Automatic Magnetic Loop Controller
Bill of Materials and Building Instructions
for Printed Circuit Boards Rev3/Rev4 (using Pololu A4988 or DRV8825)

Resistors, Capacitors and Inductors:

R1: 4.7 ohm 1/4W	(example: Digikey 4.7QBK-ND)
R2, R3, R4, R8, R9, R10, R11, R12, R13, R14: 1 kohm 1/4W	(example: Digikey 1.0KQBK-ND)
R19, R21: 4.7 kohm 1/4W	(example: Digikey 4.7KQBK-ND)
R20: 2.2 kohm 1/4W	(example: Digikey 2.2KQBK-ND)
R22: 10 kohm 1/4W	(example: Digikey 10KQBK-ND)
R23, R25, R26: 100 ohm 1/4W	(example: Digikey 100QBK-ND)
R5, R6, R7, R24: Not used	
R15, R16: See description of Power / SWR meter	
R17, R18: See description of Power / SWR meter	
RV1: 10 kohm linear	(example: Digikey 1993-1072-ND)
C1, C6: 100 nF, ceramic 50V	(example: Digikey 478-6008-ND)
C2, C3, C4, C9, C10, C11, C12, C13, C14 C15, C16: 10 nF, ceramic 50V	(example: Digikey 478-5739-ND)
C23, C24: 1 nF, ceramic 50V	(example: Digikey 478-5109-ND)
C25, C26: 4.7nF, ceramic 50V	(example: Digikey 445-4746-ND)
C5, C17, C18: 100 uF, electrolytic, tantal or aluminum, 35V or higher	(example: Digikey 565-4085-ND)
C7, C8: Not used	
T1, T2: 2x51 uH Common Mode Chokes	(example: Digikey CMS1-8-R, see also text on next page)

Semiconductors:

D1, D5: SD101C (Schottky diode)	(example: Digikey SD101CVSCT-ND)
D4: 1N5820 (3A Schottky diode)	(example: Digikey 1N5820-E3/54GICT-ND)
U1: Teensy 3.2 microcontroller with pins	(http://www.pjrc.com/store/teensy32_pins.html) (There may be a local source near you: http://forum.pjrc.com/threads/23601-Official-Distributors)
U2, Pololu DRV8825 (or A4988) Stepper Motor Driver Carrier	(https://www.pololu.com/product/2133) (There may be a local source near you)
U4: LM7805 (5V Regulator, 1A)	(example: Digikey LM7805CT-ND)
U3: Not used	

PCB headers for Connectors (if used):

SW1, SW2, SW3, Power: 2 pin SIL connectors (male)	(example Digikey: SAM1053-50-ND)
Serial Data: 3 pin SIL connector (male)	- 50 pin breakable header, only one needed. Can
Encoder, Stepper Motor, Expansion: 4 pin SIL connector (male)	be broken up and used for all the PCB headers)
LCD: 16 pin SIL connector (male)	

External components:

SW1: SPST Pushbutton switch	(example: Digikey 679-1021-ND)
SW2 and SW3 combined, using one Mom-Off-Mom rocker switch	(example: Digikey 563-1661-ND or 360-2995-ND)
SW4, SW5, SW6, see SW1 (only if Power/SWR Autotune Option)	
20x4 LCD, HD44780 compatible	(example eBay search words: “LCD 20x4 HD44780”)
Rotary Encoder, mechanical or optical – 32 PPR or higher	(see text on next page)
Stepper Motor: Two Coil/Four Wire	(see text on next page)

IC Sockets (if used – highly recommended):

2x 16 pin sockets for U2 and U3	(example: Digikey ED3046-5-ND)
2x 14 pin SIL connectors for U1 (female)	(example: Digikey SAM1093-14-ND)

End Stop Sense Option:

R25, R26: 100 ohm	(example: Digikey 100QBK-ND)
C19, C20, C21, C22: 10 nF	
T3: 2x51 uH Common Mode Choke	
D2, D3: Zener diode, 4.7V	
End Stop Sense: 3 pin SIL connector (male)	

Common Mode Chokes

The 2x 51 uH common mode chokes are used to squash any RF flowing back from the antenna through the stepper motor cable, potentially locking up the microcontroller. The chokes also ensure that the controller does not radiate RFI into the antenna.

I really hate to wind toroid cores, so I use surface mount chokes, Digikey CMS1-8-R, specified at 2.2A.

The surface mount chokes are a bit expensive. Another option which may be just as good is to use FT37-43 or FT50-43 ferrite cores, 2x 10 or more turns, bifilar or sectional wound.

Rotary Encoder

I use a 128 pulses per revolution (ppr) Optical Encoder (attached to the big black knob, Digikey KN1251B1/4-ND, on the front plate, see Figure 1). Another good choice may be a 64 ppr Optical Encoder with a builtin pushbutton (Digikey EM14A0D-C24-L064S-ND) for the Menu/Enact switch (SW1).

Here is a 128 ppr Optical Encoder that I have used:
http://www.ebay.com/itm/Oak-Grigsby-Rotary-Encoder-128-Pulses-Rev-/281135090460?pt=LH_DefaultDomain_0&hash=item4174f50b1c

This Encoder plugs directly into the ENC header on the PCB. Take care with polarity, GND on one side and 5V on the other. It will not survive being reverse polarized.

To change the Encoder resolution (sensitivity), the following parameters can be adjusted in the ML.h file, prior to compiling the firmware for the Magnetic Loop Controller:

```
#define  ENC_MENURESDIVIDE  16 // Encoder resolution reduction when in Menu, 1 = no reduction
#define  ENC_TUNERESDIVIDE   2    // Encoder resolution reduction when turning stepper motor, 1 = no reduction.
```

These values are for a 128 ppr Encoder. For a 64 ppr Encoder you would set these values to 8 and 1 respectively.

Stepper Motors, choices and considerations

The DRV8825 stepper controller circuit used in this project is capable of driving the stepper motor at up to 1.5 A per phase, possibly slightly more. Microstepping is achieved through providing current to both phases at the same time, but in different proportions for each microstep. In other words, the Stepper Motor controller circuit provides accurate current limitation. The current limiting is a useful feature as it enables you to dial in the torque necessary to reliably turn the capacitor, but not very much more so, a good feature to protect a multiturn capacitor against being destroyed if turned to far.

The DRV8825 stepper controller circuit has a small trimmer potentiometer which used to adjust the torque of the Stepper Motor by adjusting the maximum current to the Stepper Motor. The trimmer potentiometer should be adjusted to a setting which is sufficient to reliably turn the motor without missing steps, but not much further.

I have used several types of Nema-17 format two-phase Stepper Motors, here are a few choices that work well with the controller:

- 30 ohms per phase, each phase specified at 0.4A max. Approx USD $20 including shipping if purchased on eBay. Use the search criteria: "Nema 17 Stepper 2800g/cm 0.4A" or similar. I have found that typically about 250mA per phase is required to reliably turn the capacitors when using this motor (see antenna descriptions further down the page). By limiting the torque of the motor to the bare minimum needed, there is no need for end stop switches. This stepper motor is rather weak and will not go very fast before losing torque, 100 RPM is probably the highest usable speed.
- The motor I would like to recommend is a Nema 17 format Stepper with a specified Holding Torque of 4.2kg/cm, considerably stronger than the first choice above. Having a much lower phase coil impedance, it is also capable of quite high speeds before losing torque. I have set this motor up to Autotune for SWR at a whopping rate of 360 RPM (1200 steps/second, 4 microsteps, 4x speedup). For this performance the motor is adjusted for a current of 1.4A. I am currently using it with one of my antennas at this speed. Feels like using a broadband antenna. On eBay, use the search criteria: "Nema 17 1.7A 2 phase 4-wire 4.2kg.cm"
- If the Capacitor is very difficult to turn, then you may need a stronger stepper motor, for instance one with a 5:1 reduction gear, like this one:
  http://www.omc-stepperonline.com/gear-ratio-51-planetary-gearbox-high-torque-nema-17-stepper-17hs191684spg5-p-40.html Note that these guys have US/EU/… websites – select the one that fits you.
- Here is an example of a Stepper Motor that would be suitable for Butterfly capacitors. This one has a gear ratio of 100 to 1. I have been told that this one is available at most hobby/robot kit type stores:
  http://www.robotshop.com/en/12v-17a-667oz-in-nema-17-bipolar-stepper-motor.html
  You would set this motor up at maximum speed of 450RPM, the real speed would of course be divided by 100.

