direct copy of the original project.

The forward and reverse SWR amplifiers were changed significantly. These two-stage operational amplifiers have sufficient gain to amplify the SWR bridge output to a level high enough to utilize close to the full input range of the microcontroller's ADCs. Another function of these amplifiers is to linearize the output of the SWR bridge, which is a pair of diode detectors. A diode feedback circuit is used in the input stage, which linearizes and improves the dynamic range of the SWR measuring system.

The original system used 741 operational amplifiers with a dual $\pm$5volt supply. This requires a negative 5 volt power supply board.

The 741 op-amps are replaced by LM358s, which can operate with a single supply and have a useful input range to ground. Since the SWR detectors output positive voltage only, this is acceptable. These op-amps have low input offset voltage, however, it is enough to disrupt the SWR measurement accuracy. Input offsets are effectively removed in software. The circuit is very similar to the original with the exception of the single supply replacing the dual.

Two PMOS transistors controlled by GPOs are added to reduce current drain during stand-by. The +5 Volt PMOS switch also provides a signal to the audio muting board during auto-tune. A third PMOS outputs +12 Volts to an RF relay to switch the antenna to the controller during auto-tune.

A 1N4004 diode is in shunt with the +12 Volt DC power input. This diode is used with a fuse mounted in the rear panel for reverse-polarity protection.

An A4988 stepper motor driver module is added. This replaces the large external motor driver used in the original project. The A4988 module is connected to an Ethernet connector on the rear of the PCB. Limit switch inputs are also routed to the Ethernet connector.

The Raspberry Pi Pico module was wired according to the directions in the documentation:

https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf

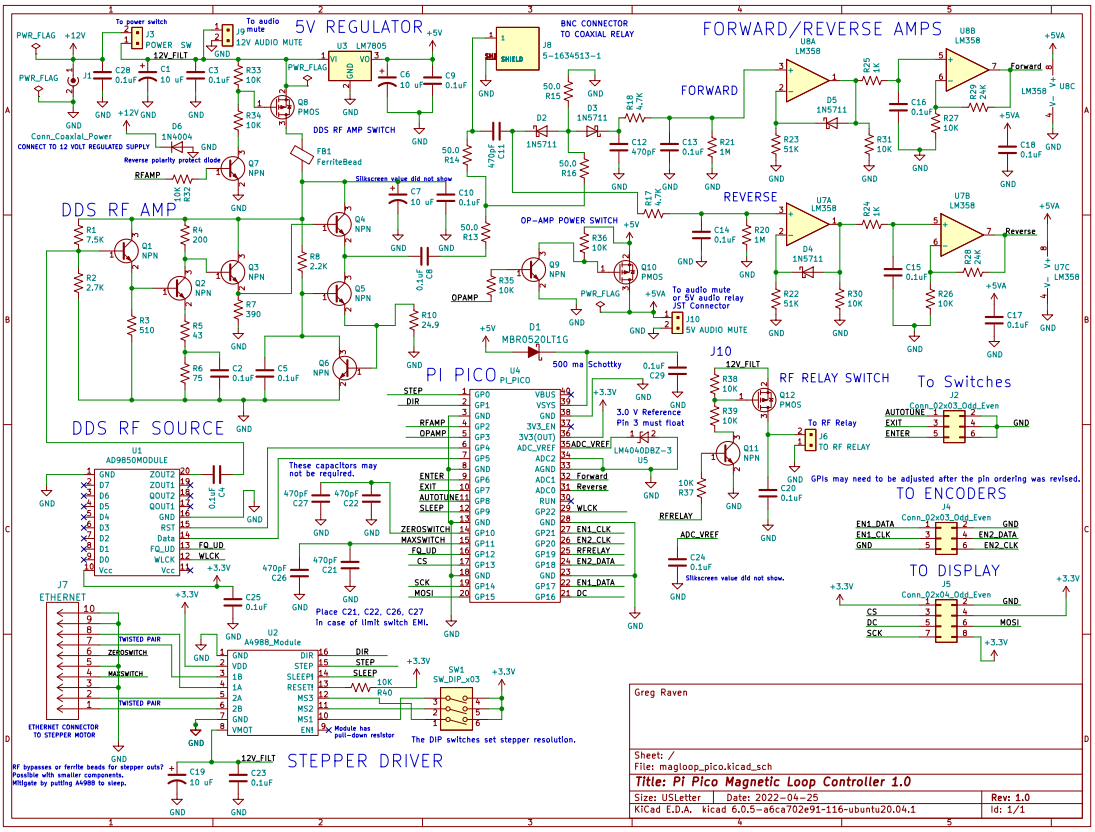


**Figure 3.1: Magnetic Loop Controller Schematic**

# Chapter 4

# Printed Circuit Board

## 4.1 Printed Circuit Board

The first version PCB was derived from the original STM32 project. Since the Pi Pico is approximately the same size, it was dropped in place.

The first Pi Pico board used a through-hole wire terminal for the stepper motor and limit switch connections. These are connections which must be made through a long cable to the remote antenna tuner.

Cheap lawn sprinker cable was used to connect the controller board to the tuner via the terminal strip. This would function fine with a short cable. However, upon trying a 50 foot long cable the controller went nuts!

What happened is that the square pulses output from the stepper driver were coupling to the limit switch inputs. This causes big problems as the controller senses an incoming pulse train on the limit switch inputs, rather than a single step which is what should happen when a limit switch closes.

So I took this problem to the Groups.io Software Controlled Amateur Radio group, which is frequented by builders and users (and the authors of the book) of the original magnetic loop controller. Similar problems were seen, with the best solution to change to a shielded CAT7 Ethernet cable.

After successful testing breadboards of the Ethernet cable “fix” for the limit switch problem, it was decided to make a new PCB with an Ethernet connector mounted to the board.

This version of the board added PMOSFET switches for an antenna relay, the 5 volt supply to the op-amps, and 12 volt supply to the RF amplifier. Unfortunately there was only one spare GPIO, so all of these switches operated simultaneously.

The new external RF relay switch is required to disconnect the controller from the antenna during normal transmit/receive operation after autotuning is complete. The other switches are mainly intended for current drain reduction when the circuits are

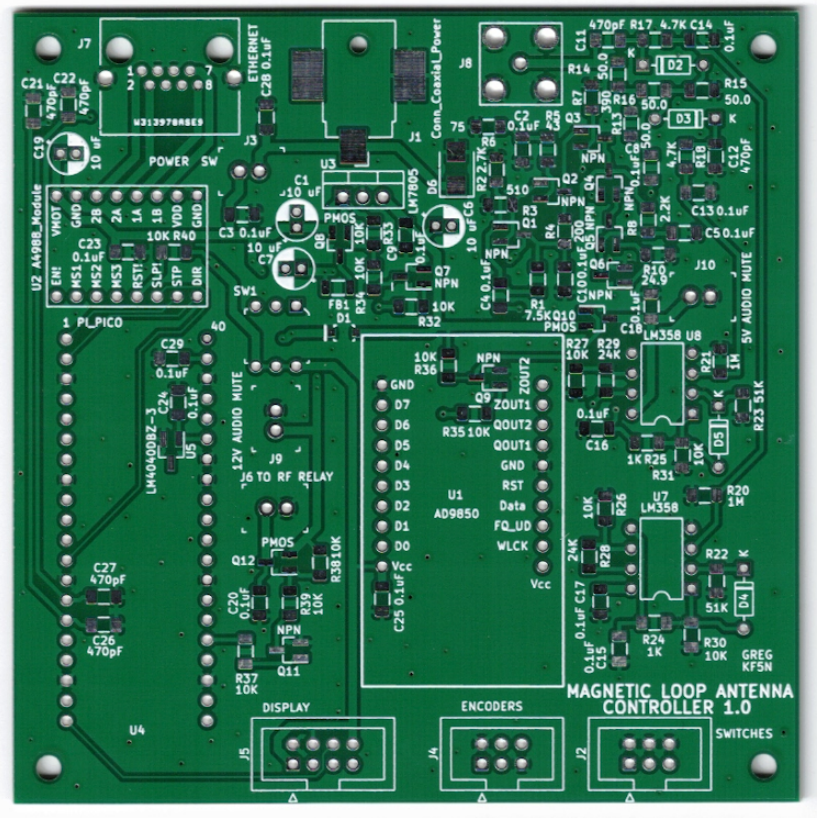


**Figure 4.1: Controller Printed Circuit Board**

not required in normal operation of the transceiver.

The second prototype board was a success, however, the Ethernet connector was located on the side of the board. This means an Ethernet jumper would be required to connect to another Ethernet connector on the rear of the case. Not good!

So it was decided to fundamentally rearrange the main components on the PCB for the third prototype. The Ethernet connector was moved to the rear of the board, adjacent to the power connector. This worked well, as the stepper driver was also moved to the back and this made the routing of 12 volts to the stepper driver much shorter.

Both the Pi Pico and the DDS boards were moved to new locations. When the re-routing was complete, the board routing was simplified, and performance should be improved.

Routing of signals to the IDC connectors for display, encoders, and pushbuttons was done such that making the ribbon cables is as easy as possible.

Gerber and drill hole files for fabrication of the board are located in the magloop_pico_pcb repository.

# Chapter 5

# Firmware

The firmware for the project was developed using a Raspberry Pi 4 single-board computer. Raspberry Pi has done an outstanding job providing a script which installs everything required to develop code for the Pi Pico, including Visual Studio Code, which is an advanced text editor. Although VS Code is a text editor, the various enhancements make it into an IDE.

Debugging is done via Serial Wire Debugger interface (SWD). The SWD is connected to the Pi Pico via three jumpers from the Raspberry Pi 4 GPIO connector to a three pin header on the Pi Pico. No special "dongle" is required!

Please refer to the Pi Pico documentation for all of the information you will need to edit, debug, and load code. The documentation is thorough and high quality.

The supported debugger is the main reason this project was developed using the Pi Pico SDK. Debugging is only recently available in the Arduino IDE 2.0. However, an initial look at the Arduino debugger indicated it was not mature compared to the Pi Pico SDK. In the future, the Arduino IDE may be revisited if the Pi Pico support includes a solid debugging feature.

Please note the script "upload" is included to allow a quick command-line load of the elf file to the Pi Pico.

## 5.1 Organization of the Code

The code is organized differently than the original project. The original project is an Arduino style project, with the use of C++ classes (libraries) and multiple files with associated C-style functions in each. It is the typical "setup" and "loop" structure of Arduino.

This new project based on the Pi Pico is more strongly C++. Rather than files with numerous C-style functions, these functions were gathered up and put into C++ classes. The main function instantiates these classes in the form of several objects. I

found this "C++ embedded programming" helped me to create easier to understand code. The fundamental programming principle is called "separation of concerns":

https://en.wikipedia.org/wiki/Separation_of_concerns

"Inheritance" was used where it naturally fell into place. For example, the "StepperManagement" functions were organized into a C++ class, and this class inherits from the AccelStepper class library.

The "state machine" pattern was used in several places including the GUI menu in the main function.

The CMake build system is used to compile, link, and create various associated files. This is one disadvantage of using the Pi Pico SDK. You don't have to deal with this in Arduino. After a bit of struggle with CMake, I got it all working together smoothly.

## 5.2 Data Class

There is a new "Data" class which stores most of the constants used to configure the controller. This class is intended to replace all or most of the #define statements which appear in the original design. A data object is instantiated in the main program at start-up. The data object is referenced by the other objects and is used to access the relevant constants.

Future versions of the firmware will include more constants in the Data object.

## 5.3 EEPROM Class

The EEPROM class performs the same function as the original project. It provides non-volatile storage for the band limit stepper motor positions after initial calibration. Like the original project, it uses FLASH memory to mimic EEPROM functionality. The Pi Pico SDK FLASH functions were used to create this class.

## 5.4 ILI9341 Driver

As mentioned before, the library support for the Pi Pico is in the early stages. I needed a library for driving the ILI9341 TFT display. There is an Arduino driver for the Blue Pill, and I set off on a mission to hack it to work with the Pi Pico.

The problem with hacking an Arduino library is that the code is designed to work with multiple architectures. The code is littered with #if statements which select blocks of code depending on which architecture is in use. This was painful to work with, on top of the fact that it is a complex driver.

After hours and hours of hacking, I got the ILI9341 working with the Pi Pico. If a driver specifically intended for the Pi Pico appears, I will certainly consider replacing the monstrous hack I created. Having stated that, it has been working reliably and

consistently with all of the prototypes. Having the display is also great for debugging because you can print anything you want to the display. That combined with the debugger made the rest of the code development quite pleasant.

The original Arduino library:

https://github.com/adafruit/Adafruit_ILI9341

## 5.5 Stepper Motor Library

The original "AccelStepper" library was modified to work with the Pi Pico. The modifications for this library were minimal.

The original Arduino library:

https://www.airspayce.com/mikem/arduino/AccelStepper/

## 5.6 Rotary Encoder Library

The original "Rotary" library was modified to work with the Pi Pico. The modifications for this library were minimal.

The original Arduino library:

https://github.com/buxtronix/arduino/tree/master/libraries/Rotary

## 5.7 De-bouncing

De-bouncing functions are included in the Button class.

The original project used simple time-delays for debouncing the buttons. This project has state-machine based de-bounce. This was challenging to implement, as it also had to include de-bounce when moving up and down the menu levels. An auxiliary function was required along with careful placement of the debounce functions in the code.

## 5.8 Limit Switch Protection

The low and high limit switches are connected to GPIs via the CAT7 cable. These GPI states are continuously monitored during stepper motor movements. If either switch closes, the stepper motor will be stopped. This is to protect the limit switches from damage due to over-rotation of the stepper motor. This is most likely to happen during "bring up" of the controller as connections could be accidentally reversed. It is also possible to hit the high limit if the controller is commanded to go to a frequency beyond the limit of the upper limit of antenna resonance.

In case of an upper limit event, the user is notified of an ”Upper Limit Hit” via the display. Note that the primary function of the low limit switch is for calibration of the stepper motor. Its secondary function is to protect itself from destruction.

# Chapter 6

# Loop Antenna Construction

The original project uses a ”double-double” loop structure formed from copper plumbing pipe with 3D printed supports and PVC plumbing pipe.

That structure was a bit too much for my limited ship capabilities, so I opted for the common single-loop design. I had a chunk of semi-rigid coaxial cable lying around. So that became the “loop” part of the antenna. It is possible to use copper or aluminum tubing, and I am especially interested in building a version which uses lightweight aluminum refrigeration tubing.

The 3D printed parts for the double-double inspired the parts used in the single-loop antenna. The mechanics of the stepper motor drive for the butterfly capacitor are also similar to the double-double.

The stepper motor, bracket, limit switches, and pulley parts were sourced from Amazon. With regards to the pulley, you must make sure to order the pulleys with the same mounting hole diameter as the motor and capacitor shafts. A list of links to these parts is provided in the BOM file.

