RF Attenuator - Version 2.0

Table of Contents

Introduction

Recently, I've been playing with SDRs and various preamplifiers. In some cases, the signal levels coming out of the preamplifiers are much higher than needed for the SDR and applications such as WSJT-X, so I decided I needed some attenuation. Not having an attenuator, I decided to build one!

Version 1.0 was designed to use a pair of PE4312 attenuator chips in a series arrangement which would have allowed anywhere from zero to 63 dB of attenuation in half dB steps. Unfortunately, after several attempts to actually build version 1.0, I was unable to properly mount the PE4312 chips on the PCB (my SMD soldering skills leave a lot to be desired).

Thus, plan 'B' is to use one of the pre-built PE4302 modules available on eBay and settle for only 31.5 dB of attenuation. I didn't include a link as they change too frequently. As all the preamps I've tried generally have gains in the 15 to 25dB range, this unit should provide adequate compensation when needed.

The attenuator is controlled by an Arduino Nano processor (clones are available and cheaper).

This is not intended to be a well calibrated laboratory type device; however it can be calibrated fairly accurately.

The Hardware

The major components of the hardware are one of the PE4302 modules available on eBay, an Arduino Nano processor, a rotary encoder and a 128 x 32 OLED display.

Theory of Operation

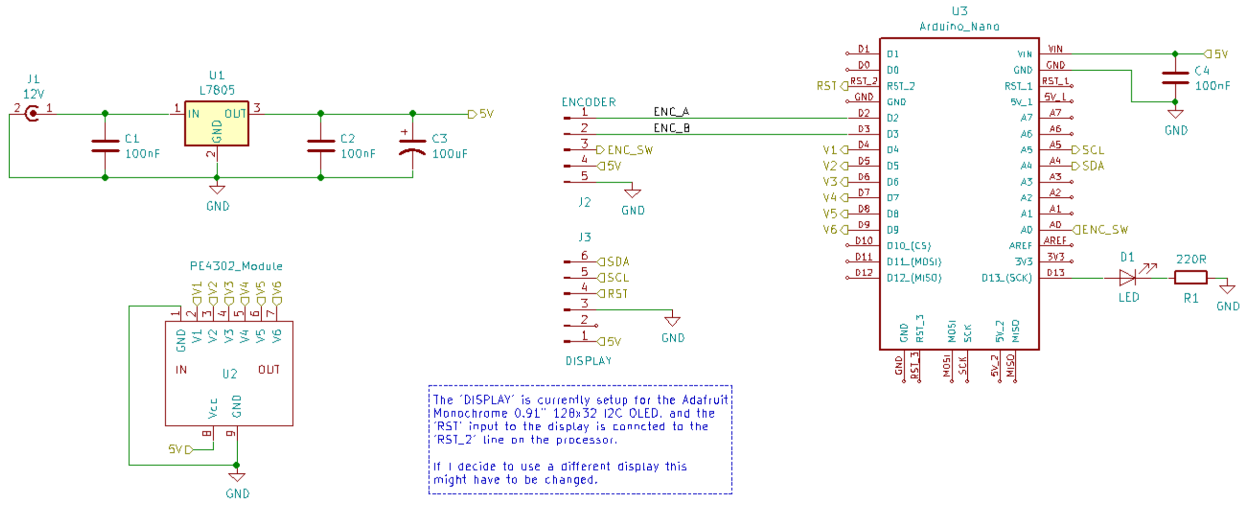
The PE4302 module is capable of providing zero to 31.5 dB of attenuation in 0.5 dB steps.

The amount of attenuation is controlled by the encoder; each click will add or subtract 0.5 dB of attenuation. The amount of attenuation is displayed on the front panel display.

As there is a slight amount of attenuation even when the module is set for no attenuation, there is a provision in the software to account for the inherent loss in the attenuation value displayed on the screen (See the *Specifications* section). The amount of inherent attenuation is also frequency dependent.