You will find that there is a considerable amount of backlash in the coupling and even in the capacitor itself. To overcome this the

Loop Controller has an Adjustable Backlash Compensation function. The Backlash Compensation can be adjusted from 0 to 400 steps. Default value is 25 steps. Some Vacuum Capacitors may need slightly more, others slightly less. If you are using a reduction gear-box, then you will need to increase this value. A Butterfly capacitor with a 100:1 gear may need a value around 250.

Power / SWR and Autotune Option, see pages 9 – 11.

Serial Port, proper RS232 levels or not

The Serial port can work with both TTL and RS232 levels. IF RS232, then the firmware is set up to invert the RX and TX levels (see the Setup Menu Function: Serial Port). The transmitted levels from the Controller are 0 and 3.3V, which is not really compatible with the formal standard for RS232 levels, however, in all cases tested so far, this has been found to work without any problems – provided that the serial cable is not very long.

Normally this is not needed unless your serial cable is very long (>5 meters), but if for some reason full RS232 level compatibility is required, then add a **MAX232 circuit**. Ready built MAX232 type circuits including a 9 pin D-sub connector can be purchased on eBay for US $4 or less.

eBay search criteria: "max232 rs232 to ttl converter"

An aluminum box to house the Magnetic Loop Controller

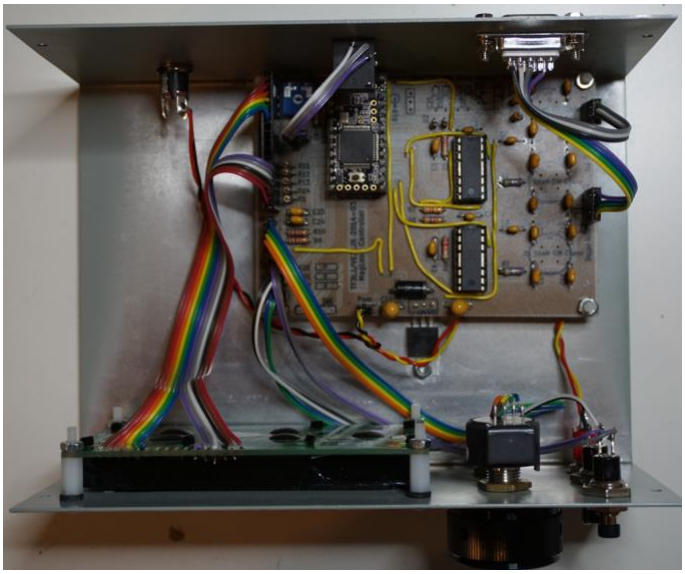
The box I used for the controller is a generic aluminum box, 7x5x3 inches. It is quite roomy, the critical issue is to fit the LCD, knob and buttons on the front.

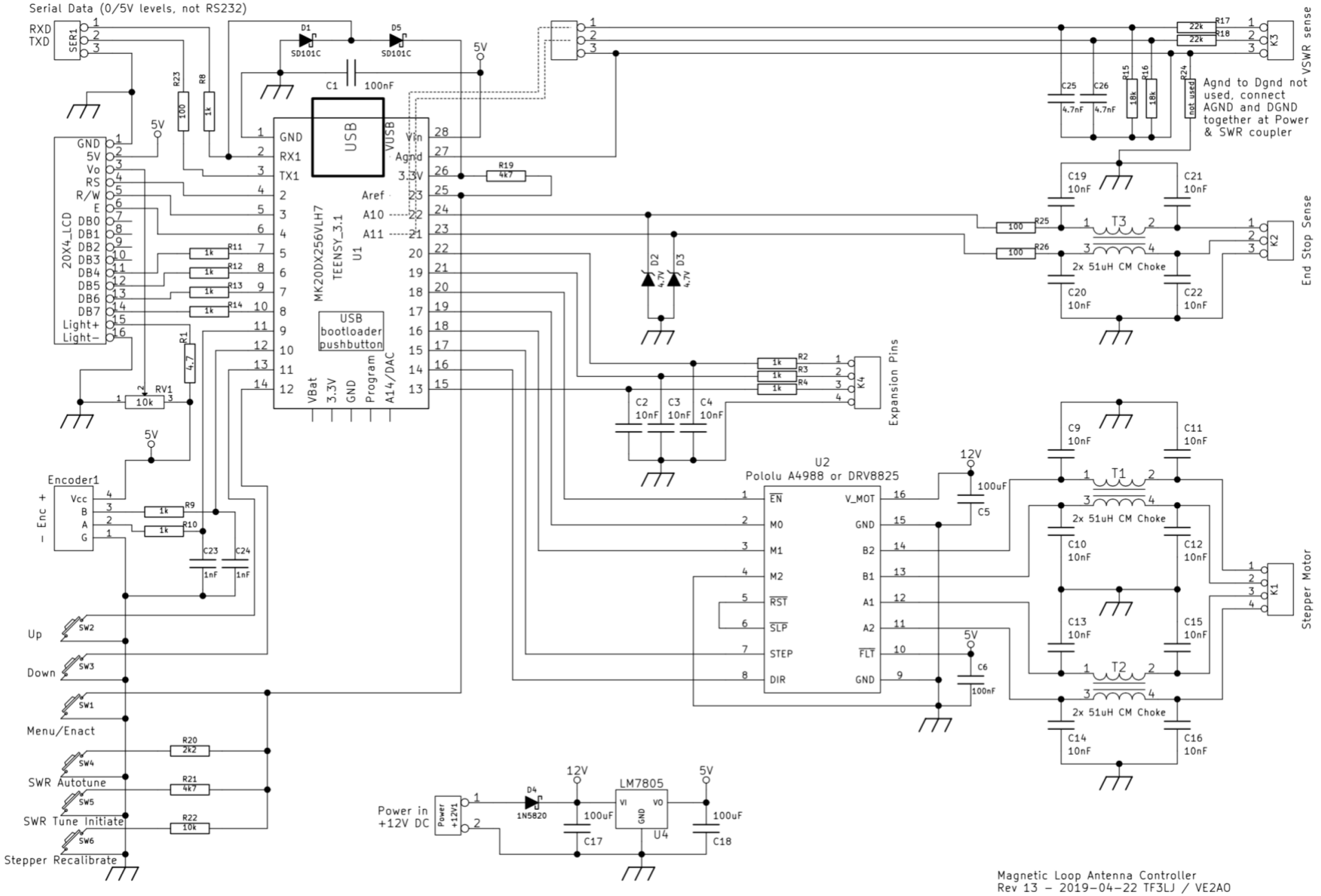
This is the box I used: Digikey HM326-ND

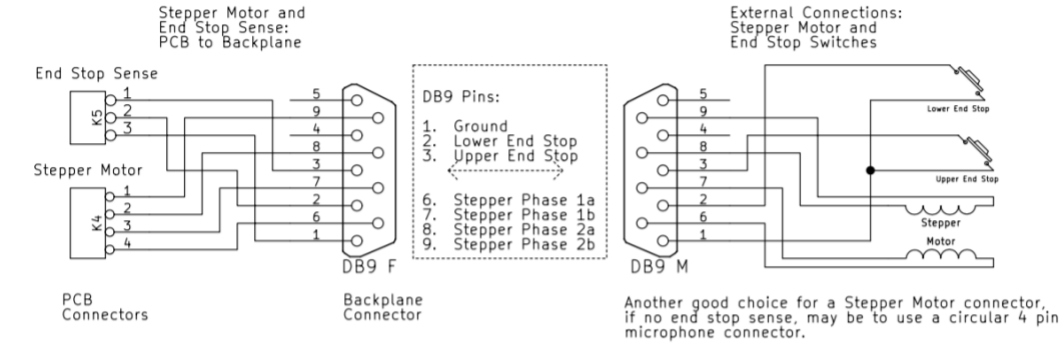
Bezel around the 20x4 LCD

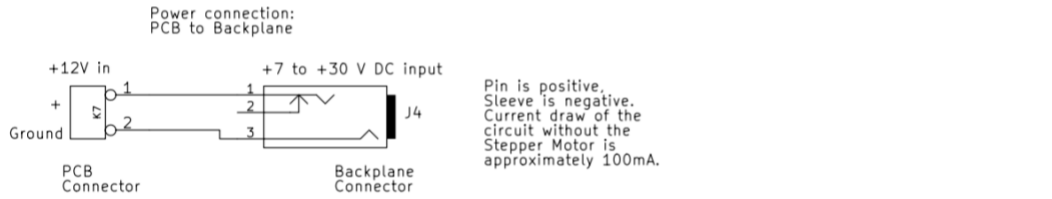
Some people have asked me about a Bezel for the 20x4 LCD. I have ordered Bezels a couple of times from dhmicro.com. Cost $3.50 + shipping. Very happy with them:
http://dhmicro.com/plastic.html