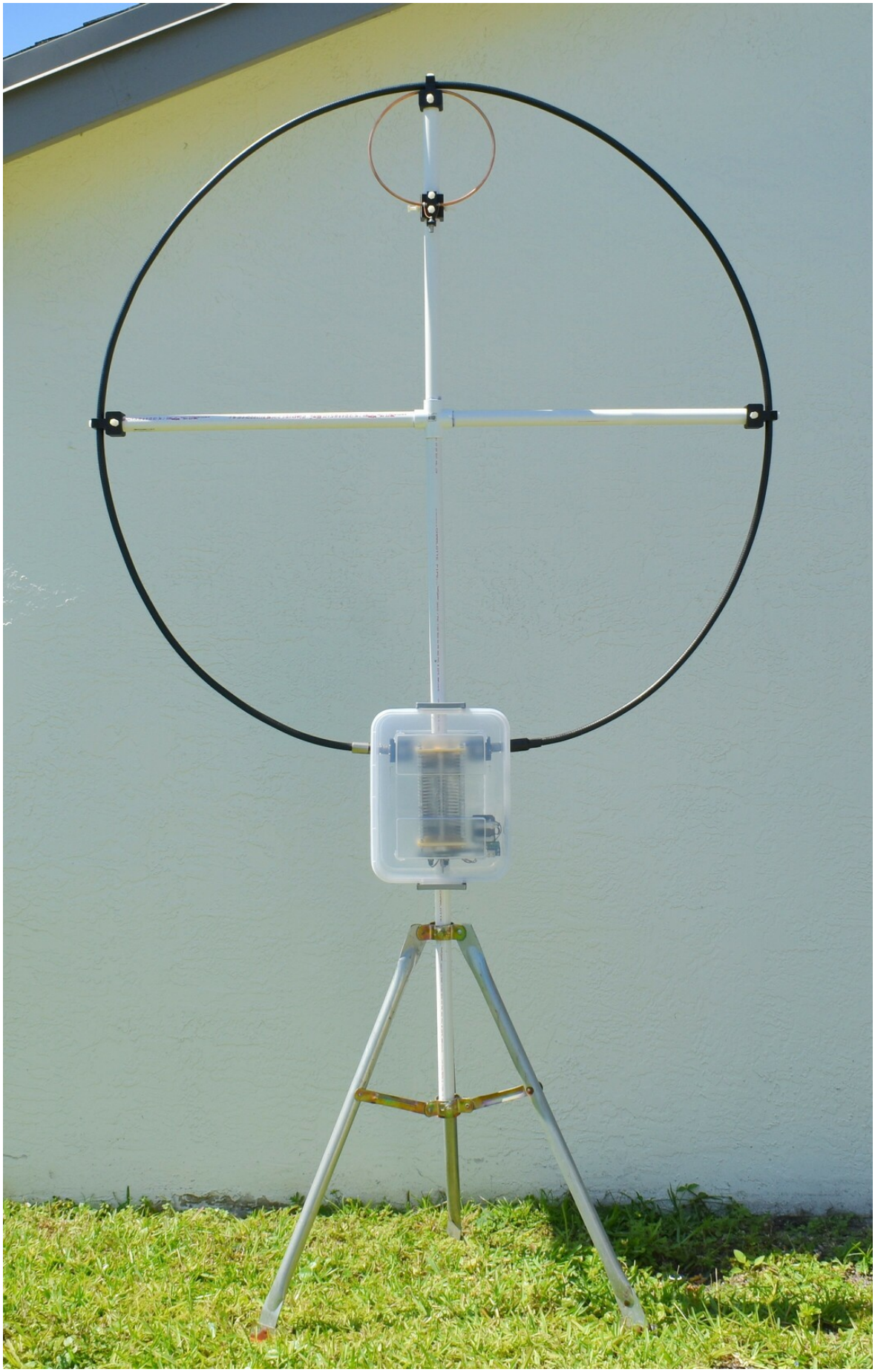

**Figure 6.1: The complete loop antenna.**

## 6.1 3D Printed Parts

I am going to warn you that the 3D printed parts for the antenna are to be considered ”prototypes”. You will probably need to change them to fit your material and requirements. But that is the beauty of 3D printing! I tried to keep the structure as simple as possible.

The antenna structure is supported by one-inch schedule 40 PVC pipe. 3D printed “clamps” are used to hold the loop in place. Another set of clamps holds the “tuner” assembly in place at the bottom of the loop. The antenna tuner components are bolted to a rectangular piece of Plexiglas. Nylon bolts are used throughout to keep the weight down.

The original 3D parts were printed using PLA filament. This material does not withstand high temperatures! Parts manufactured from PLA will distort in warm climates with Sun exposure. I have switched materials to ABS and also using the color white. Although ABS is more difficult to print, the ABS parts have withstood a very warm climate without any noticeable distortion.

### 6.1.1 Left and Right Loop Supports

These are relatively simple parts which clamp the loop to the ends of the horizontal PVC pipes. Nylon bolts are used for both the loop clamp and also to secure the support to the PVC pipe. It is also possible to use glue to secure the supports to the PVC.

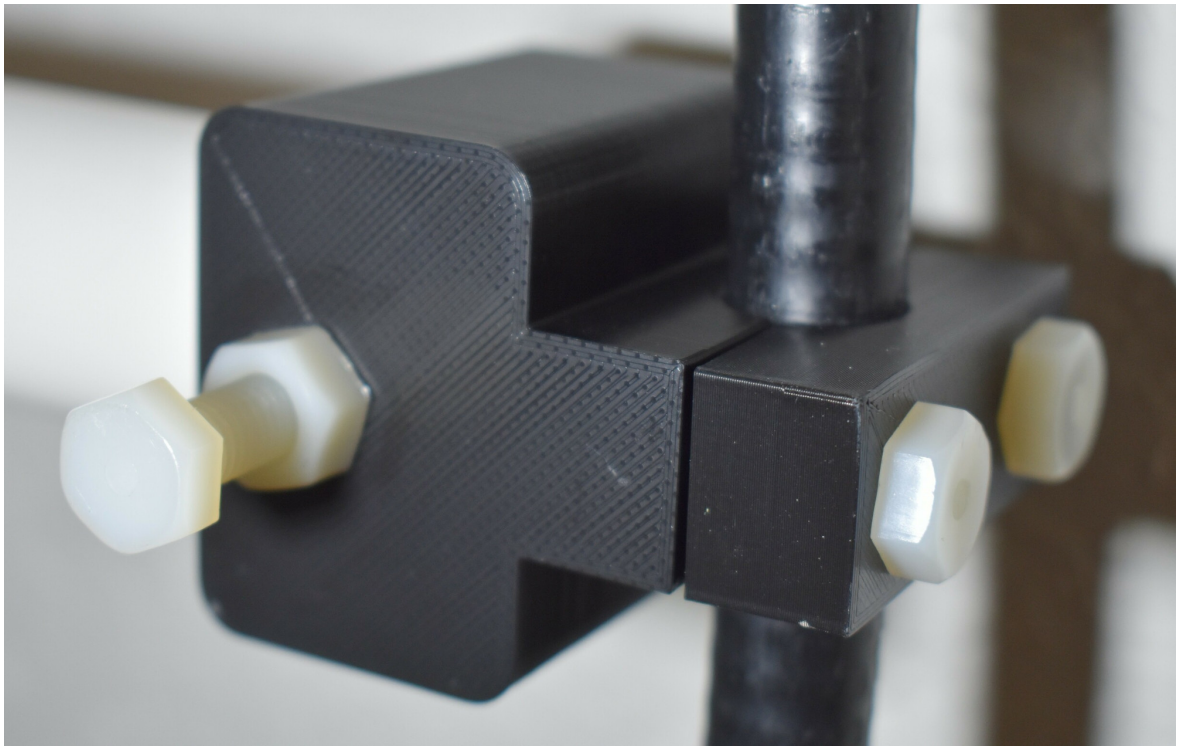

**Figure 6.2: Side support for the loop conductor. The support is the same for left and right.**

### 6.1.2 Top Support

The top support is similar to the left/right supports, with the additional feature of a slot for the coupling loop.

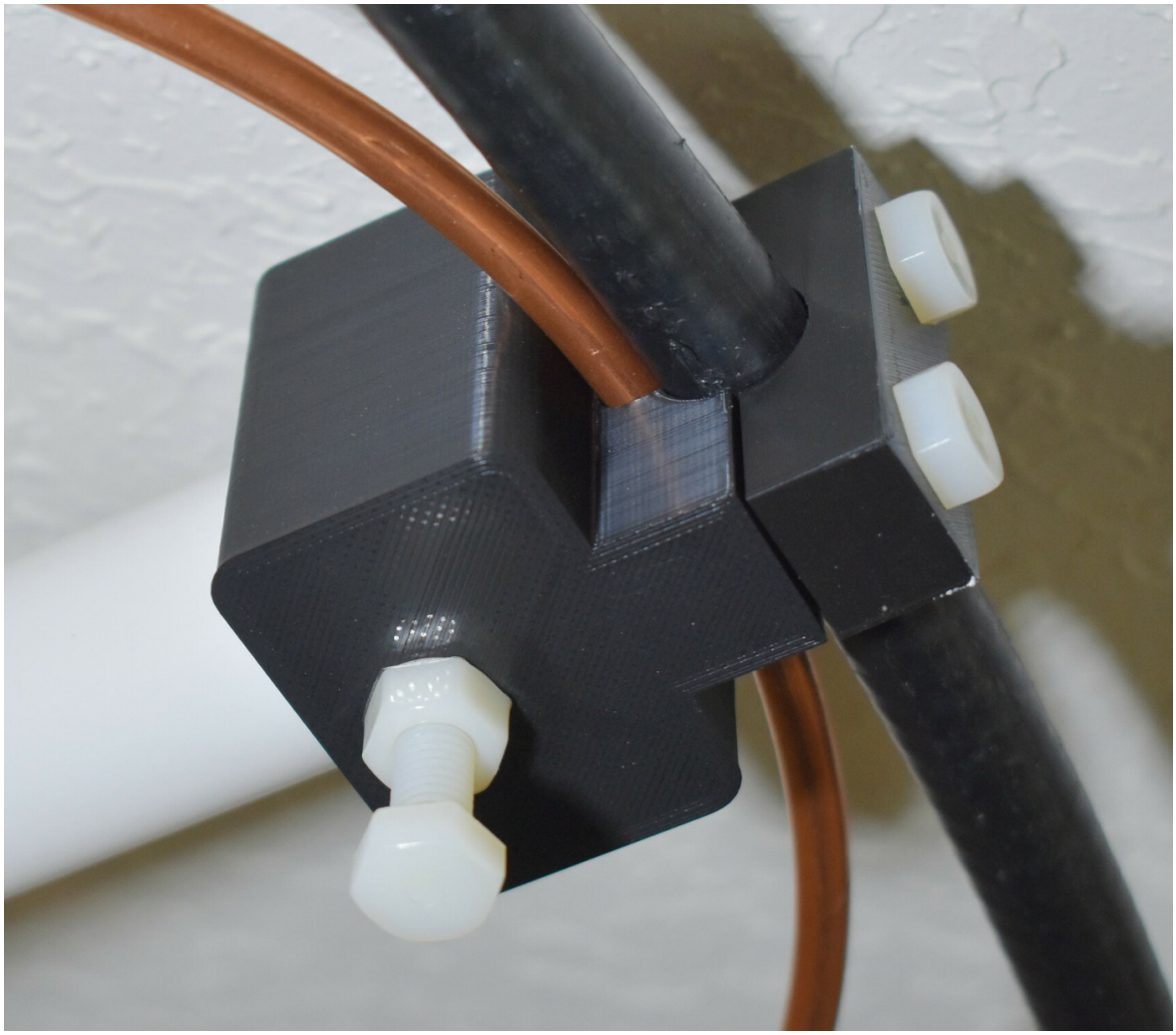

**Figure 6.3: Top support for the loop conductor. Note the slot for the coupling loop.**

### 6.1.3 N Connector Brackets

The cable I used for the loop has male N connectors. I designed simple brackets for chassis-mount style female connectors. Note that the center conductor is not used and is floating. Soldered terminals and solid copper wire connect the outer “shield” of the coax to the butterfly capacitor.

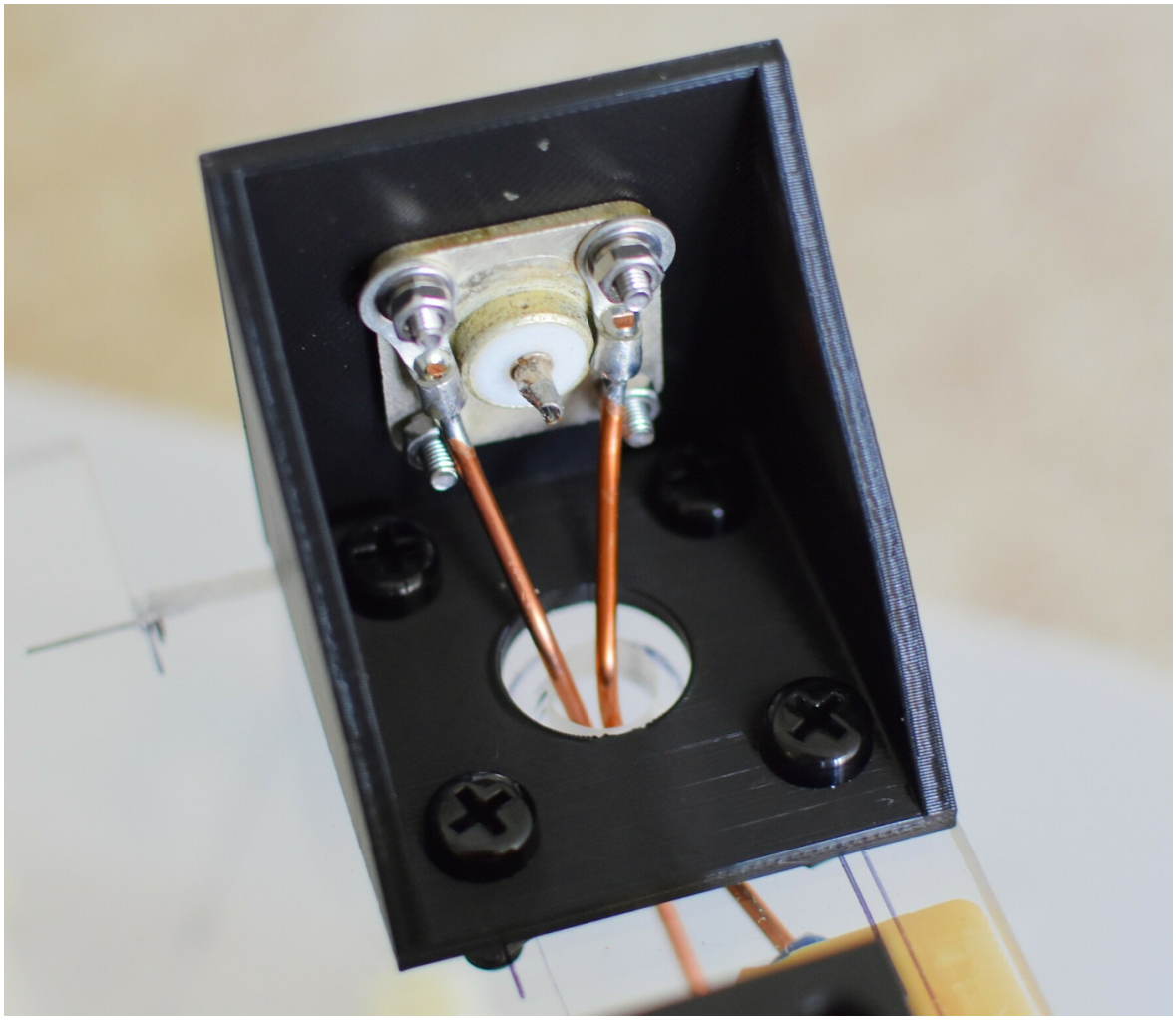

**Figure 6.4: N connector bracket.**

### 6.1.4 Coupling Loop and BNC Connector Bracket

The coupling loop is formed from quarter-inch copper plumbing pipe. The diameter was experimentally derived by making loops of three different diameters. A "NanoVNA" was used to check the SWR over the desired frequency range.