The unit can be powered via a 9 to 12VDC source or via the USB connector on the processor.

Here’s the schematic:

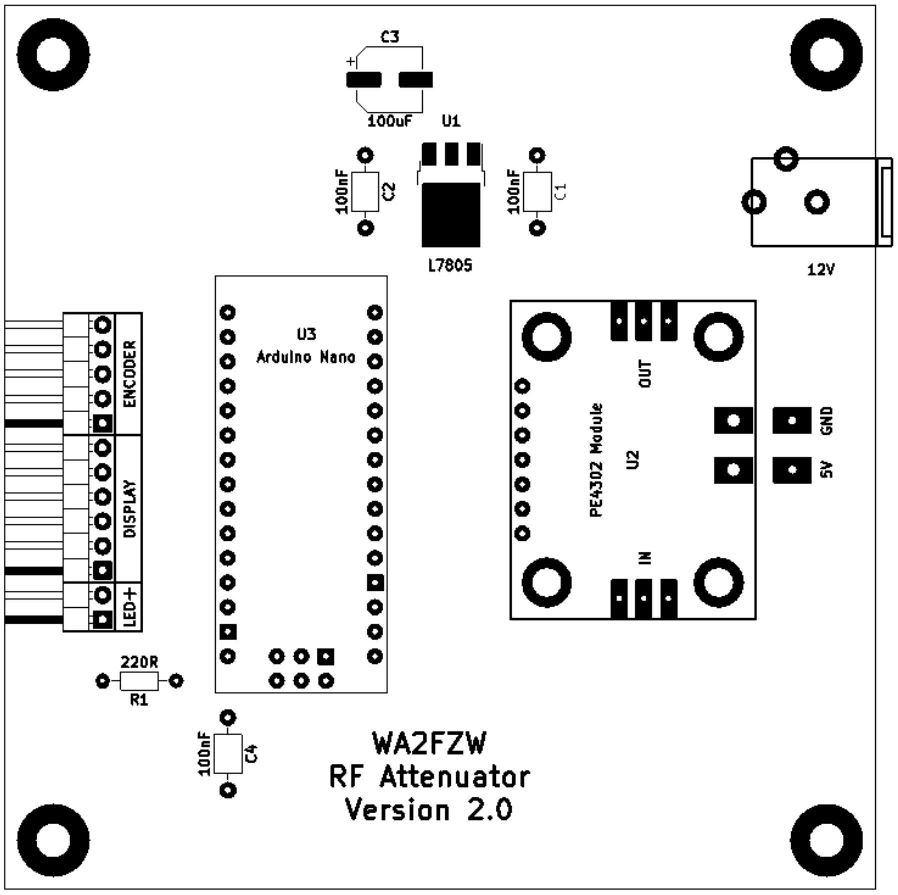


The ‘DISPLAY’ is currently setup for the Adafruit Monochrome 0.91" 128x32 I2C OLED. and the ‘RST’ input to the display is conncted to the ‘RST_2’ line on the processor.

If I decide to use a different display this might have to be changed.