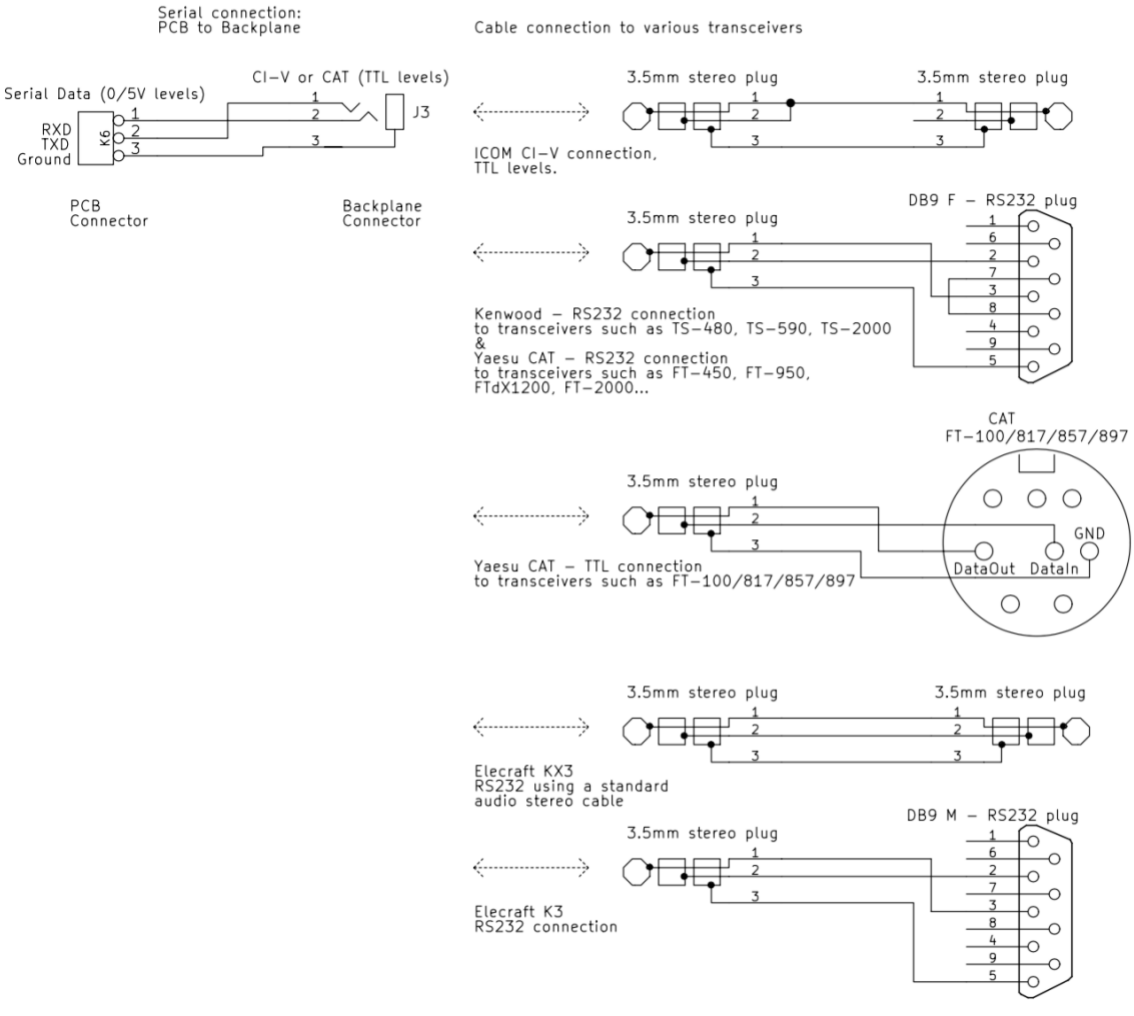
Here are a couple of pictures of the Controller, without the Power/SWR and SWR Autotune Option and using push switches rather than a rocker switch for Up and Down (SW1 and SW2):




Serial Data (0/5V levels, not RS232)
RXD
TXD
SER1
D1
SD101C
D5
SD101C
C1
100nF
5V
R23
100
R8
1k
20X4_LCD
GND
5V
Vo
RS
R/W
E
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
Light+
Light-
R11
R12
R13
R14
R1
4.7
RV1
10k
USB
MK20DX256VLH7
TEENSY_3.1
U1
USB bootloader pushbutton
VBat
3.3V
GND
Program
A14/DAC
Vin
Agnd
3.3V
Aref
A10
A11
VUSB
R19
4k7
D2
4.7V
D3
4.7V
R2
R3
R4
K4
Expansion Pins
C2
C3
C4
10nF
Encoder1
Enc +
Enc -
Vcc
B
A
G
R9
R10
1k
C23
C24
1nF
Up
Down
Menu/Enact
SWR Autotune
SWR Tune Initiate
Stepper Recalibrate
SW1
SW2
SW3
SW4
SW5
SW6
R20
2k2
R21
4k7
R22
10k
U2
Pololu A4988 or DRV8825
EN
MO
M1
M2
RST
SLP
STEP
DIR
V_MOT
GND
B2
B1
A1
A2
FLT
GND
12V
100uF
C5
C6
100nF
Power in
+12V DC
Power +12V1
D4
1N5820
LM7805
VI
GND
VO
C17
C18
U4
100uF
100uF
C25
C26
4.7nF
4.7nF
R15
R16
18k
18k
R17
R18
22k
22k
R24
not used
K3
VSWR sense
Agnd to Dgnd not used, connect AGND and DGND together at Power & SWR coupler
C19
C21
C20
C22
10nF
R25
R26
100
T3
2x 51uH CM Choke
K2
End Stop Sense
C9
C11
C10
C12
C13
C15
C14
C16
T1
T2
2x 51uH CM Choke
K1
Stepper Motor
Magnetic Loop Antenna Controller
Rev 13 – 2019-04-22 TF3LJ / VE2AO







The top diagram above shows one possible way of wiring the Controller to a Stepper Motor (and end stop switches if used) through a DB9 connector on the Controller back plane.

The middle diagram shows the power connector.

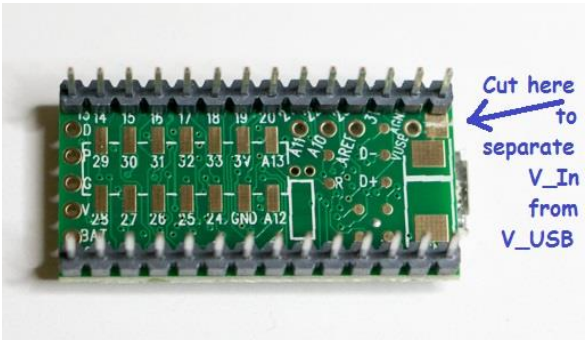
The bottom diagrams show the serial connection to the backplane and several serial cable versions for the various types of Radios that the Controller can communicate with.

In the case of ICOM CI-V, the Controller TXD output is configured through software to be open-drain (open-collector equivalent) style and the Controller RXD input is configured to provide a 30 kohm pullup to 3.3V.

Note that the Controller does not output proper RS232 levels, however this has not been found to be a problem with any of the radios tested to date. If proper RS232 levels are desired, then the simplest way is to use a MAX232 converter circuit. In that case, the Controller has to be configured to output TTL levels – the MAX232 inverts the levels. There is no need to build a MAX232 circuit, these can be purchased for approx three dollars on eBay.

Modification to the Teensy 3.2 chip:

The USB port is used to upload firmware updates. It is also used as a second serial port providing for functions such as storing and recalling setup data. The firmware can relatively easily be modified for the USB port to receive the frequency information from the Radio as well.