The coupling loop diameter is 26 centimeters. It is possible it could be further optimized, but the improvement if any would be minimal.

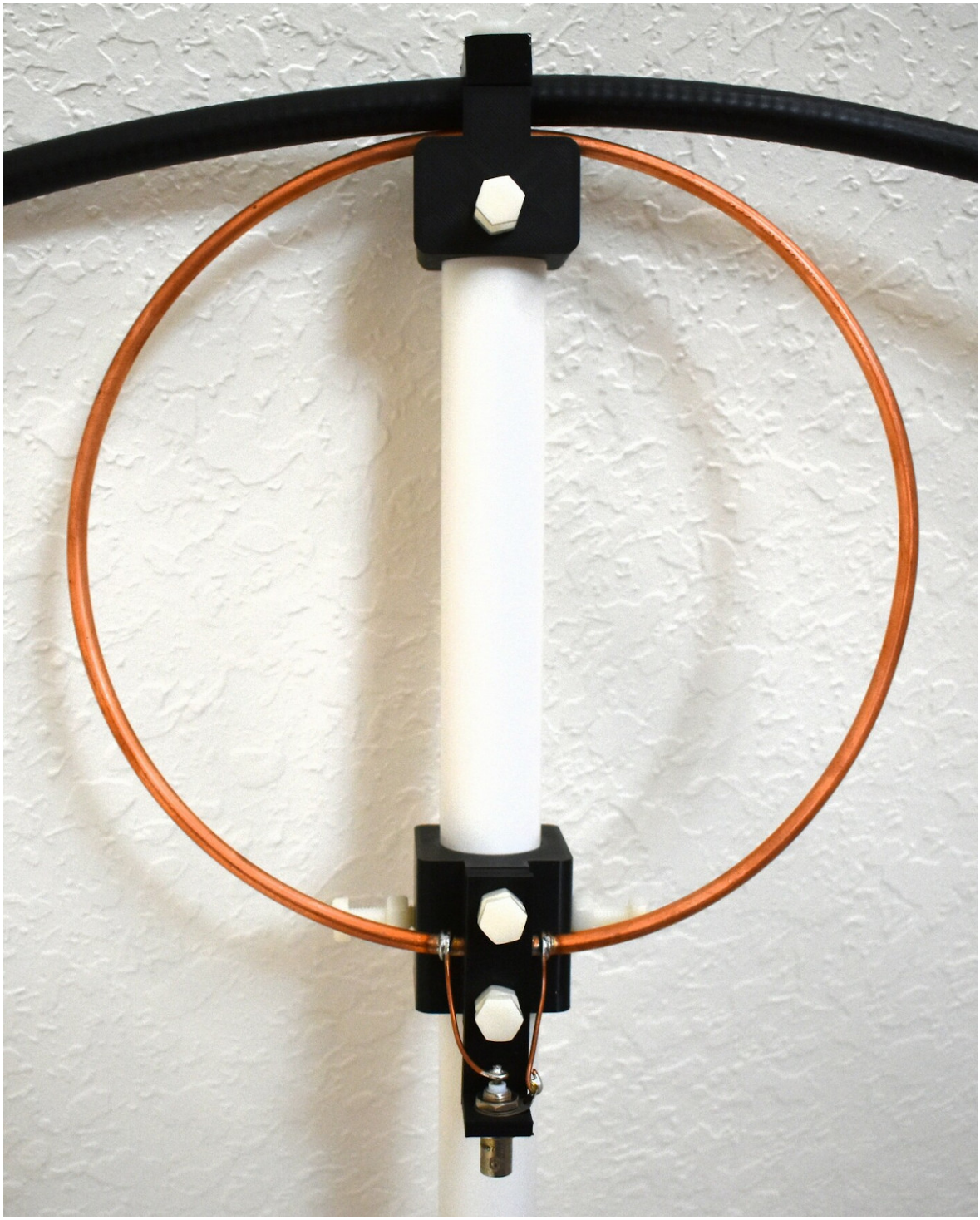

**Figure 6.5: The coupling loop assembly.**

The coupling bracket slides over the one-inch PVC pipe and is secured in place with nylon bolts. A 3D printed bracket holds the BNC chassis-mount female connector, and it also bolts to the bracket to secure the ends of the coupling loop. Solid copper wire is used to connect the coupling loop to the BNC connector.

The lower bracket for the coupling loop is composed of three parts:

1. The main bracket piece which slides over the PVC pipe.

2. A copper tube clamping piece which is sandwiched between parts 1 and 3.

3. The angle bracket for the BNC connector

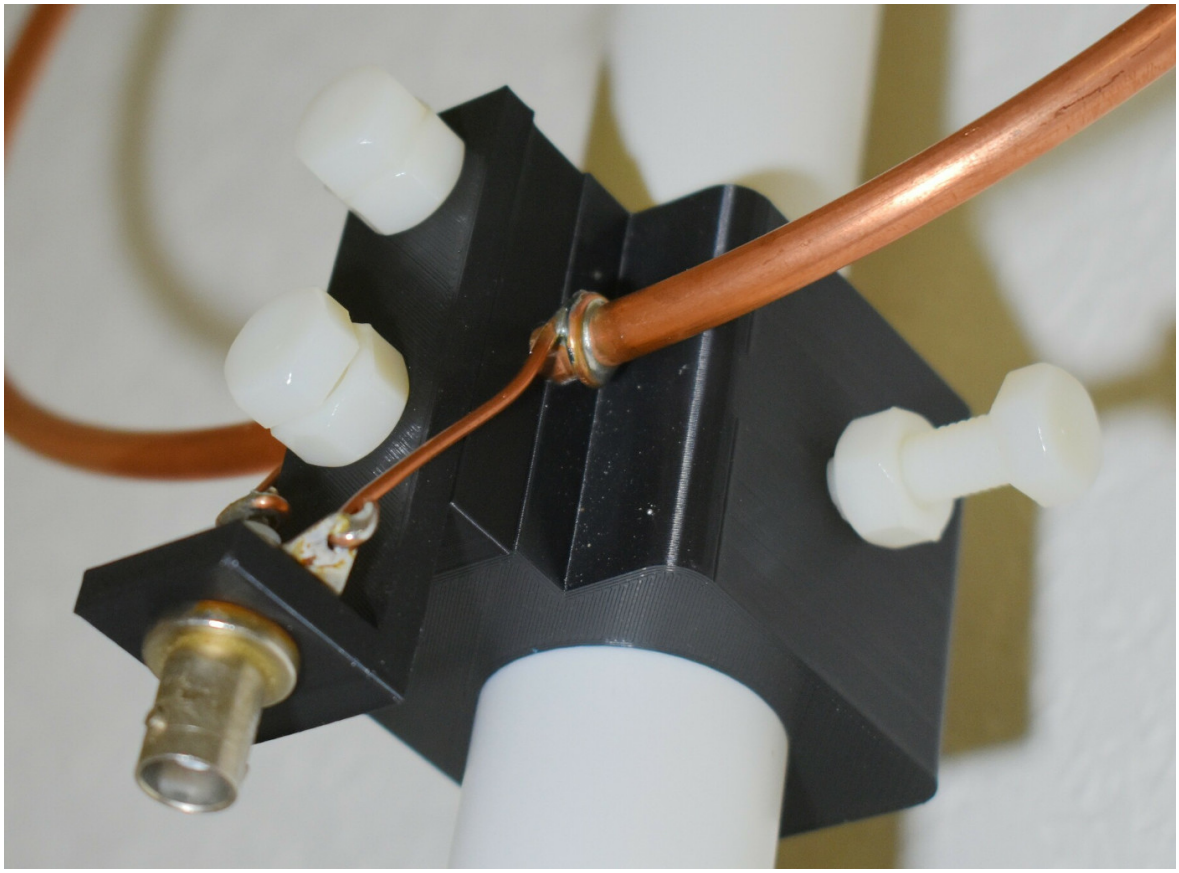

**Figure 6.6: The lower coupling loop bracket with BNC connector bracket "sandwich" assembly.**

### 6.1.5 Tuner Assembly Clamps

Four clamp pieces are required to secure the tuner assembly to the vertical PVC pipe. These clamps grip the pipe and hold the tuner assembly via holes drilled in the Plexiglas tuner assembly base. Thus there are a total of six bolts required for each set of clamps, for a total of twelve bolts. The photos below will clarify the construction of this assembly.

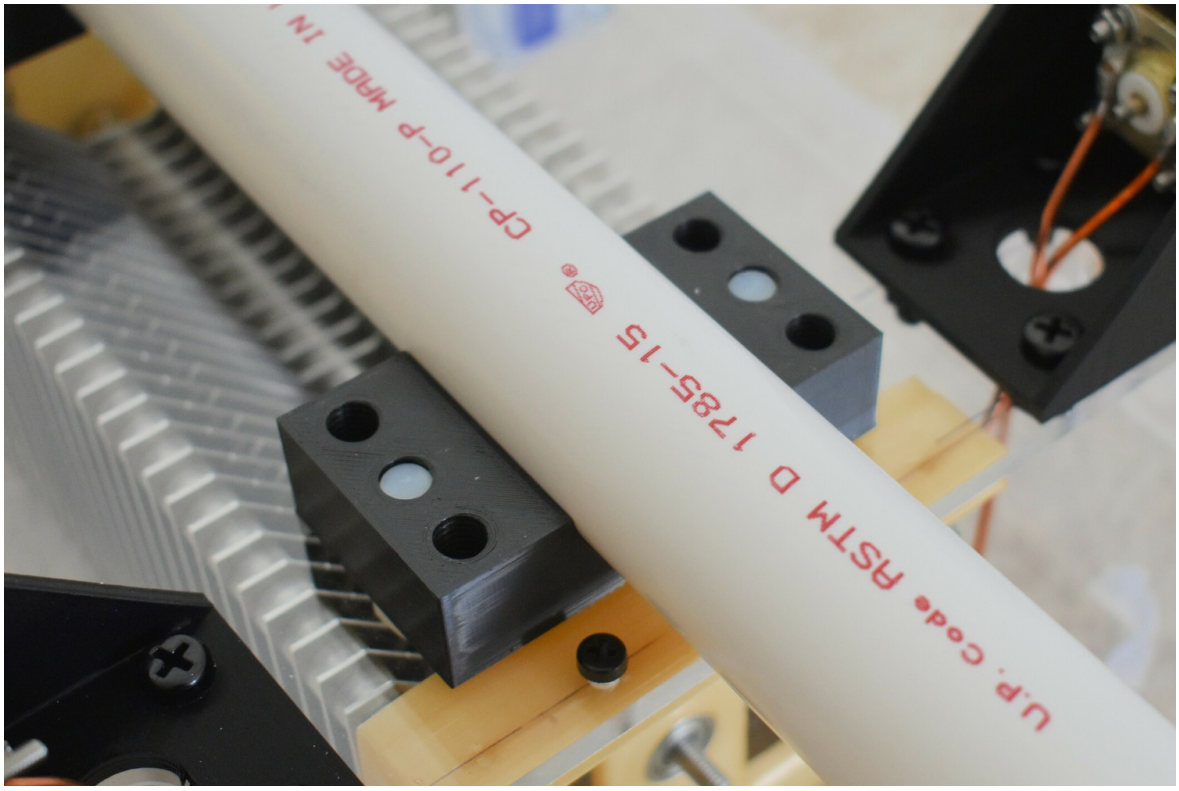


**Figure 6.7: One of two pipe brackets before adding the clamp piece.**

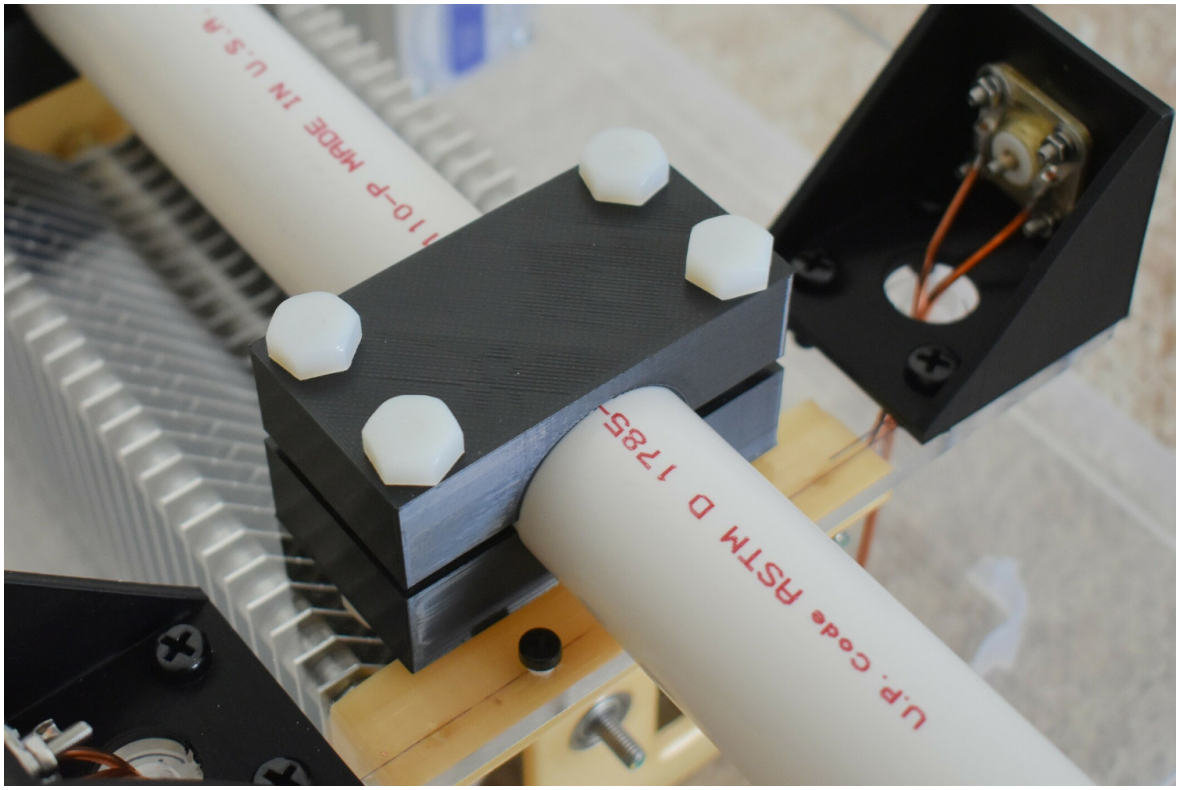


**Figure 6.8: One of two pipe brackets after adding the clamp piece.**

### 6.1.6 Limit Switch Bracket

The limit switches are small modules ordered from Amazon. The modules include the plug and wiring. The switches must be held at the proper height such that the switches close at positions which allow calibration of the capacitor and protection of the switches (especially the high limit switch). A 3D printed bracket was designed to accomplish this. The height may require adjustment depending on the exact mechanics used. However, only the "zero" limit is critical. This is the switch which closes on the high-capacitance end of the rotation. This switch provides the primary calibration which is used by the controller to steer and auto-tune the resonant frequency of the antenna.