The Printed Circuit Board

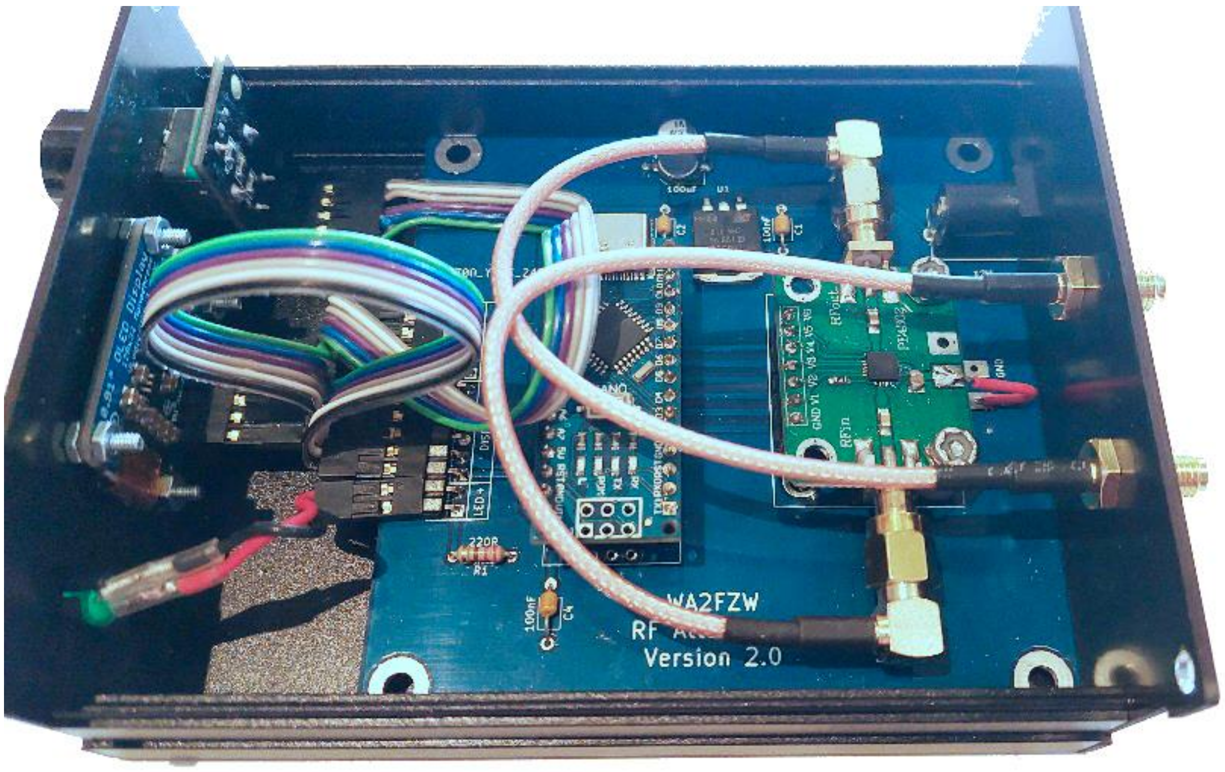
Here's the layout of the PCB:



The PE4302 module comes with female SMA connectors installed; short cables between those and the rear panel will provide the input and output connections.

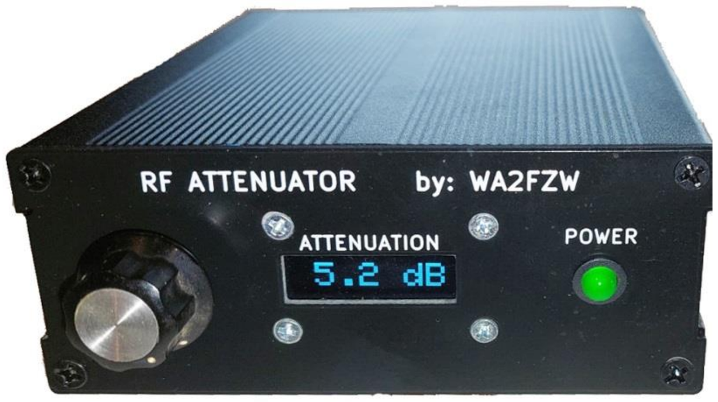
I used SMA female connectors on the rear panel; however, one could enlarge the holes and use BNC connectors as well.

Here’s the finished PCB mounted in the enclosure:



Front Panel Controls and Indicators

Here's what the front panel looks like:



When on, the 'POWER' LED indicates that the processor is running normally and the current attenuation setting will be displayed on the screen. Turning the encoder clockwise will increase the attenuation; moving it counter-clockwise will decrease the attenuation. If yours works backwards, you can fix that in the *Software*.