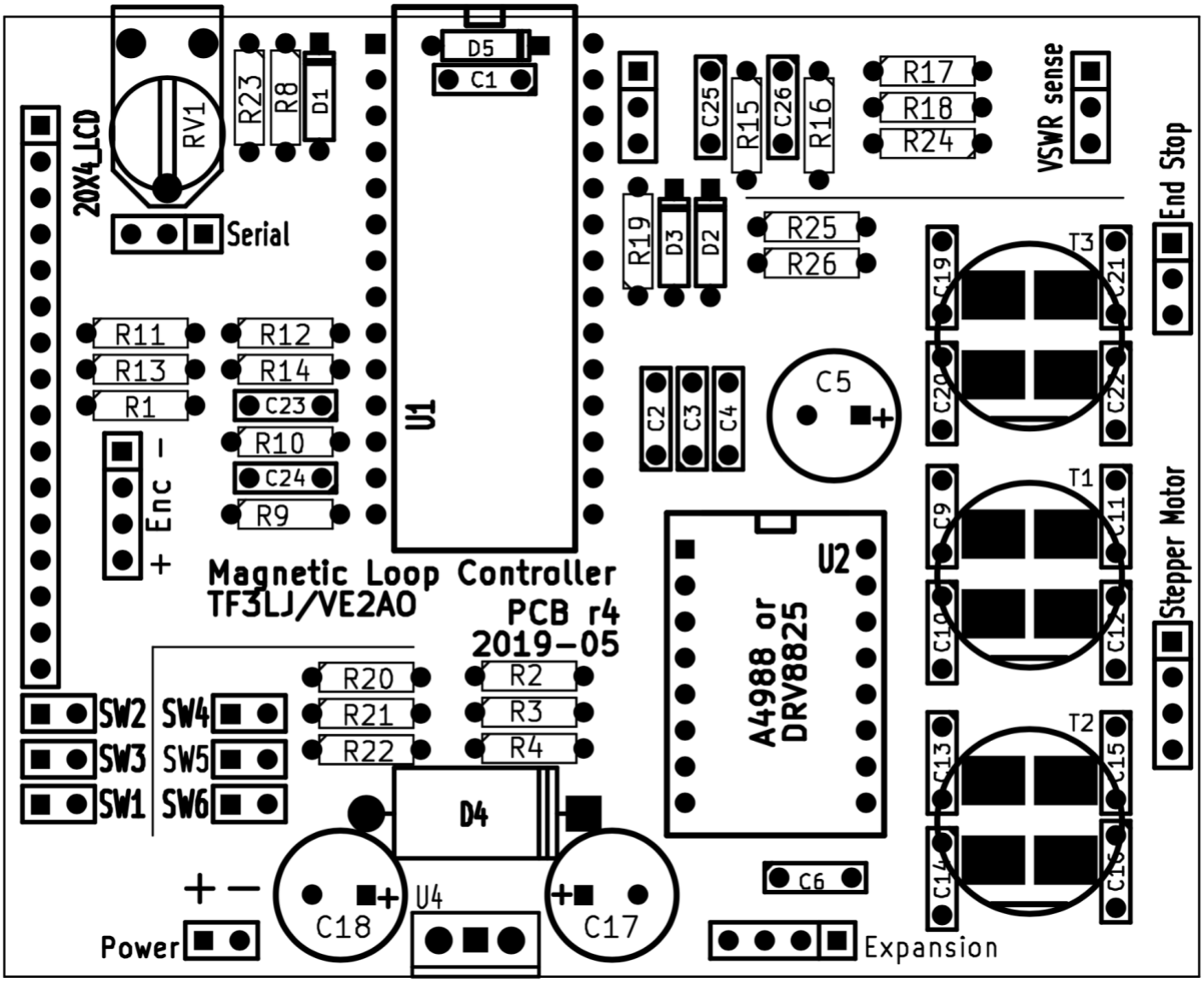


Normally the Teensy 3.2 microcontroller is power fed via the USB port. However in the Magnetic Loop Controller project the microcontroller is powered from a LM7805 voltage regulator (U4). In order to be able to connect the microcontroller USB port to a Computer, it is better to disconnect the voltage feed from the USB cable. This is done by cutting a narrow trace separating V_in from V_usb, see the picture above.

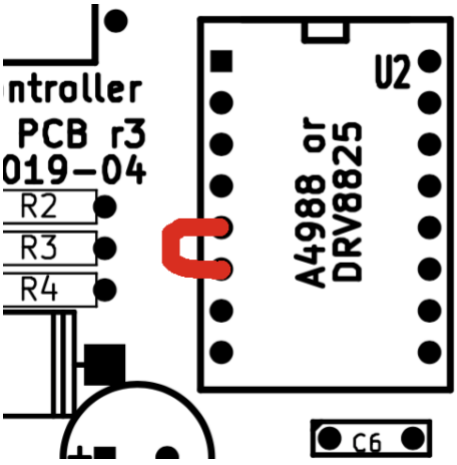
Two versions of the PCB:

There are two different versions of the PCB, Rev 3 and Rev 4. (The initial version is the Rev 3, however I will issue a Rev 4 once my stock of the Rev 3 boards is finished). The only difference between the two revisions is that Rev 3 is missing a link between pins 5 and 6 on the footprint of U2. Hence a solder link is required for the Rev 3 board, as shown on the next page.

Component placement layout PCB rev 3 and rev 4:



If you have a PCB revision 3, then a solder link is required on the underside, between pins 5 and 6, as shown here:



Assembly:

Note that there are no R5, R6, R7, C7, C8 and U3. R24 is not used.

A number of components do not need to be placed unless you plan to use the End Stop Sense:

D2, D3, R25, R26, C19-C22, T3, End Stop Sense header

These components do not have to be placed unless you are planning to use the Power / SWR Meter + SWR Autotune option:

C25, C26, R15 – R22, the two 3 pin headers around the SWR circuit.

Start by placing and soldering the resistors.

If using surface mount common mode chokes, then solder these in place before placing the capacitors. This is best done by pre-melting a dab of solder onto one of the four pads for each choke on the PCB. Next - orient the choke accurately on top of the four pads and re-melt the solder on the one pad until the choke sinks onto the pad. Now solder the three other legs in place.

If using ferrite core common mode chokes, then these should be soldered in last.

Place and solder capacitors. Observe correct polarity of C5, C17 and C18.

Place and solder the trimpot resistor RV1.

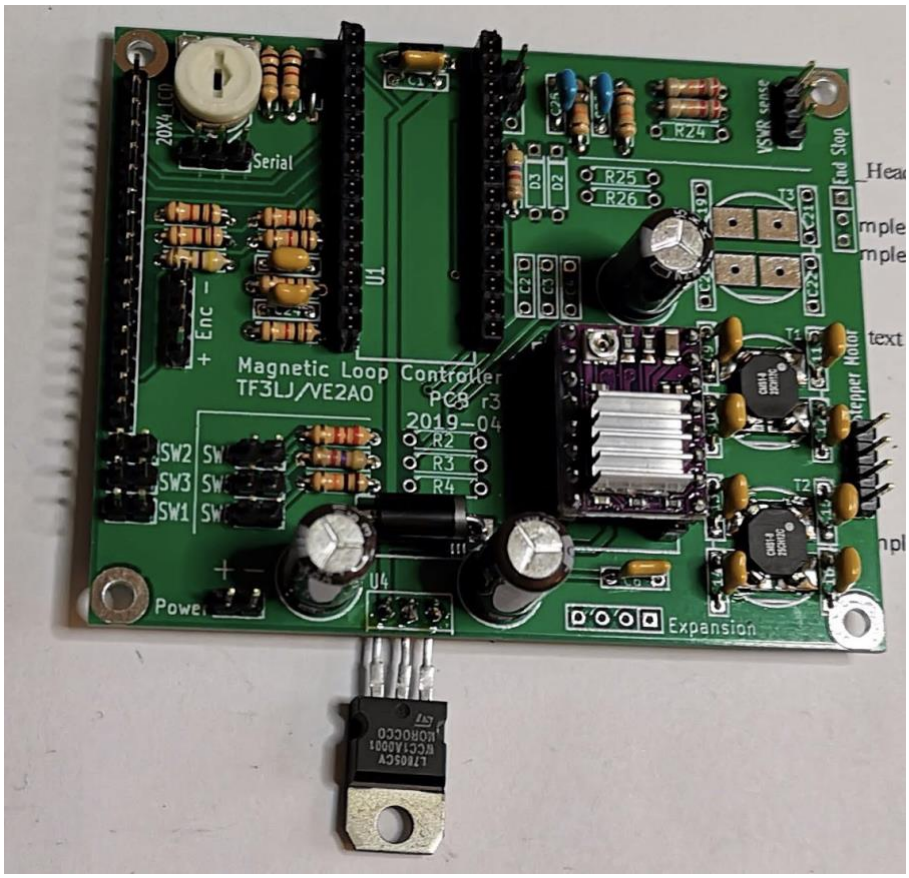
Place and solder the sockets for U1 and U2 and place and solder the PCB headers, if used.

Place and solder the diodes. Orient the diodes as indicated on the silk screen.