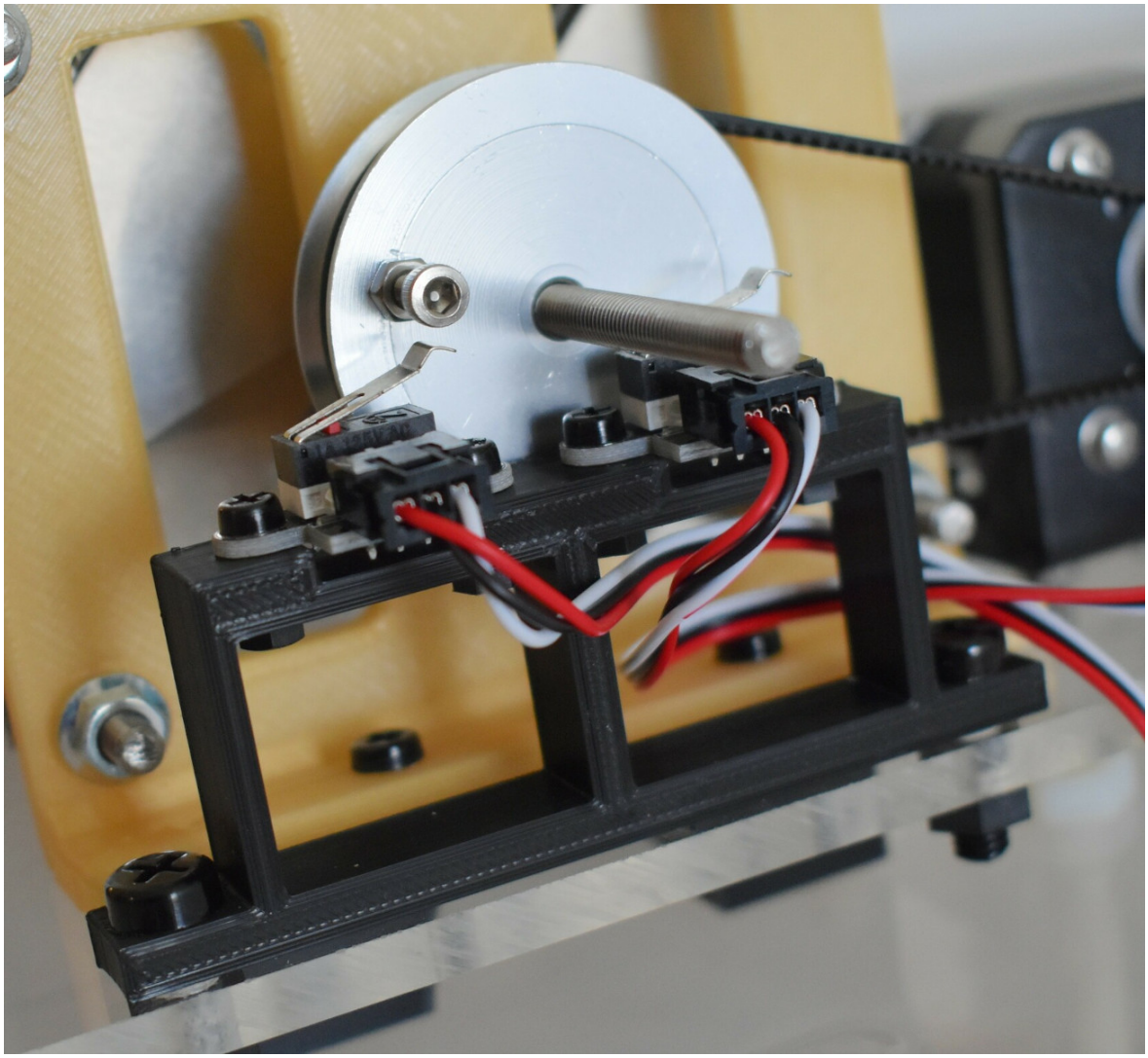

**Figure 6.9: The limit switch bracket.**

### 6.1.7 Ethernet Connector Bracket

The stepper motor and limit switch connections use a “breakout” board ordered from Amazon. This board is a simple Ethernet to terminal block. A 3D printed bracket and four bolts attaches the board to the Plexiglas tuner base. The four stepper motor wires and four limit switch wires connect to the terminal block for a total of eight wires.

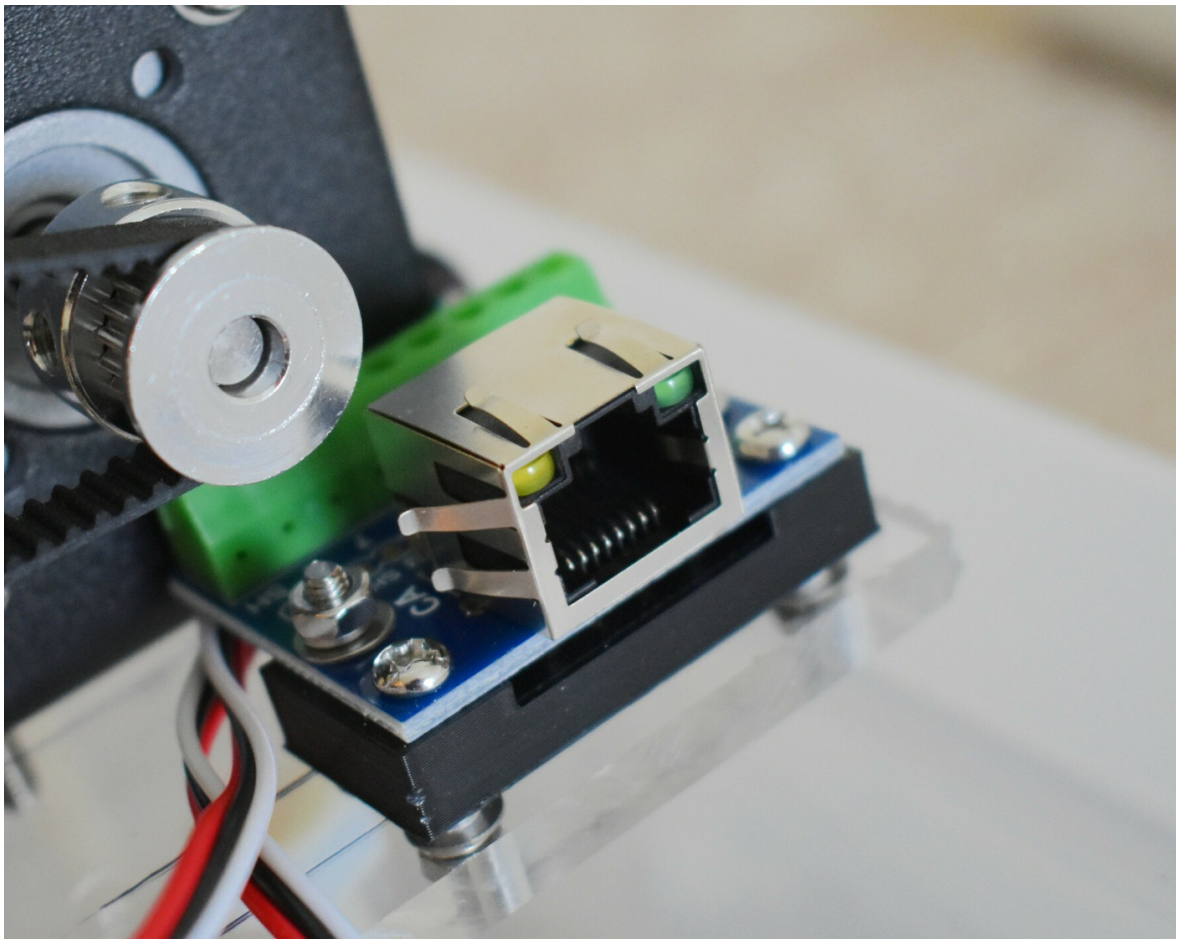

**Figure 6.10: The Ethernet connector bracket.**

### 6.1.8 Stepper Motor Bracket

The motor mounting bracket was purchased from Amazon along with the stepper motor. This bracket has slotted mounting holes, which allow for adjustment of the motor location, which is a good feature.

The motor is a NEMA 17 1.5 Amp rated motor. The torque required to turn the butterfly capacitor is low, so it is possible for a smaller motor to work, but this has not been tested.

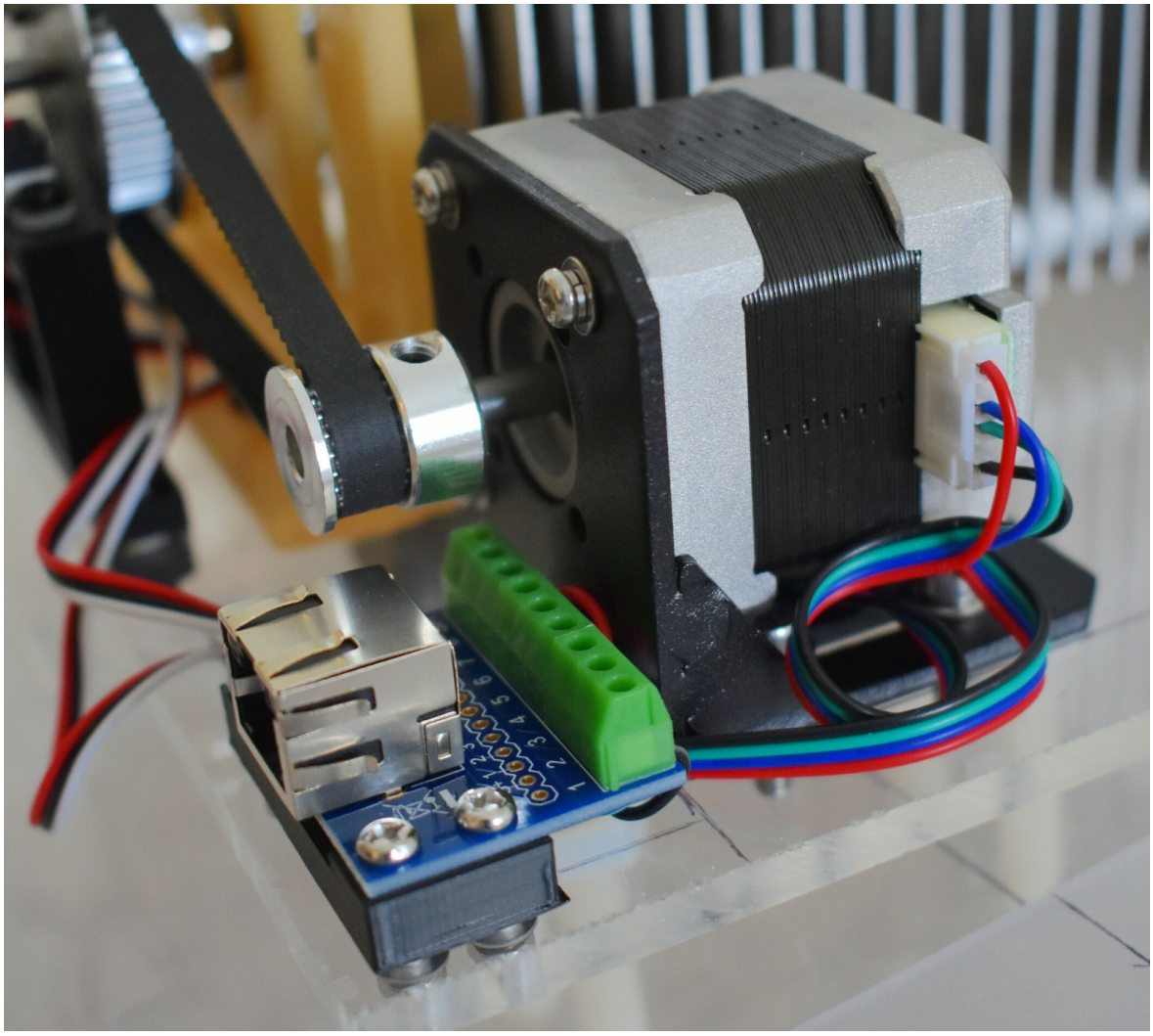

**Figure 6.11: The stepper motor and bracket.**

### 6.1.9 Pulleys and Belt Drive

The pulleys and fiber belt are 3D printer parts ordered from Amazon. Be sure to get the correct pulley hole diameter to match the shafts of the motor and capacitor. Belts are available in various lengths. You will have to determine an approximate belt length, and then adjust the position of the motor to achieve sufficient tension. Don't drill the mounting holes for the motor until you get all of the components. Perhaps a 3D printed bracket will be designed which allows both motor position and belt tension to be adjusted. For now, the stock bracket has been used successfully.