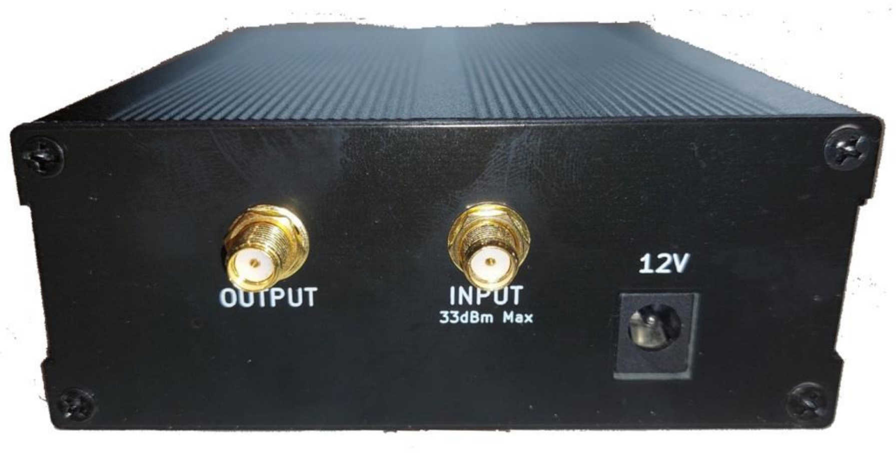
Operating the push-button switch built into the encoder will reset the attenuation to the default value (see the *User Customization* section).

Sometimes, for some reason, the 'POWER' LED may flash rapidly when the unit is first powered up then it should come on solid. If it continues to flash, it is an indication that the processor can't connect to the display for some reason.

Notice that as stated in the *Introduction*, the PE4302 works in 0.5dB steps, however, here the display shows 5.2dB. There is a software setting (see the *User Customization* section) that can be set to compensate for the inherent loss when the module is set to 0.0dB. In the case of this particular unit, the error is 1.2dB and the PE4302 is actually set to 4.0dB.

Rear Panel Connections

Here's what the rear panel looks like:



The 'INPUT' and 'OUTPUT' connectors are SMA females and the '12V' connector accepts a 2.1mm coaxial plug.

In the Gerber files for the rear panel that are included in the distribution, I moved the input and output labels down lower should someone want to enlarge the holes and use BNC connectors.

Some Construction Notes

Front & Rear Panels:

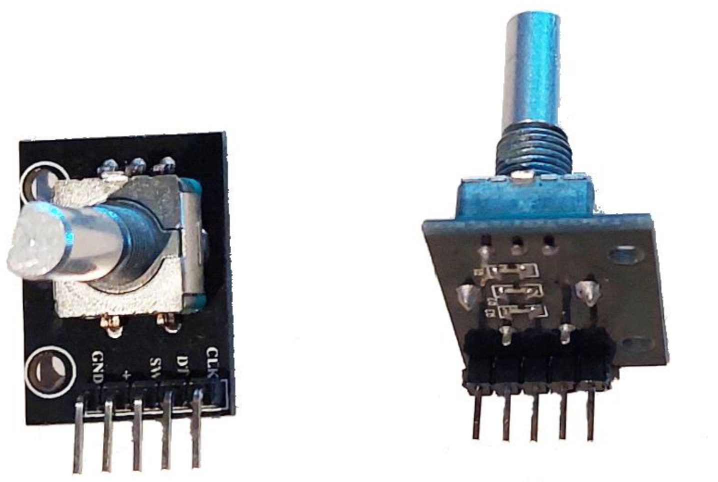
Both panels and the main PCB are designed to fit this specific extruded aluminum enclosure from Amazon (you might also find them on eBay). The front and rear panels are actually printed circuit boards; the Gerber files to fabricate them are included in the distribution package. Of course you can use a different enclosure.

You may notice that the edges of the front and back panel PCBs are more gray than black. I found that going over the edges with a black Sharpie makes for a much nicer look.