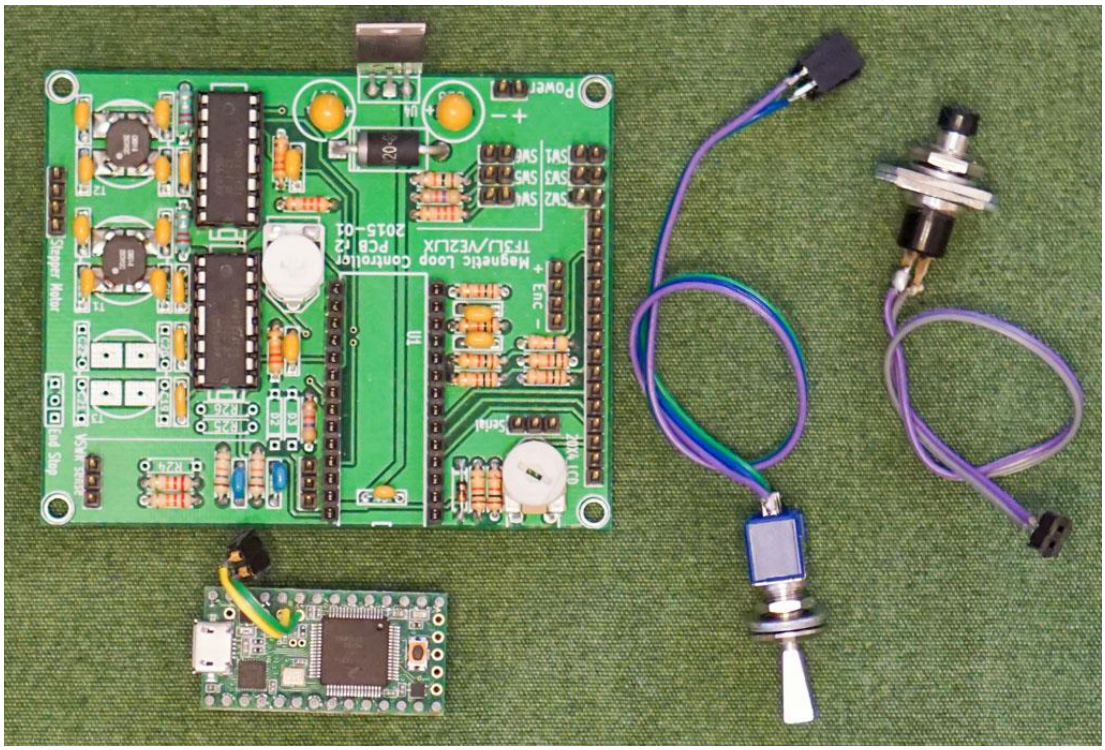
Place and solder the 5V Regulator U4. Observe that the regulator will need to be bolted to the enclosure or to a heatsink.

Finally seat U1 and U2 into their sockets.

Here is a picture of an assembled Controller. The components for the End Stop Sense have not been placed:



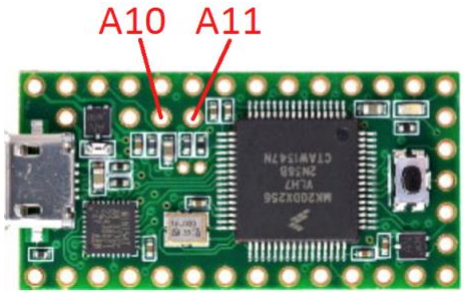
Here is a picture of an older version, utilizing two A4975 stepper motor controller chips instead of the DRV8825 carrier board:

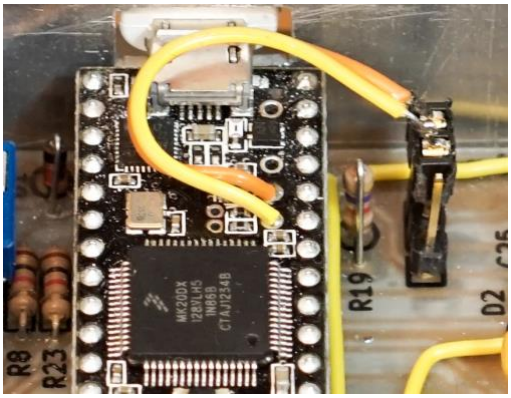


This picture shows an older version of the loop Controller, using two A4975 stepper controller chips, rather than the DRV8825. Since this picture was taken, I have modified C17 and C18 to be 100uF/63V rather than 10uF. Also, it is better to solder the regulator to the under side of the PCB, so that it can be bolted to the case for heat sinking, see pictures on pages 3 and 12.

Power / SWR meter and SWR Autotune Option

If implementing the Power/SWR meter to enable SWR based Auto-tuning, then we need to connect AD signals A10 and A11 from the Teensy 3.1 to a connector on the Controller PCB:



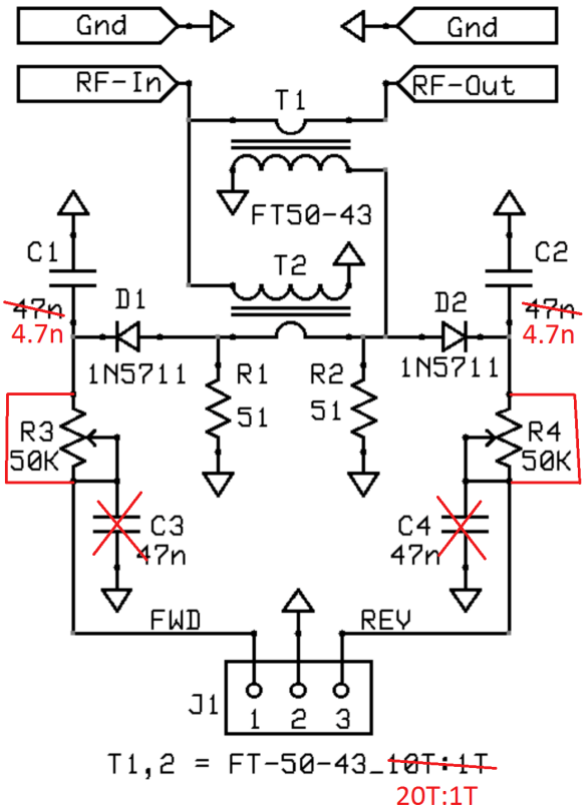


Two choices are provided for Power/SWR sensing:

Choice 1: A Tandem Match Coupler circuit

A Tandem Match coupler may be a bit more accurate than a Bruene Bridge, however it is more difficult to achieve very good isolation between the Forward and Reverse outputs, as this is dependent on how similar the two transformers are. On the other hand, this is not intended to be a super-precision instrument, the below circuit is certainly as good as the average HAM grade Power/SWR meter.

Modify a popular $9 kit, from Kits and Parts (http://www.kitsandparts.com/bridge.php)



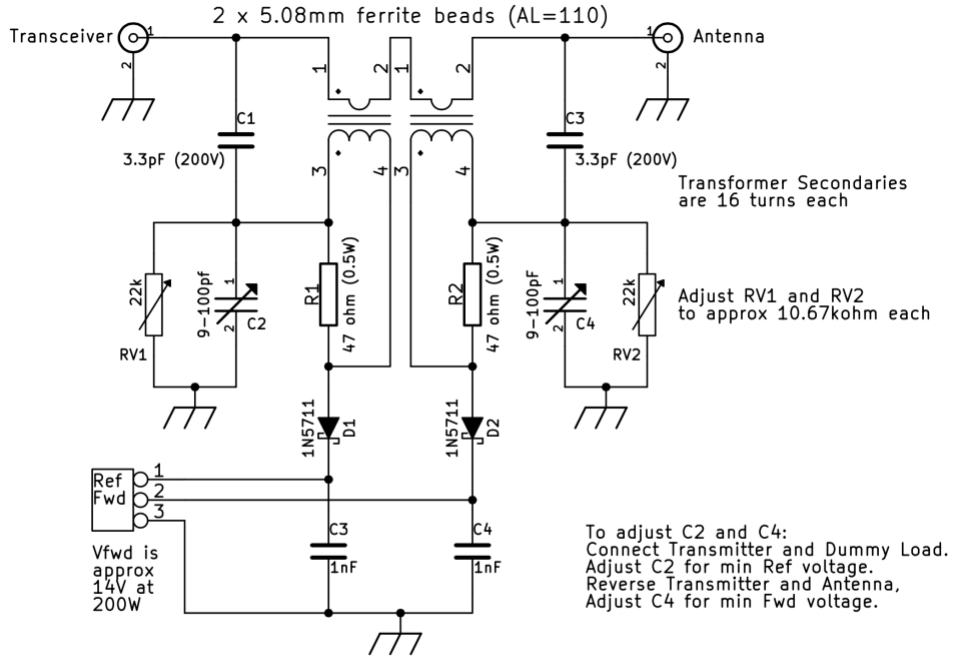
Modifications to the kit from Kits and Parts:
20 turns wound on the two ferrite cores. This will make the meter suitable for 100W (power dissipation in R1 and R2 is 0.25W max at 100W), should be able to handle 200W SSB without upgrading the resistors to 1/2W type.