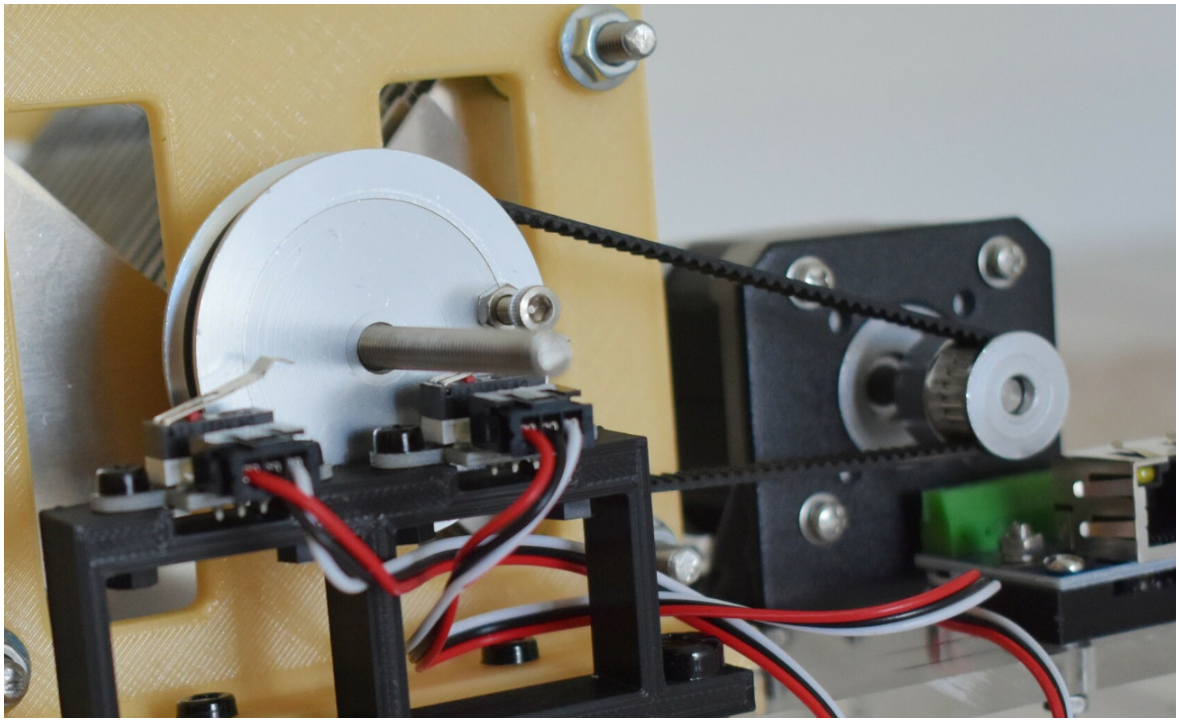

**Figure 6.12: The pulley and belt.**

### 6.1.10 Plexiglas Tuner Assembly Base

A rectangular piece of Plexiglas was ordered from Amazon. This Plexiglas should be reasonably thick for mechanical stability. A link to the part used is provided in the BOM.

A lot of hole drilling is required to mount the various pieces of the tuner assembly.

It is best to get all of the parts ready, including the 3D printed parts, before proceeding with preparation of the base. Otherwise it is easy to make a mistake. Arrange all of the parts on the base, and make sure all clearances make sense, and that there is room for all of the components. Make very careful measurements, and mark out the hole locations on the Plexiglas.

Measure twice, drill and cut once!

Drilling Plexiglas requires very sharp drill bits. Go slowly or the Plexiglas will melt!

I recommend a little slop in the motor mount holes. This will allow some adjustment of the motor position in order to tighten the belt. The belt needs to be snug, but not overly tight.

I used a combination of nylon and steel hardware for mounting the components to the base. Amazon has good hardware "kits" of both nylon and steel hardware. I used exclusively metric hardware, and the tuner assembly is working well.

## 6.2 Comments on Threads in 3D Printed Parts

Getting the 3D printed threads to work well required experimentation. I recommend designing a simple rectangular block with a threaded hole to test the quality of the threads. I used Freecad 0.20 which includes a threading tool. This tool has a "clearance parameter" which can be adjusted to adjust the tightness of the threads. This will have to be adjusted to get the threads "just right".

It is better to use larger and coarse threaded bolts. I used M8-1.25 nylon bolts. In future designs, larger bolts may be used. Links to the parts ordered from Amazon are included in the BOM spreadsheet.

## 6.3 Butterfly Capacitor

The butterfly capacitor is assembled from a kit which can be ordered from an eBay store. The URL to the store is included in the BOM spreadsheet.

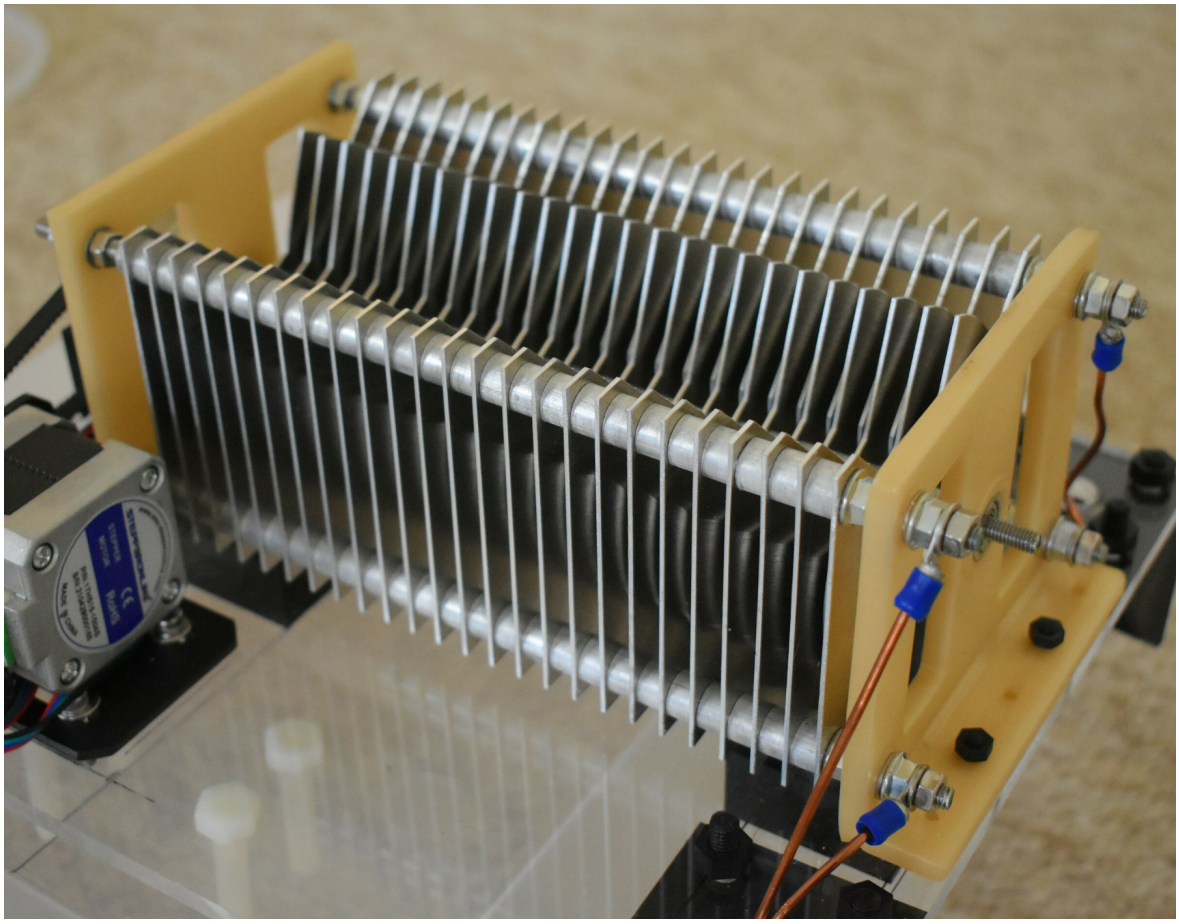

**Figure 6.13: The "butterfly capacitor".**

The butterfly capacitor can be ordered with more or less plates in order to adjust the capacitance range. This is critical in order to cover the desired amateur radio bands.

Online calculators were used to determine the number of plates required for operation from 40 to 20 meters.

The calculators will get close, but you should include a bit of margin, especially for the lowest frequency. In my case, the maximum number of plates was ordered which is 25. I used 1 millimeter thick plates with 6 millimeter spacers. This capacitor provides full coverage of 7.0 to 14.35 MHz amateur radio bands.

The capacitor endplates are manufactured from a material which does not withstand high temperatures. The eBay seller provided the stl file, and endplates were printed using a white ABS filament. The ABS endplates have withstood a hot climate without distortion. ABS filament is strongly recommended for all 3D printed parts of the antenna.

## 6.4 Capacitor Protective Enclosure

An inexpensive plastic storage box was fitted onto the tuner assembly for some protection from moisture, insects, and animals. It is by no means waterproof, so don't leave it outside in severe weather!

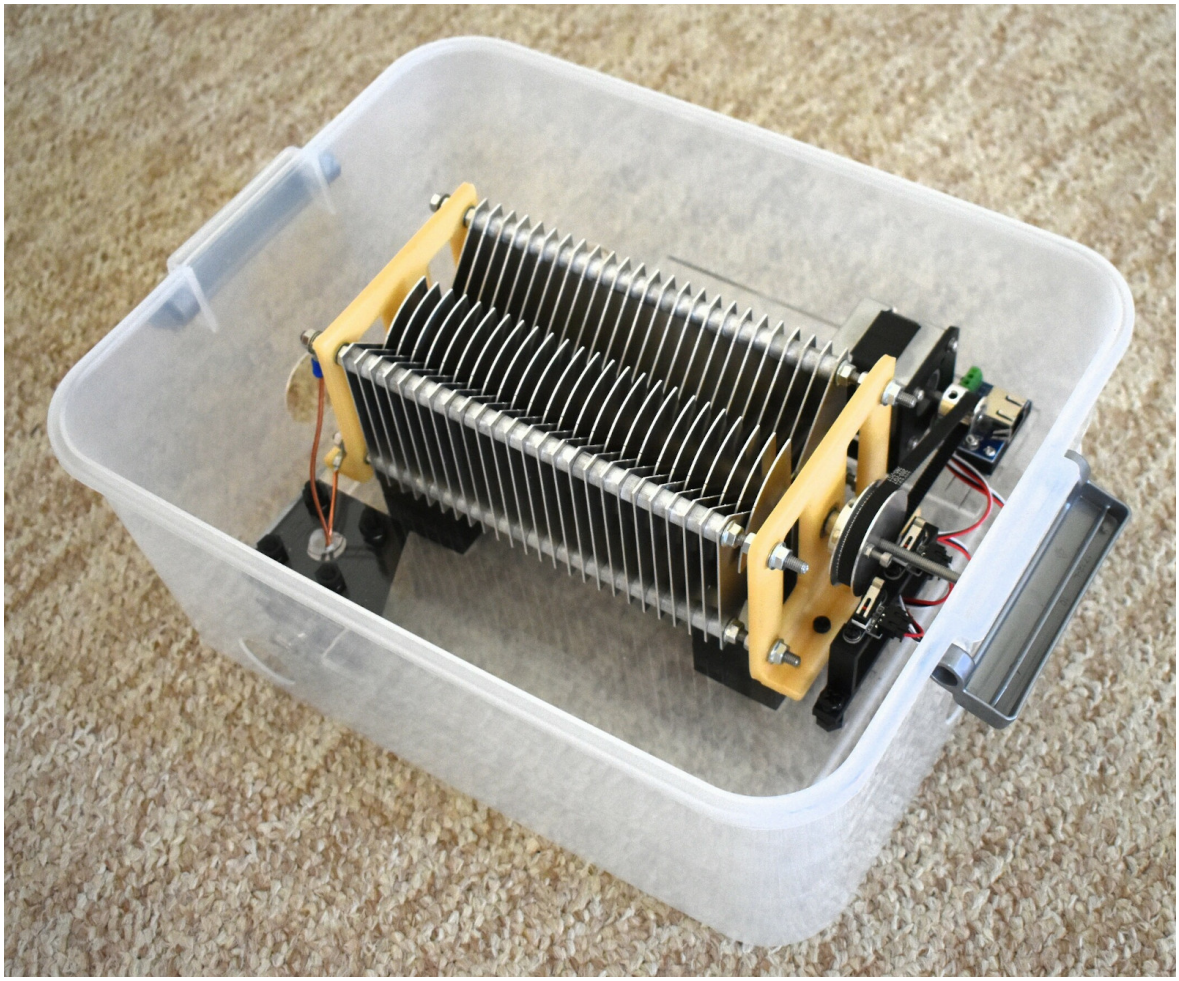

**Figure 6.14: The tuner's plastic enclosure without top.**


**Figure 6.15: The tuner's plastic enclosure with top.**

# Chapter 7

# Using Ethernet Cable for Stepper Motor Connection

The controller works by rotating a stepper motor connected to a tuning capacitor. As the antenna approaches resonance, there is a sharp “dip” in the SWR, which is continually monitored by the controller via the antenna coaxial cable.

Thus it is required to drive a stepper motor from the controller to whatever remote location the antenna is located.

The challenge with this is that the stepper motor drive is a series of sharp pulses! This means the cable connecting the stepper motor to the controller must not have excessive resistance, parasitic inductance, and capacitance. Fortunately, the pulses are very low frequency so it does not have to be perfect. I was skeptical at first that such remote operation of a stepper would work reliably, but this scheme has proven robust enough to rotate the variable capacitor. The fact that this is a low-torque application of a stepper motor helps make it possible.

In addition to the four wires for the stepper motor, there must be wires for two “limit switches”. There is a “low” switch and a “high” switch. Both of these switches are used to protect from over-rotating the capacitor. More importantly, the low switch is important for calibration of the system. The controller will automatically “zero” out the stepper when the controller is powered on. The zero step point coincides with the fully-meshed, maximum capacitance setting of the variable capacitor. All further operations of the controller are dependent on the zero step point being accurately established at power on.

A major problem can occur as the limit switch wires are bundled in the same cable as the stepper wires. Stray capacitance in the wires causes the sharp pulses to be coupled to the low and high limit switch wires, which are connected to the Pi Pico’s GPIs. This will cause “falsing” of the limit switch detect, and the result is erratic operation of the controller.

At first, I was using cheap utility wire from the local home improvement store. The pulses on the limit switch wires were extreme, so I decided to consult with other users

of the controller who frequent the Software Controller Ham Radio group (groups.io). Some were using capacitors on the limit switches, and others were using Ethernet cables. The Ethernet cable idea is suggested in the book, but somehow I had failed to make note of that suggestion. Cheap utility cable seemed an easier and cheaper route. That was proven wrong!

After looking at the various forms of Ethernet cable, I decided to try Cat 7. This has shielded twisted pairs, which will minimize cross-talk. There is also a shield around the entire cable assembly which can be grounded.