The screws that come with the enclosure are the countersunk type. In order to use those neatly, you'll need to modify the mounting holes in the panels with a countersink bit or use round head 3mm screws.

Encoder:

The KY-40 encoders that I have come with angled header pins attached to the front side of the PCB. They won’t fit in the enclosure. You need to remove the angled header pins and install a straight header on the rear side of the encoder PCB as shown here:

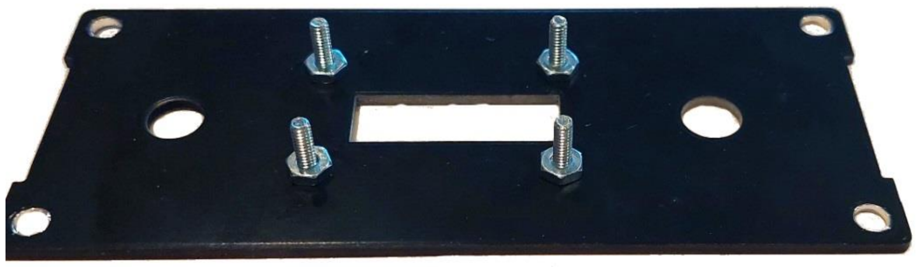


Or, you can hardwire the encoder to the PCB.

Display:

The mounting holes in the Adafruit display that I used accept either an American #2 screw or a 2mm screw. I used ones with countersink type heads and modified the mounting holes accordingly.

The display needs to be mounted with spacers. I found that the #2 nuts were exactly the correct thickness, so here’s what the mounting arrangement looks like:



PE4302 Module:

The Chinese PE4302 modules come with a 7-pin vertical header mounted on the top side of the PCB. The attenuator PCB is designed such that the module mounts top side up, so I moved the pin header to the bottom side of the module.

Connection Between the PCB and Front Panel:

If you look at the picture above showing the PCB installed in the enclosure you might notice that the Dupont connectors for the encoder and display are jammed right up against the connectors on the PCB.

It would be better to hardwire the cables to the PCB and only use the Dupont connectors on the encoder and display.

Software

Required Libraries

The software is fairly simple thanks to the availability of Arduino libraries for the display, and the encoder.

The non-standard libraries required are:

- Adafruit_GFX
- Adafruit_SSD1306
- Rotary (from https://github.com/brianlow/Rotary)

The first two can be installed using the ‘Manage Libraries…’ selection under the ‘Tools’ menu in the Arduino IDE. The ‘Rotary’ library should be obtained from the link provided; there are other similar libraries, but I can’t say whether they would work or not.

Note, I’m still using version 1.18.xx of the IDE; I don’t expect there would be any issues using the latest version though.

User Customization

There are three definitions in the Attenuator_V2.0.ino file that you might want to change:

- The value assigned to ‘ATT_INIT’ specifies the initial attenuation setting when the unit is first powered up or when the encoder push-button switch is operated. But note, until the program actually starts running, the amount of attenuation is undefined.

  This is useful if you know that your application normally requires a specific amount of attenuation +/- a few dB.

- The value assigned to ‘ATT_OFFSET’ can be used to compensate for the small attenuation that will exist even when the unit is set for zero attenuation. The amount may vary with frequency. Refer to the *Specifications* section which shows the amount of inherent attenuation for my unit across a wide frequency range.
- If you set the definition of ‘DEBUG’ to ‘true’, you will get status messages on the serial monitor whenever the attenuation changes.

If you change anything else, you’re on your own!

Serial Monitor Messages

If you have the processor connected to your computer and have the Arduino IDE’s Serial Monitor running, there are 4 messages that could be displayed:

- ‘WA2FZW RF Attenuator - Version 2.0’

  Indicates that the program is up and running
- ‘SSD1306 allocation failed’

  Indicates that the processor could not to connect to the display for some reason. The ‘POWER’ LED will also be flashing rapidly.