R3 and R4 are shorted, C1 and C2 are exchanged to 4.7nF capacitors (example: Digikey 445-4746-ND) to speed up the response of the outputs. This is necessary for fast SWR tuning. C3 and C4 are omitted.

Note the different order of the pins at the J1 connector, when compared with the VSWR sense connector on the Controller PCB.

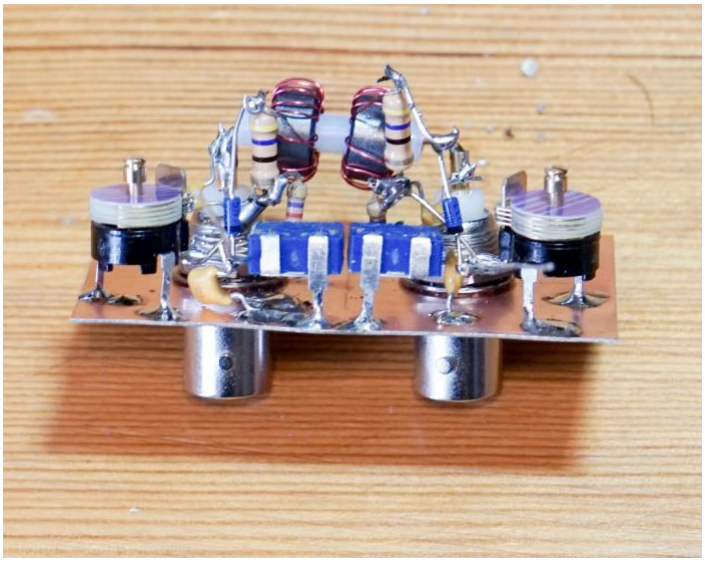
Resistor values R15, R16 and R17, R18 on Controller PCB:
For a useful range of 0.5 – 200W, R15 and R16 on the Controller PCB should be 18 kohm 1/4W (example: Digikey 18KQBK-ND) and R17, R18 should be 22 kohm 1/4W (example: Digikey 22KQBK-ND).

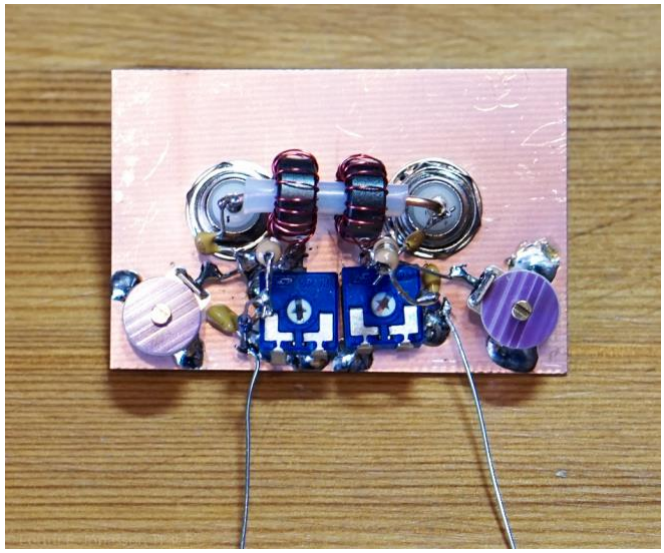
Choice 2: A double modified Bruene bridge



Modified Double Bruene bridge SWR Meter
for the Magnetic Loop Controller

2015-01-11
TF3LJ & VE2LJX





My prototype, shown on the two pictures on the previous page, is built “ugly style” using a single sided printed circuit board. The BNC connectors are 2.5 cm apart. The ferrite beads are Laird 28B0375-400 (Digikey 240-2296-ND). You could also use cores such as T37-43. Due to the higher permeability (AL value = 350) of this core, RV1 and RV2 would then have to be adjusted to 33.9 kohm.

If you look closely, you will see a couple of unexplained resistors. Those are 4.7k ohm resistors in series with the 10 kohm potentiometers I was using instead of 22k as specified on the schematic.

The main advantage of this circuit is that you balance it almost completely for a nice SWR reading of close to 1.00:1 at 50 ohm. However this needs accurate adjustment of C2 and C4.

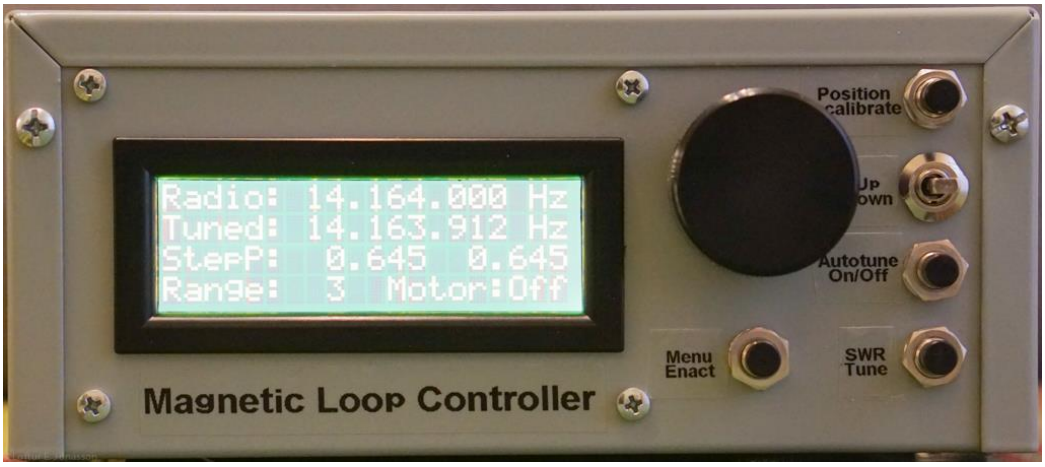
Resistor values R15, R16 and R17, R18:

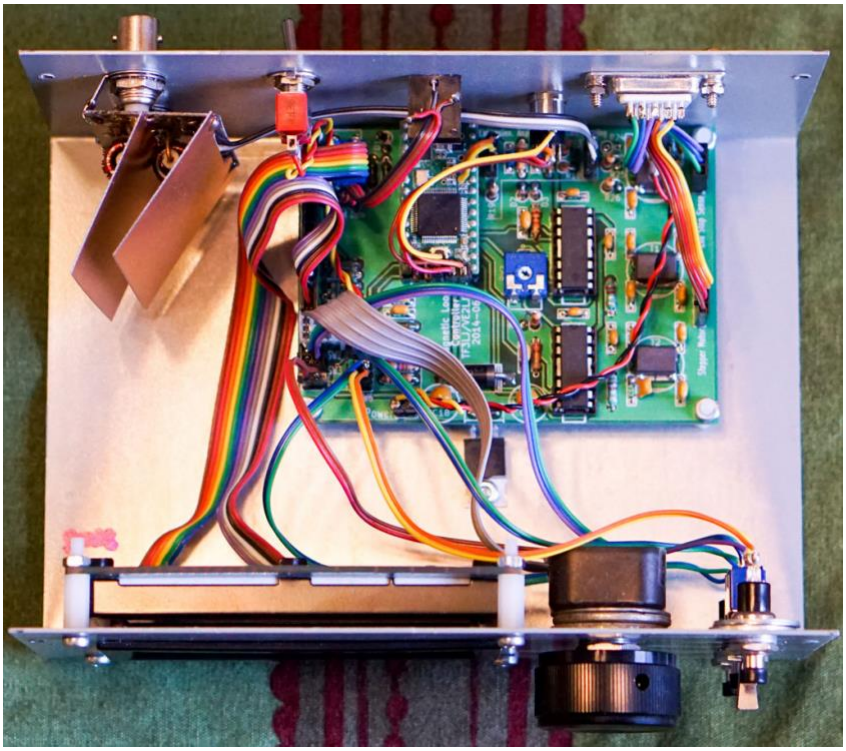
For a useful range of 0.2 – 200W, R15 and R16 on the Controller PCB should be 18 kohm 1/4W (example: Digikey 18KQBK-ND) and R17, R18 should be 68 kohm 1/4W (example: Digikey 68KQBK-ND).