There are a total of four twisted pairs. This is perfect, as the stepper motor signals can be connected to two of the twisted pairs, and the limit switches can be connected to the other two.

Note that the shield around the entire cable assembly should be grounded. This requires an Ethernet connector which makes this connection. This connector must be carefully selected.

The stepper driver module is located adjacent to the Ethernet connector to allow short and simple routing. The Ethernet cable plugs into the back of the controller.

Note that shunt capacitors were added to the board in case of limit switch problems. I haven't detected problems so far, however, my Ethernet cable is only 50 feet in length. The shunt capacitors are unused. Others have reported successful operation with cables of 100 feet.

An Ethernet connector to screw terminal adapter board is used at the antenna. This board is available from Amazon or eBay. The BOM has a link to this board. An example 50 foot length Cat7 cable is also in the BOM. The cable is claimed to be waterproof and outdoor rated.

In spite of the CAT 7 cable shielding, there is enough RF inteference created by the stepper pulses to create audible noise in the receiver. However, this only occurs during auto-tune functions. So the controller deactivates the stepper driver during normal radio operation. This completely eliminates the noise source.

Also available from Amazon is a bulkhead mount style Ethernet feedthrough connector. This is useful if the wiring needs to cross some barrier, for example, if you intend to operate the controller from inside the ham shack. This connector is listed in the Amazon BOM.

# Chapter 8

# Controller Enclosure

The enclosure components are 3D printed. I'm a beginner in 3D printing, so what I did was print out some of the components of the original controller enclosure, which were published as STL files. I have adapted these designs for this new controller, and published the design files in FreeCad and STL format. The builder should probably optimize these files for their particular printer using a "slicer" application like Cura.

I have used nylon hardware for assembling the enclosure, although typical steel or brass hardware should be OK. There is a good selection of nylon hardware in a kit available from Amazon. A link to this kit is in the BOM.

The enclosure was adjusted for minimal convenient size to fit the PCB and display. The rear panel has several holes for the various connectors exiting the rear of the controller.

There are four holes in the base for attaching silicone feet. You will need to drill holes in the adhesive silicone feet to allow them to be attached with bolts. The adhesive is not strong enough for this application. The feet will not stay in place. The feet were purchased from Amazon. A link to them is in the BOM.

One problem with the style of enclosure construction is that the last piece is impossible to bolt on because you can't get your fingers inside a six-sided enclosure. There are two possibilities:

1. Leave the top unbolted. Insert the bolts, but they won't be secured. But they will keep the top cover from moving around.

2. Use "melt in" threaded inserts. These are commonly used in 3D printed parts. A threaded brass cylinder is melted into the plastic using a soldering iron with a special tip. I haven't tried this yet, but it is probably what I will do for the final assembly step.

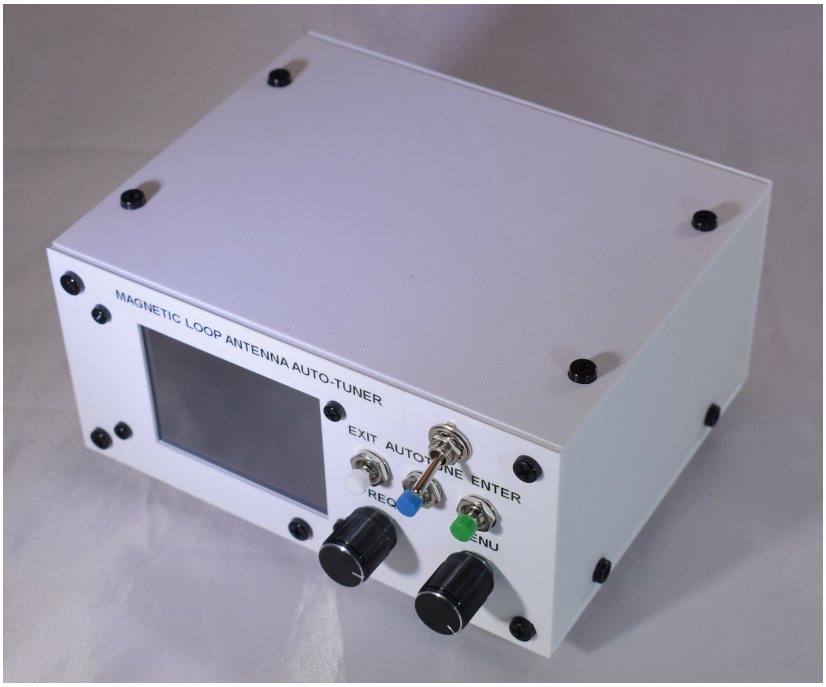


**Figure 8.1: Front view of the controller enclosure.**

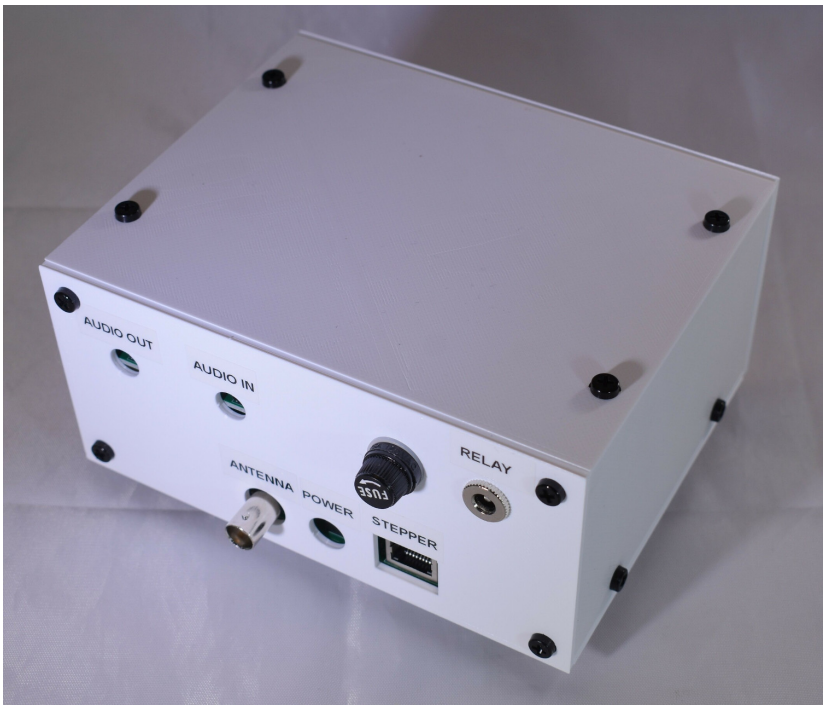


**Figure 8.2: Back side view of the controller enclosure.**

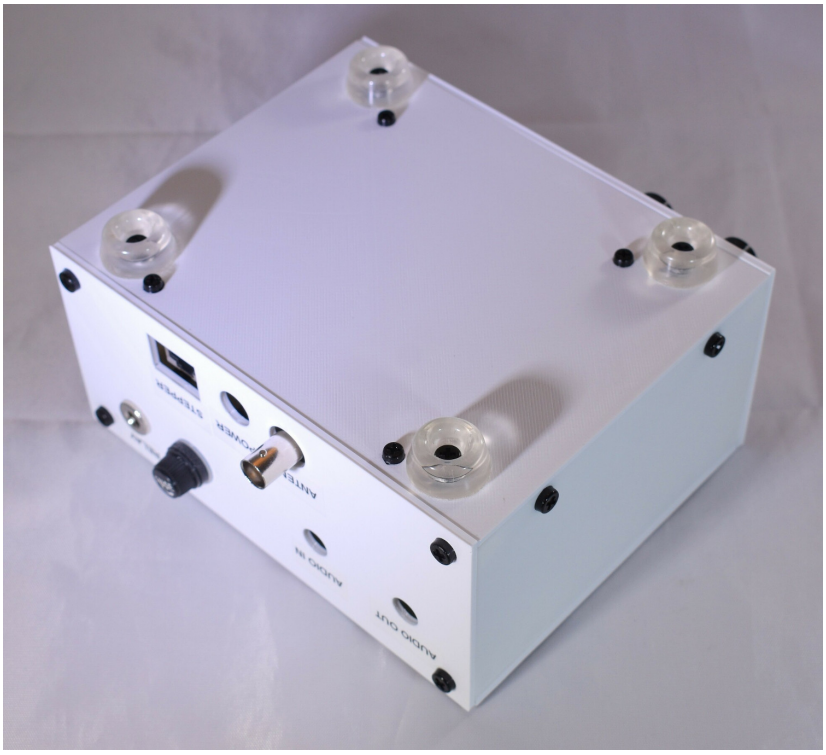


**Figure 8.3: Bottom view of the controller enclosure.**

The design files for the enclosure components are located in the magloop_pico_enclosure repository.

# Chapter 9

# User Interface and Controller Operation

The user interface is both simplified and expanded compared to the original project. The original project includes a total of seven pushbuttons: five discrete pushbuttons, and two buttons integrated into the encoders.

The force required to push the encoder buttons is enough to make the controller slide around. Those buttons were the first priority for elimination.

Changing the menu system to use “Enter” and “Exit” buttons eliminated two more buttons. The only remaining dedicated button is “AutoTune”, which is central to the mission of the controller.

A “menu function map” follows.

## 9.1 Controller Menu Function Map

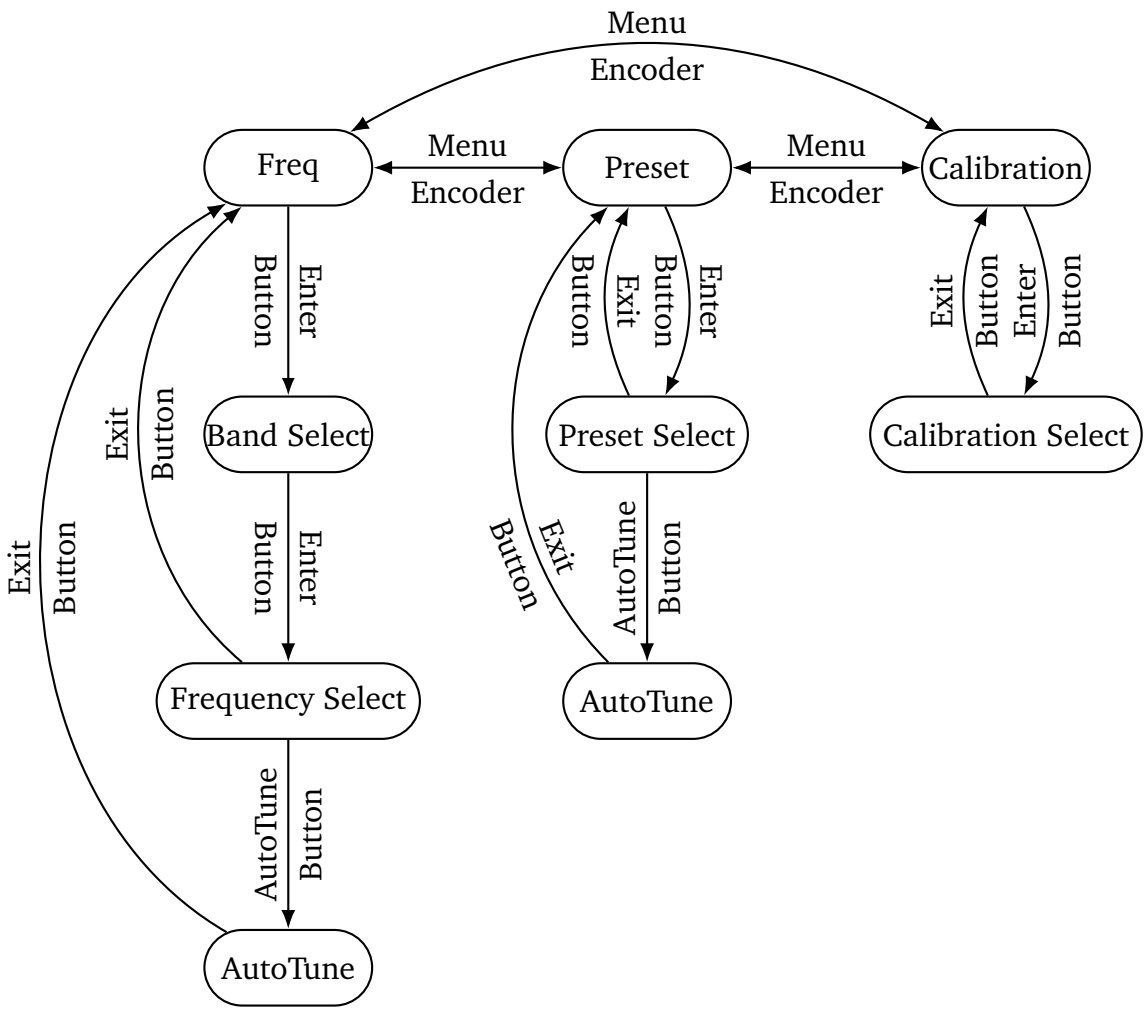


Figure 9.1: Magnetic Loop Antenna Controller Menu Function Map

## 9.2 Operating the Controller

The controller should be connected to an RF relay via 50 ohm coaxial cable and the relay control voltage, to the tuner via CAT-7 Ethernet cable, to +12 Volt DC power, and optionally the receiver audio should be connected in and out of the audio mute connectors.

Upon power-up, the display will show a “splash screen”. The stepper will be “set to zero”, and then it will move off of the zero switch. The display will transition to the top level of the menu system shown in the above diagram, and it is ready to tune the antenna to some desired frequency.

### 9.2.1 How to Check the Motor and Limit Switches Connections

First, make sure the DIP switches of SW1 are in the “ON” (closed) position. This will set the stepper driver to 1/16 micro-steps, which keeps the rotational speed as slow as possible.

The first thing to do is to check that the motor turns in the expected direction at power up. At power up, the “ResetStepperToZero()” function causes the stepper motor to start moving. With the power off, center up the limit switch bolt head (or whatever you used to close the switches) in the middle between zero and maximum limit switches.

Now, power up the controller and watch the motor. Power down as soon as you determine the direction of rotation! If the motor turns in the opposite direction you expect, reverse the stepper motor coil connections at the antenna tuner. The motor should now move in the desired direction.

Now you need to make sure the zero and maximum limit switches are not reversed (it’s easy to do). With power off, center up the limit switch bolt head between the two switches. Power up, and when the motor begins rotation, close one of the switches with your finger. Hold the switch and watch the behavior of the motor.

If the motor stops and does no further movements while the controller moves to the main menus, then that is the maximum limit switch.

If the motor stops, and then reverses direction and moves a little bit, that is the zero switch.

If you find that the switches are reversed, reverse the connections at the antenna tuner. If you used the recommended limit switches, all that is required is to pull the plugs and switch them.