If the definition of ‘DEBUG’ is set to true:

- ‘PE4302 Attenuation = nn.n dB, Binary = nnnnnn’
  Shows the actual setting of the PE4302 module in decimal and binary.
- ‘Display Attenuation = nn.nn’
  Shows the actual attenuation that should be on the display. The value should be the actual setting of the PE4302 module plus the value assigned to ‘ATT_OFFSET’.

Functions

Here are some brief descriptions of what the various functions in the software do. There are plenty of comments in the actual code that have more detailed explanations of the logic in each.

The functions are listed in the order in which they appear in the code.

setup:

This is the first function executed in any Arduino application. It sets up the GPIO pins and creates and initializes the encoder and display objects. It then sets the attenuator to the amount specified by the ‘ATT_INIT’ symbol and displays the setting (adjusted by ‘ATT_OFFSET’ value).

There are several lines of code at the end of the *setup* function that are commented out. These can be un-commented when testing a PE4302 module or when calibrating one as shown in the *Specifications* section.

loop:

Again, this is a standard function on all Arduino programs. It is a forever loop that never ends.

It really does only two things. On each pass, it checks to see if the encoder was used to change the attenuation and if so updates the PE4302 and the display.

It also checks to see if the encoder pushbutton was pushed and if so, it resets the attenuation to the value of ‘ATT_INIT’.

ReadEncoder:

The 'ReadEncoder' function is called via the Arduino's interrupt mechanism, and is never invoked by any other means. Based on which way the encoder moved, we either increment or decrement the 'attenuation' by 0.5dB per click.

SetAttenuation:

This function breaks the current value stored in the ‘attenuation’ variable into individual bits which are sent to the V1 through V6 pins of the attenuator module.

ShowAttenuation:

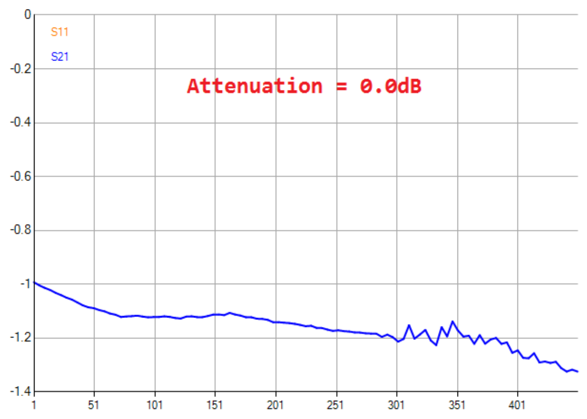
This function displays the current attenuation setting on the display. The value shown is the current setting of the PE4302 module plus the value of the ‘ATT_OFFSET’ symbol.

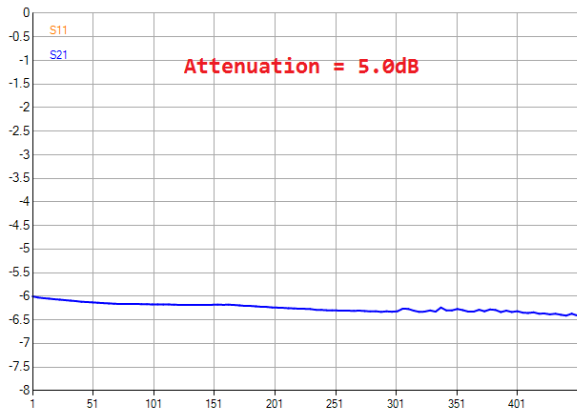
BlinkLED:

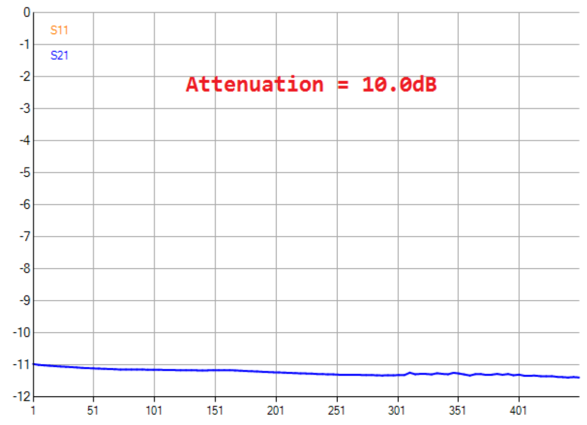
This function blinks the Power LED to indicate that some error has occurred. So far, the only error condition is when the software cannot initialize the OLED display. If that happens, the LED will blink forever.