Three additional Switches, SW4, SW5 and SW6:

For the SWR Autotune option, two additional Push Switches, SW4 and SW5, are required for the Front Panel.

In addition, when using this option, the Stepper Position Recalibrate function has been moved from the Menu/Enact button to a new push switch, SW6:





Note: The I2C connector shown on this picture is not used by the Magnetic Loop Controller. This was added for a different project I have been working on.

Preparing for uploading of Firmware and first tests:

To test the assembled Controller for the first time, connect the LCD and connect a 12V power source to the power connector. If the Firmware has already been uploaded to the Microcontroller, then you should see text appear on the LCD. If not, then in all likelihood a LED on the Microcontroller will start blinking in a once-per-second on-off pattern. This is the “Blinky” program that most Arduinos and clones are pre-loaded with.

The LCD needs to be connected for the Controller Firmware to run. However nothing else needs to be connected for uploading the firmware and first tests.

When powering the Controller up after programming the firmware for the first time, do not panic if nothing is seen on the LCD. Usually all that is needed is to adjust RV1 for LCD contrast.

Uploading the Firmware

Below is a step-by-step description of how to upload the firmware. Two methods are given, the first method is only applicable if you are not implementing the Power/SWR Meter Option.

1) Uploading of a precompiled HEX file.
2) Compilation of the source code, followed by automatic uploading. This is slightly more complex than method 1, but not by much.

Method 1, uploading of a precompiled HEX file

In the ML_vxxx.zip file you will find full source code and three precompiled HEX files. The HEX file you choose depends on your end-stop configuration:

1. Soft End Stops. Vacuum variable, no end-stop switches. In this case one has to take care that the stepper motor is just powerful enough to turn the capacitor but not excessively more so. The Up/Down switches will not work beyond the lowest/highest stored frequency/position and the Radio cannot tune the capacitor beyond the lowest/highest stored position. To go beyond an already "proven" range, one needs to turn the capacitor by turning the Encoder, and store new frequency/positions to extend the range. The downside of this method is that it only works if there is Frequency input available from the radio ("smart" mode).
2. Hard End Stops. Vacuum variable, end-stop switchces. All as 1) except no software "intelligence" to inhibit use of Up/Down buttons or tuning beyond an already "proven range". Can be used in "smart" mode with frequency input from radio, or "dumb" mode with no frequency input.
3. No End Stops. Butterfly capacitor. Otherwise same as 2).

The Teensy Loader application, found at the URL below is used to flash the HEX file into the Microcontroller:
http://www.pjrc.com/teensy/loader.html

Select the correct version depending on your operating system, and download. The Teensy Loader can be copied to the desktop and run with a simple click. Once Teensy Loader is running, then:

1. Extract the ML_vXXX.zip. The files will extract into a folder with the same name. You will find the three precompiled HEX files in a subfolder with the name Precompiled_HEX_Files.
2. Turn the Magnetic Loop Controller on and connect its USB port to the computer.
3. Running the Teensy Loader, click on "File" and then on "Open HEX file"
4. browse to the location where you saved the HEX file, and select it.
5. push the little button on the Teensy Microcontroller inside the Magnetic Loop Controller.
6. click on "Operation" and select "Program". The new firmware is now uploaded.

Method 2, tailoring your own options and settings and compiling the firmware

This method is not much more complicated than Method 1 and gives you the opportunity to tailor some of the functionalities of the firmware to your own liking.

First, download the Arduino environment, including the Teensyduino extension available here:
https://www.pjrc.com/teensy/teensyduino.html
Make sure to select all Libraries when you install the Teensyduino extensions, the source code uses several of those.

Once you have the Arduino environment installed, then:

1. Extract the ML_vXXX.zip. The files will extract into a folder with the same name. You will find a number of files with the ending .ino and .h in this folder. These files are the source code.

2. Browse into the folder and double click on the file called ML_vXXX.ino, this is the "master" file. The Arduino software environment should now open a window displaying the source code. If Arduino opened another window as well, called sketch something, just close that one.

   The Source code window will have 9 tabs. The first tab is the "master" file, the second tab is the ML.h file. In the ML.h file you can change or tailor many things. Note that any text written after "//" is regarded as comments and is ignored.

3. Start by verifying that the line which contains the text #define ENDSTOP_OPT has the end-stop option selected which best matches your setup. I would recommend you also browse down to the line starting #define DISP_CALLSIGN. Change the call signs indicated here to your own and fill with spaces after it to fill up to 20 characters in total. If you will be using the dual antenna option, then set the Antenna Switchover frequency using #define ANT_CHANGEOVER.

   If you are implementing the Power/SWR meter and Autotune Option, then here you will also need to modify the following #define settings in the ML.h file:

To turn on the Power/SWR and SWR Autotune function:	#define PSWR_AUTOTUNE 1
	#define BRIDGE_COUPLING 20.0
Define the correct Bridge coupling, typically 20 or 12.04:	#define METER_CAL 1.08
Define the correct resistor values, R15,R16:	#define VALUE_R15 18000
and R17, R18, typically 22000 or 68000:	#define VALUE_R17 22000

   If on the other hand implementing a 2x AD8307 Power Meter, then you would modify the #define AD8307_INSTALLED accordingly.

4. In the Arduino Tools tab, double check that Teensy 3.1/3.2 is selected as a "Board:"

5. Turn the Magnetic Loop Controller on and connect its USB port to the computer (not time critical, can be done at any time).

6. In the top left corner of the Arduino window, click on either the tick sign or the right arrow. Now everything should hopefully compile without errors and automatically load itself to the Controller. No worries, if something went wrong, then nothing will be uploaded.

That's it.

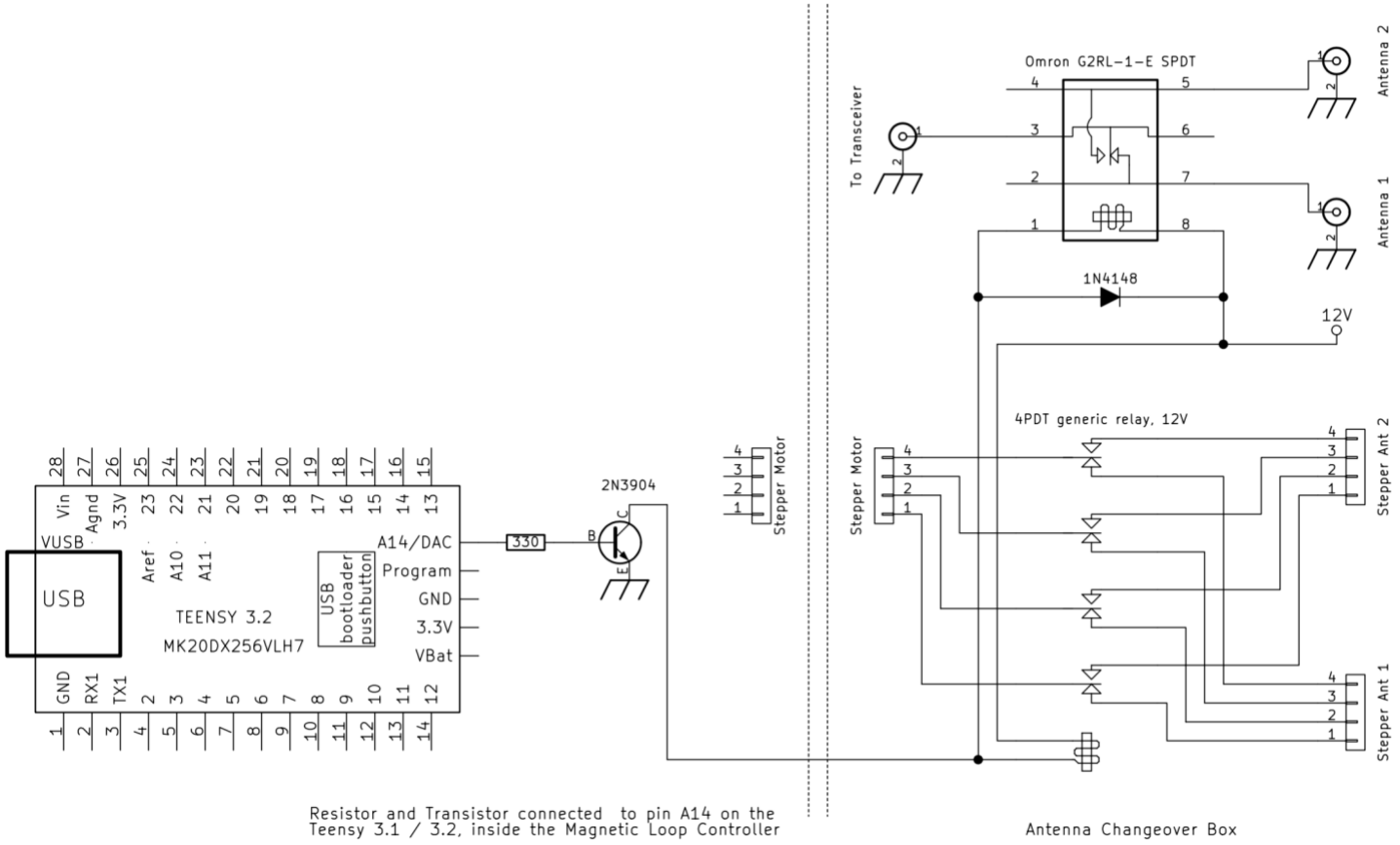
Addendum 1 – Managing two Antennas with one Controller.