### 9.2.2 Zero Limit Switch Set Up

The zero limit switch and the capacitor plates must be properly “synchronized”. This is critical for proper operation!

Upon power up, the controller will run the “ResetStepperToZero()” function. This function rotates the stepper in the direction of the zero limit switch. When switch closure is detected, the stepper rotation is stopped. The stepper is then rotated away from the zero limit switch in order to allow the switch to open. This becomes the stepper’s “zero calibration”. From now on, the controller will be moving the stepper from this zero reference step.

The capacitor should be fully meshed at the stepper zero. You will be able to see this visually by simply powering up the controller, and then see if the plates are fully meshed. If they are not, then loosen the pulley on the capacitor’s shaft, and rotate the plates to fully mesh them (maximum capacitance).

You may have to do this a few times; re-check the zero position by powering up the

controller. Make sure to get the synchronization correct before proceeding with the next step of initial calibration!

### 9.2.3 Initial Calibration

Before commencing to use the controller in normal operation, it must be calibrated. The calibration results will be stored in FLASH memory, and it is not necessary to perform this calibration again under most circumstances. If something on the antenna changes, or perhaps it is moved into a new location close to a building or a large metallic object, it might be a good idea to re-run the initial calibration routine.

The initial calibration is selected in the "Calibration Select Menu" which is in the top level of the menu system. Use the Menu encoder to highlight the Initial Calibration, and press the Enter button. The calibration will begin. This is a slow process which takes several minutes to complete.

What the Initial Calibration algorithm does is find the stepper motor positions at the band edges for each band. These positions are stored in FLASH memory. The band edge positions are then used to approximate the stepper motor position for a desired frequency entered by the operator. The AutoTune function then takes over and fine-tunes the stepper position for minimum SWR. This is a good system, and the originators of the original project should be congratulated for their clever algorithm design.

There are two other calibration routines which can be selected. The "Full Calibration" uses the stored stepper positions to speed up the process. This could be used after the antenna is moved to another location as suggested above. The "Band Calibration" runs the calibration process on a single selected band only.

### 9.2.4 Normal Mode of Operation

The operator can choose the desired antenna resonant frequency using either the "Freq" or the "Preset" menu selections.

The Freq menu allows for manual entry of the target frequency using the Mode and the Freq encoders. After the target frequency is selected, push the "AutoTune" button. The controller then proceeds to find the stepper position for minimum SWR. When the process is complete, a plot of the successful AutoTune will be shown. It is also possible to steer the stepper and target frequency manually in this state. Press "Exit" to return to the frequency entry mode. The "transmit inhibit" LED indicator should turn off, and it is possible to begin normal transceiver operation on this frequency.

The "Preset" menu allows only a set of pre-determined frequencies. Otherwise, operation is similar to that of the Freq menu.

Please note that the controller generates a lot of noise during the AutoTune operation. This is why it is recommended to add the audio mute board to the system. The audio mute board will halt receiver audio when the controller is operating in AutoTune mode.

It is recommended to always ascend back to the Freq or Preset menu levels. This puts the controller in “Normal Operation” mode which means it is also in minimum power and noise mode. Don’t leave the controller in the manual control mode state for an extended period. In the future, a change may be implemented to move the manual tune mode to its own place in the menu system so that this is no longer a concern.

# Chapter 10

# Speaker Audio Muting

One annoying problem became apparent during system testing of the controller with antenna and radio attached. The architecture of this controller is that it provides the RF signal for measuring SWR during auto-tuning. This is the purpose of the AD9850 DDS module.

However, this large signal is going to be heard in the receiver if it is tuned to the desired frequency! It is a very loud tone, and although brief, it is very irritating.

The solution to this problem is simple, but it requires additional circuitry. There was not enough room to implement this on the controller board, so an auxiliary PCB was designed to take care of this function.

The circuit functions by simply muting the receiver audio during auto-tuning. The audio muting PCB is designed so that it can be built as a "line level" muting circuit or a simple electromagnetic relay to connect-disconnect the speaker output.

I didn't know the simplest way to mute audio (besides a relay), but an internet search yielded this web page:

https://www.sound-au.com/articles/muting.html

The line-level muting circuit is based on information from this web page. The simple relay circuit is sort of intertwined with the line-level circuit, but it should be easy to see how to build the board depending on the user's preference.

The board requires two simple two-wire cables to connect to the controller board. Because these connections have power and ground, the connectors had to be polarized. JST type connectors were used.

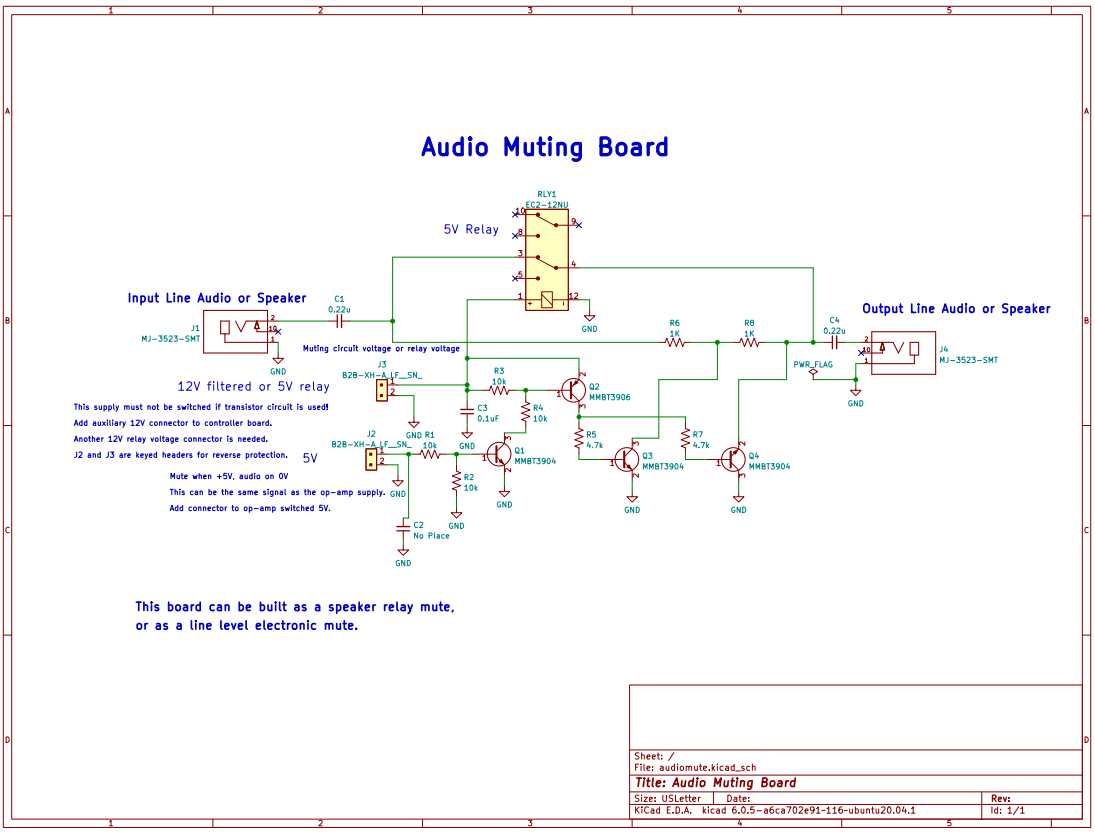


**Figure 10.1: Audio Muting Board Schematic**

# Chapter 11

# Bill of Materials

Obtaining all of the parts for the controller and antenna is (in my opinion) the most difficult part of the project!

I have used three sources for parts, not including the 3D printed parts:

- Amazon
- Digikey
- Official Raspberry Pi Pico dealers

In the current environment as I write this in 2022, supply chain issues require some extra work in getting all of the components together. Substitutions may be required. Fortunately, none of the parts used are particularly exotic.

I was having the most trouble with the SOT-23 bipolar and PMOS transistors. Fortunately, there is usually a "substitute" listed on the component's page at Digikey.

What I have provided in terms of parts lists are three spreadsheets:

- Links to Amazon parts.
- A spreadsheet downloaded from a Digikey "list" generated by the website.
- A CSV file exported from the Kicad schematic.

The spreadsheets are located in the "bom" folder inside the magloop_pico_pcb repository.

I can't guarantee the accuracy of these BOMs, as the parts may come and go from the websites. With regards to substitutions, you can find lots of expertise at the Software Controlled Amateur Radio groups.io.

One problem with ordering parts, particularly from Amazon, is that you may have to order more than you need. Fortunately the prices are still quite low, so you have some spares and a stock of parts you could use in a future project. Some of the parts may be ordered from either Amazon or Digikey. You may have good luck with one or

the other, however, if you are worried about the authenticity or quality of the parts then order from Digikey.

The printed circuit boards were fabricated by PCBWay. I have enjoyed excellent service and quality boards from this company.

A few items are not listed in the BOMs, for example, the PVC pipe and the labels made on a home labelmaker.

You will need to provide an RF relay with a 12 volt coil. I have used a "Dow Key" type with BNC connectors found at a local hamfest.

With regards to ordering the Pi Pico, you should check the official dealers first. They are listed on the Raspberry Pi website by country. These dealers charge for shipping, so take that into account when ordering. You might be able to get a better price on Amazon, eBay, or Digikey.

## 11.1 Hardware Issues

I'm using nylon bolt kits from Amazon which are inexpensive, but I am using only certain pieces in the kit. This is wasteful, and I need to find a source which will allow me to order only what I need.

There are other instances where ordering from Amazon is cheaper, but you get more than you need. I encourage you to try to lower costs by ordering exact quantities from Digikey or other distributors. Sometimes it takes quite a bit of effort searching catalogs to find the right combination of cost and availability.

# Chapter 12

# Construction

In this chapter I will throw out a few ideas to make the assembly of the controller faster and easier.

## 12.1 PCB Assembly

First, get all of the parts ready and reasonably sorted. Get all of the surface-mount resistors and capacitors into a bin, semiconductors in a bin, the leaded parts in another bin, the modules in another, etcetera.

It is a good idea to assemble everything on a grounded static mat. I did not adhere to this 100% of the time and nothing has failed yet, although latent defects can show up later. Just try to keep from zapping your parts, OK?

In general, you should start with the lowest height parts and then the next height, and the next, etcetera.

Start with the passive surface-mount parts. These are all very low height on the board, and it is easier to solder them without taller leaded parts obstructing the soldering iron. Get all of the surface-mount resistors and capacitors soldered to the PCB. Use a good clean fine-point soldering iron time, and fine solder.

Up next are the transistors and diodes. Be especially careful with the Schottky diode which routes power to the Pi Pico. The cathode marking is hard to see, so you might want to check with the diode function on a VTVM to make sure you are orienting the diode in the correct direction. The SOT-23 NPN and PMOS transistors are marked on the PCB. There is also a SOT-23 3.0 volt regulator device which is located underneath the Pi Pico.

Next, solder the four (DO35) Schottky detector diodes, making sure to insert with the correct polarity (“K” on the PCB means cathode).

Next are the through-hole LM358 operational amplifiers and DIP SW1.

Next are the JST connectors. Be sure to make the orientation of these connectors

consistent with respect to the ground pin. These cables carry supply voltage, so do not cross-wire them to ground!

Next, there are six headers for the modules. Be sure to get these soldered in straight, as this makes insertion of the modules easier.

Next, the IDC connectors for the display, switches, and pushbuttons.

Next, the BNC and coaxial power connector. The BNC connector has a lot of thermal mass, so you should change to a larger soldering iron tip. You may need to use a larger soldering iron, or use two soldering irons.

Next, the four leaded electrolytic capacitors, with care to insert them with the correct polarity.

Next, the Ethernet connector, and make sure the ground tabs are properly soldered.

Next, the LM7805 regulator, making sure it is inserted with the correct orientation. Add an aluminum heatsink to the LM7805, as it dissipates enough power to get quite warm.

## 12.2 Ribbon Cables

The ribbon and power cables should be completed before installing the PCB in the enclosure.

**Figure 12.1: Constructing the display ribbon cable.**

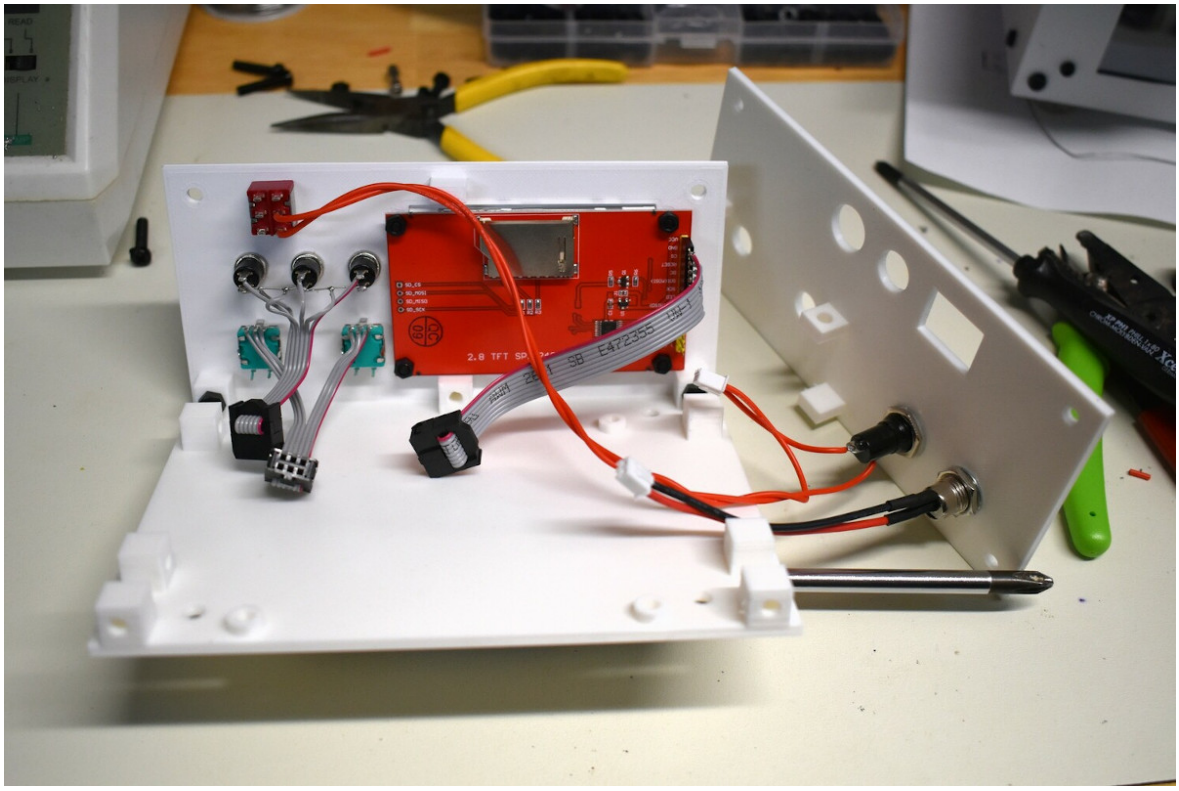

**Figure 12.2: The switch, encoder, display, and power cables.**

## 12.3 JST Cables

There are two JST cables which connect to the audio mute board. The relay and main power and fuse circuit also use JSTs which plug into the controller board. JST was used because they are polarized connectors. Thus it is important to carefully construct them in order to avoid short circuits.