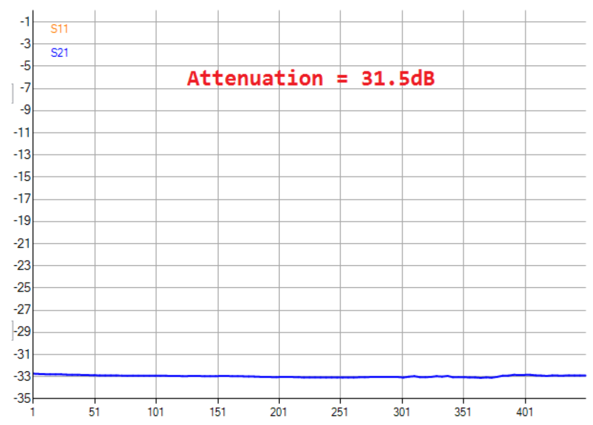
Specifications

The attenuation measurements shown here were done using a calibrated NanoVNA. The following graphs show the actual attenuation for the particular PE4302 module in my unit from 1 to 450 MHz:









Note that the response curves will vary slightly from module to module, thus if accuracy is semi-critical for your particular application, you should measure your specific module. For this particular module, I selected a value of ‘12’ (1.2 x 10) for the ‘ATT_OFFSET’ symbol. That value is added to the attenuation shown on the display.

The PE4302 Datasheet specifies a typical insertion loss of 1.5 dB and a maximum insertion loss of 1.75 dB. The datasheet also has graphs showing the expected variations with frequency.

Note that the device is not intended for use with high power signals. The maximum input level per the datasheet referenced above is 30 dBm (a bit under 1 watt). Note the rear panel is labeled ‘33 dBm Max’ (a bit less than 2 watts), which was the value given in a different version of the datasheet that I had previously saved on my computer.

As the chips on the Chinese modules probably aren’t genuine ones, I wouldn’t recommend coming even close to those levels!

Suggestion Box

I welcome any suggestions for further improvements. Please feel free to email me at WA2FZW@ARRL.net.

Bill of Materials

Here is a list of the parts you will need and in many cases, links to where you can get the less common parts.

The PCB		Gerber files are available on Github.
Front & Back Panels		Gerber files are available on Github.
Enclosure		Extruded aluminum from Amazon. The Gerber files for the front and back panels assume this specific enclosure.
Encoder	KY-40	These are available from many sources on eBay.
R1	220 Ohm 1/4W	
C1, C2 & C4	100nF	
C3	100uF Electrolytic	The PCB is designed for an SMD component due to the height of the enclosure. The footprint specification is: 6.3 x 7.7.
U1	L7805 Regulator	Available from Mouser (SMD part)
U2	PE4302 Module	Available from many sources (mostly Chinese) on eBay.
U3	Arduino Nano Processor	I've been using cheaper clones available from Amazon.
D1	General purpose 20 mA LED	Your choice of color. If yours requires a different current, you may need to adjust the value of R1, but note that the Arduino GPIO pins are rated for 40 mA maximum.
J3	2.1mm Coaxial power receptacle	Available from Amazon

J3, J4 & LED Connector	1 x 13 Male 2.54mm horizontal pin header	The PCB is designed so that a single 13 pin header can be used for all three connections.
SMA Male right angle to SMA female bulkhead jumper cable	2 Required	Available from Amazon