The firmware has the capability to manage two antennas. This works as follows:

A Changeover frequency is defined in ML.h, see:

```
//------------------------------------------------------------------------------
// Enable management of two antennas - define changeover frequency.
// If not using this option, then set as 0, else set changeover frequency in Hz
//#define ANT_CHANGEOVER     22000000  // 0 or changeover frequency in Hz
#define ANT_CHANGEOVER     0    // 0 or changeover frequency in Hz
```

A signal to select between the two antennas is available at pin A14 on the Teensy. This pin will be LOW when at frequencies below the changeover frequency and HIGH (3.3V) when at frequencies above the changeover frequency.



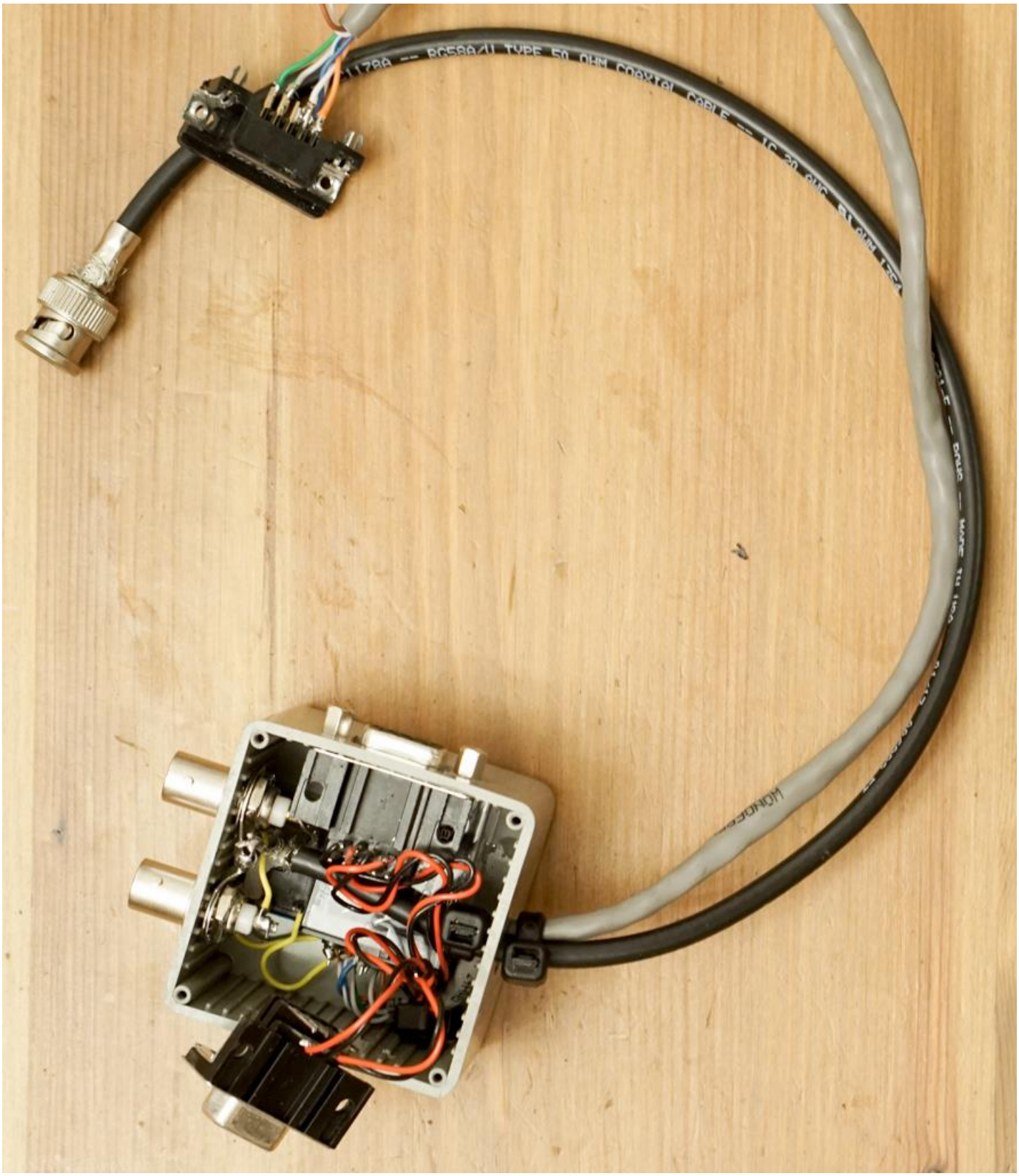
In order to select between the two antennas, the changeover signal at pin A14 is used to control a relay, which transfers the Stepper Motor signals (and endstop signals if used) between the two antennas. The changeover signal is only capable of driving a couple of mA, hence the transistor, as shown on the schematic above. The transistor is a generic small signal NPN, capable of switching a couple of hundred mA. The resistor is 330 ohm.

The relay can be any generic 4 or 6 pole, dual throw, with a 12V coil.

Another relay in parallel is used to switch the RF between the two antennas. The relay indicated on the schematic is an Omron G2RL-1-E (Digikey Z2834-ND). This relay, while inexpensive, is popular in HAM projects and is capable of switching 1.5kW at HF frequencies.

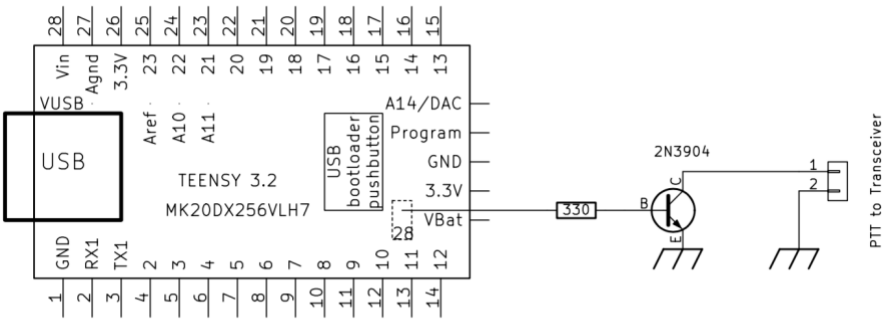
This feature works as follows:
Two separate sets of stepper tracking parameters, automatically selected by the ANT_CHANGEOVER frequency. Whenever a frequency change across ANT_CHANGEOVER is detected, then the current settings are stored in eeprom before switching over to the new settings. This effectively makes the controller behave as two separate controllers, for two non-overlapping frequency ranges. Frq/Pos memories are divided between the two frequency ranges based on the ANT_CHANGEOVER setting. In other words, the two antennas can share the 200 frq/pos memories between them at any ratio.

Here is a picture of my own slipshod build of an Antenna Changeover box. This picture was taken before fastening the second stepper output connector and closing the box. All components used came out of my own junk box. Two generic relays were crammed into an old plastic box repurposed from a previous project. Those can be seen, just barely. The RF relay is the one directly below the two BNC connectors. Everything fit with less than a millimeter to spare. Works well and doesn't catch fire at 200W ☺



Addendum 2 – PTT signal output, useful for some older transceivers such as ICOM IC-706.

Solder pad 28, accessible on the underside of the Teensy provides PTT signalling from the Teensy to the Transceiver, intended for the SWR tune cycle when using older generation Transceivers such as ICOM IC 706. The below sketch shows how this would be used:



Resistor and Transistor connected to pad 28 on the underside of the Teensy 3.1 / 3.2, inside the Magnetic Loop Controller