You can search and find youtube videos which show how to use a wire stripper and the crimping tool to make the cables. It does take a little bit of practice to get the stripped wire length just right. But once learned you will have a skill you can apply to other projects.

**Figure 12.3: A typical crimping tool which can be used to make JST cables.**

## 12.4 Power and Fuse Wiring

The coaxial power plug, fuse, and power switch are wired in series. The wires to the power switch on the front panel can be tucked underneath the PCB for neatness.

Don't forget to install a 1 amp fuse in the fuse housing.

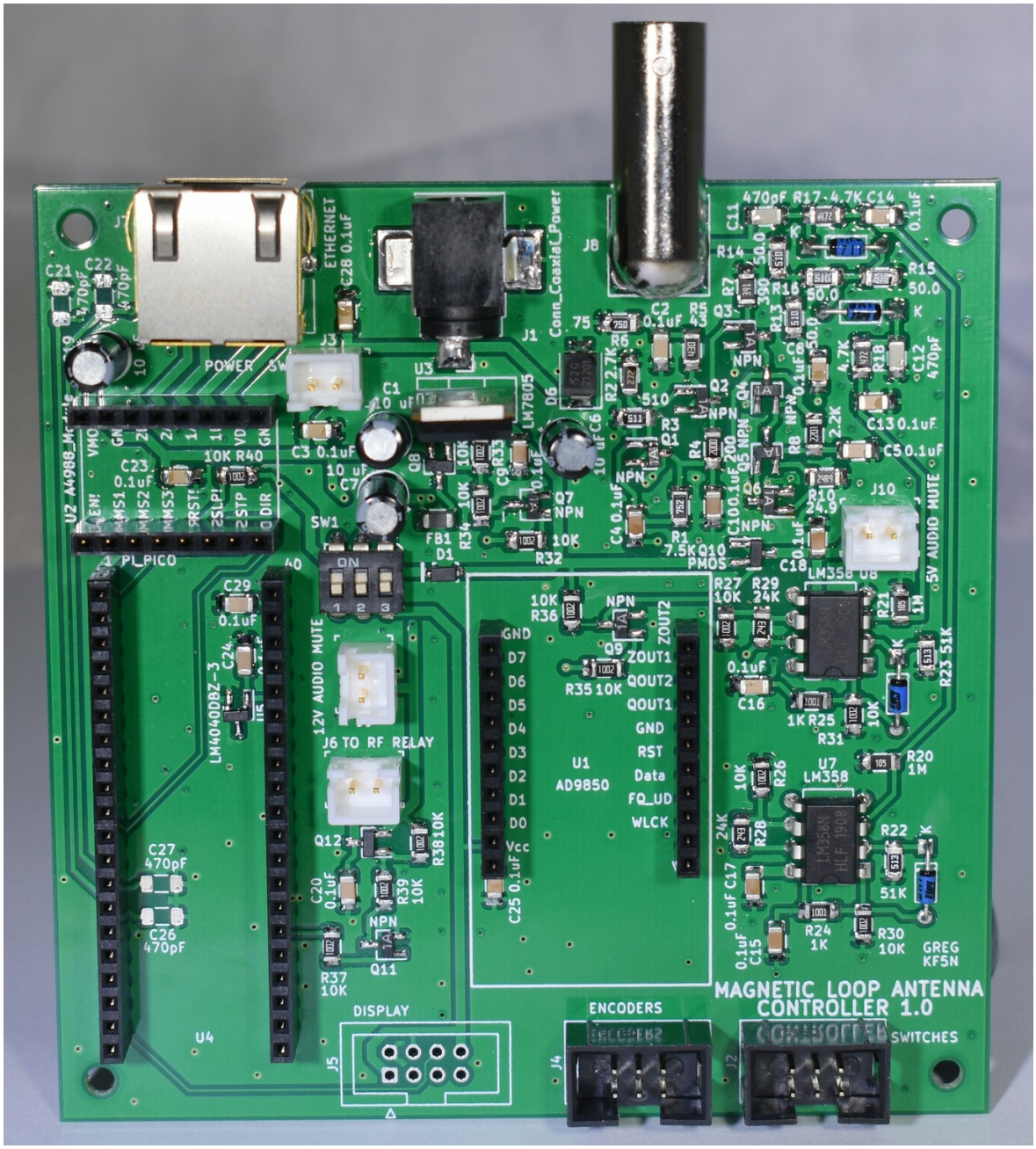


**Figure 12.4: A newly constructed board without modules.**

## 12.5 Adding a “Transmit Inhibit” LED

The controller does not include any sort of indication of when it is OK to transmit.

It is recommended to add a “transmit inhibit” LED to the RF relay. This works because the relay is energized only when the controller is actively tuning the antenna. So it is sufficient to add an LED and a series limiting resistor across the relay’s coil. This is illustrated below.

As there is no physical “transmit inhibit” signal for the transceiver, it is up to the operator to avoid transmit during AutoTune operations. During AutoTune, the relay should present an open circuit to the RF transceiver. Hopefully, the RF transceiver has a sophisicated power control, and it will back off transmit power and possibly even warn the operator of the high SWR condition.

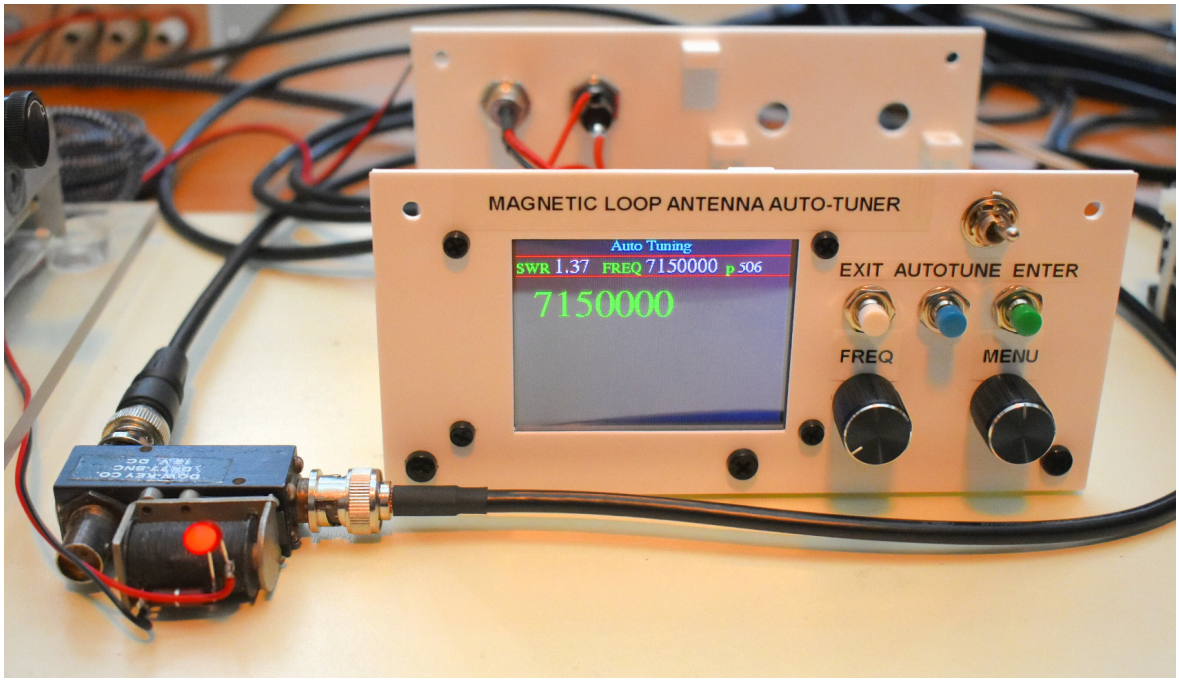


**Figure 12.5: The “transmit inhibit” LED is seen on the left.**

# Chapter 13

# Test Suite

## 13.1 Buttons Test

The buttons test is used to determine if the three buttons and two tuner switches (zero and maximum) are functioning and correspond to their labels. The user simply pushes the button and watches the response on the display.

## 13.2 Encoders Test

The encoders test verifies operation of the encoders in clockwise and counter-clockwise directions. The user rotates the frequency and menu encoders and observes the response on the display.

## 13.3 Stepper Motor Test

This test "waggles" the stepper motor back and forth a few steps. The display will indicate the direction in which the motor should be turning. If the motor is moving in the opposite direction, the motor coils should be switched. This is easily accomplished at the motor.

### 13.3.1 Stepper Motor Current adjustment

The stepper motor driver module includes a very small potentiometer. This is used to adjust the motor current. This setting does not appear to be critical, and many times the setting has been left in the middle and the motor works perfectly.

However, it is typically recommended to adjust the current to the specification of the particular stepper motor chosen for the tuner. However, this "rated current" also assures that the rated torque is achieved.

The rotation of a capacitor does not require significant torque. Thus it is suggested that the "rated current" in the stepper motor specifications is not required in this application. The system has been successfully operated well below the rated current of the stepper motor. Therefore it is suggested to operate the stepper motor below its rated current, and if this is problematic, meaning the stepper motor misses steps, the current can be increased.

The output current of the A4988 driver module is adjusted using the very small potentiometer on the board. This adjusts the "Vref" voltage which can be measured by probing the the top of the potentiometer which is the metallic wiper of the potentiometer. The current limit setting is also a function of the sense resistors on the module, which can vary depending on the manufacturer of the board.

Numerous youtube videos and web pages are available which describe setting the current of the stepper driver. This simple equation determines the current setting for the A4988 module. Note that you must plug in the current sense resistor value $R_{cs}$ which is used on your driver module.

$$V_{ref} = 8 * I_{max} * R_{cs}$$

Plug in the desired $I_{max}$ and $R_{cs}$ to calculate $V_{ref}$. Adjust the potentiometer to $V_{ref}$ and the current limit is set.

### 13.3.2 Hardware Parameters

There are several user-settable parameters which allow the controller to be "tuned" to the unique hardware of a magnetic loop.

#### Stepper Motor Driver Step

The stepper driver module uses the A4988 "Microstepping Driver" by Allegro Microsystems:

https://www.allegromicro.com/en/Products/Motor-Drivers/Brush-DC-Motor-Drivers/A4988

This driver is capable of full, half, quarter, eight, and sixteenth step operation. This is set using the DIP switch on the controller board (SW1).

The DIP switch should be oriented such that "ON" is towards the back of the board. Thus pushing a switch back is the "ON" position.

The following table shows the switch positions required for a particular step size.

#### Zero Offset

The "Zero Offset" determines the number of steps from zero switch closure to fully meshed variable capacitor. This is in effect a calibration of the stepper to a known position.

| DIP-1 | DIP-2 | DIP-3 | STEP |
|---|---|---|---|
| OFF | OFF | OFF | 1 |
| ON | OFF | OFF | 1/2 |
| OFF | ON | OFF | 1/4 |
| ON | ON | OFF | 1/8 |
| ON | ON | ON | 1/16 |

Table 13.1: DIP Switch settings for step size

After the controller powers up, the zeroing process will begin. The stepper will begin rotating the capacitor in the direction of the zero switch. When the zero switch closes, the stepper will be decelerated and brought to a stop. The capacitor should be a little past fully-meshed. The reason for this is that the stepper will then be rotated in the opposite direction towards exactly fully meshed. The stepper position is then set to zero by the firmware, establishing a "calibrated" stepper position.

The value of the Zero Offset is the number of steps away from zero switch closure at which the position will be set to zero. The Zero Offset must be empirically adjusted to get this just right.

This will be an iterative process between the hardware menu to adjust the offset, and then exit and press the zero button. It should be possible to finish the adjustment quickly if the hardware is properly adjusted.

### Backlash

The backlash setting determines how many steps to offset a normal stepper move during autotune. The stepper will always approach from the low-frequency side of resonance. This means any "slack" in the hardware will be taken up in the first few steps of stepper rotation. This is important for consistent and repeatable autotune!

The number of steps required will be dependent on the characteristic mechanical action of the antenna's tuner assembly. Belts and gear drives will introduce more or less slack.

It is better to start out with a larger value of backlash. The trade-off is that autotune will take longer. Backlash can be reduced for a shorter autotune process, however, accuracy will deteriorate if the backlash value is made too small. Emperical optimization will achieve a reasonable compromise between autotune speed and accuracy.

### Coarse Sweep

"Coarse Sweep" is used when SWR is measured high, and the stepper is moved in larger steps, typically 20. This is used to speed up the tuning process. This is most easily observed during initial calibration where the band limits are unknown and must be found by sweeping the capacitor through its entire range. The stepper will

be switched to single-step when the SWR decreases below a hard-coded level which is currently set to 3.0.

Future updates may decrease the coarse sweep threshold below SWR=3.0 if it is determined the autotune speed will be increased without sacrificing accuracy.

### Acceleration and Speed

The acceleration and speed settings are not critical. This will have the biggest impact on long stepper moves like between bands. This will also affect the zero calibration, although the zero calibration speed is divided by two in order to improve repeatability and to protect the zero switch.

# Appendix

## 13.4 Github Repositories

The project requires a total of six Github repositories:

1. magloop_pico_project. Documentation and five submodules.
2. magloop_pico_pcb. Controller PCB design files.
3. magloop_pico_antenna. 3D printer files for antenna components.
4. magloop_pico_enclosure. 3D printer files for the controller enclosure.
5. magloop_pico_code. C++ code for the controller.
6. magloop_pico_audiomute. PCB files for the audio mute board.

The entire project can be downloaded with a few Git commands as follows:

```
git clone https://github.com/Greg-R/magloop_pico_project
cd magloop_pico_project
git submodule update --init
```

The PDF documentation is located in magloop_pico_project. Five other folders in the same directory complete the list above.

To obtain the latest changes to any or all repositories, use these commands:

```
cd magloop_pico_project
git pull
git submodule update
```

The submodules can be cloned individually if desired. Here are the URLs:

- https://github.com/Greg-R/magloop_pico_pcb
- https://github.com/Greg-R/magloop_pico_antenna
- https://github.com/Greg-R/magloop_pico_enclosure
- https://github.com/Greg-R/magloop_pico_code
- https://github.com/Greg-R/magloop_pico_audiomute