Michael Foerster, WØIH / mchllfrstr@gmail.com

A Programming-Free Automatic Arduino Antenna Switch

When I'm away from home, I often operate my Elecraft K3 station remotely — either from my phone or a computer. I created my first Arduino project to read the band data lines from the ACC connector on the Elecraft K3 to automatically change my antennas through a remote antenna switch at the base of my tower (see Figure 3). I found that Yaesu radios use a similar band selection, using four band data lines.

I built another similar antenna switch for our radio club's (Winona Amateur Radio Club) remote station, which employs an Icom IC-7300 (see Figure 2). Most Icom radios generate an analog voltage varying from 1 to 8 V, depending on the band; this is pin 5 of the ACC socket. It should be noted that the Icom band voltage does not differentiate between 20 and 17 meters, or between 12 and 10 meters.

I've been running an automatic antenna switch with my K3 station for about 9 years and have found it to be reliable, even at near legal-limit power levels. It is also very useful when operating a contest not to have to worry about selecting your antenna when changing bands.

The design I'll describe in this article is somewhat simpler than what I have in my home station in that this design does not include a display (see Figure 3). It does, however, have a beeper module, which announces status, errors, and other information in CW. There are push buttons for manual antenna control and a 10-minute IDer. These buttons are not required, but if you plan to mount the case within easy reach, they can be handy.

No Programming Required

This article and code have been written so that someone lacking programming experience can successfully complete this project. Only three simple things need to be modified from the code that I provide.

- Select the radio type — Icom or Elecraft/Yaesu.
- Select which antenna is enabled for each band (160 – 6 meters).

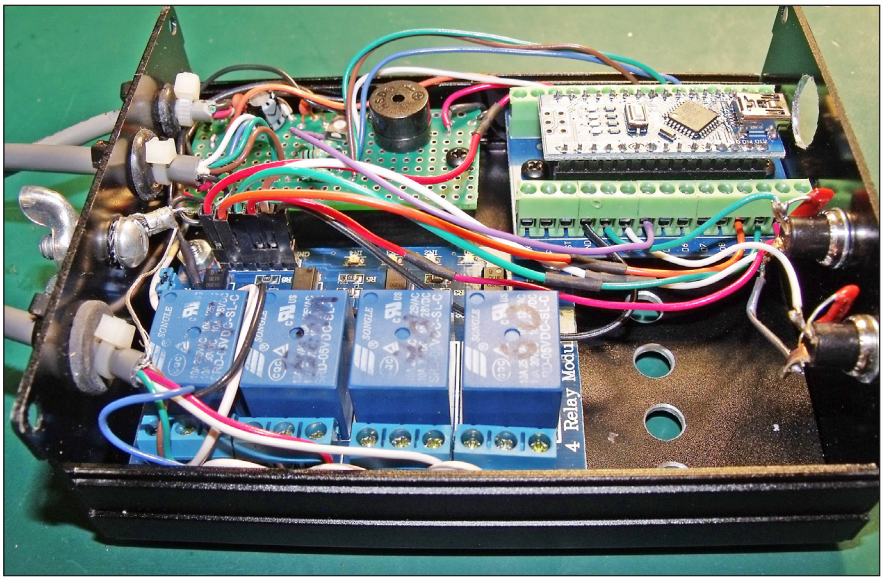

Figure 1 — Example of a four-channel antenna switch. The Arduino is on the upper right and mounted on a screw terminal adapter. The activator relays are at the lower left.

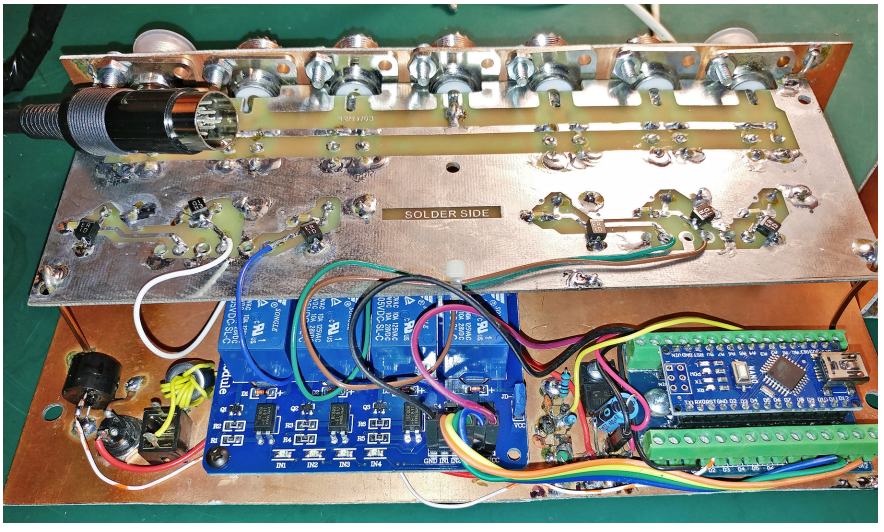

Figure 2 — An Icom combination of the antenna switch and the Arduino, mounted in the same case using four relays. This is the Icom version our club station uses, albeit a bit crude. The copper case just to the left covers the Arduino to reduce noise. The Icom ACC connector is visible at the upper left. Optional buttons were not used for this version. This six-position relay switch is from the April 2005 *QST* article, A Low-Cost Remote Antenna Switch, by Bill Smith, KO4NR.

♦ Indicate how many antennas you are switching (2 – 8).

You will need to install the Arduino integrated development environment (IDE) to allow you to edit these three items and upload the code into your Arduino. I've tried to make the instructions for this process as simple as possible.

Antenna Field Hardware

A remote antenna switch allows you to run a single coax to your antenna field and switch to up to eight different antennas. Many of the remote switches available (most as kits from eBay) use inexpensive high-current relays to switch the output power up to the full legal limit quite reliably (see Figure 4). Most available remote antenna switches ground the antenna connections when not in use.

Most kits do not come with SO-239 connectors. Be careful when ordering on eBay; some are extremely poor quality. You need to find SO-239 that have a fairly long threaded barrel, to allow the connector to go through the cover and still leave sufficient threads to match the PL-259 connector.

I typically use CAT5 cable to supply power to the remote relay box. The relays require 40 mA, so a run of even up to 100 feet should work satisfactorily. You can use a normal network connector in the shack, but I would strongly suggest a waterproof connector for outdoor connections (see Figure 5). Keep in mind that the Arduino circuit can easily drive many existing commercial remote antenna switches in parallel with their existing shack switch.

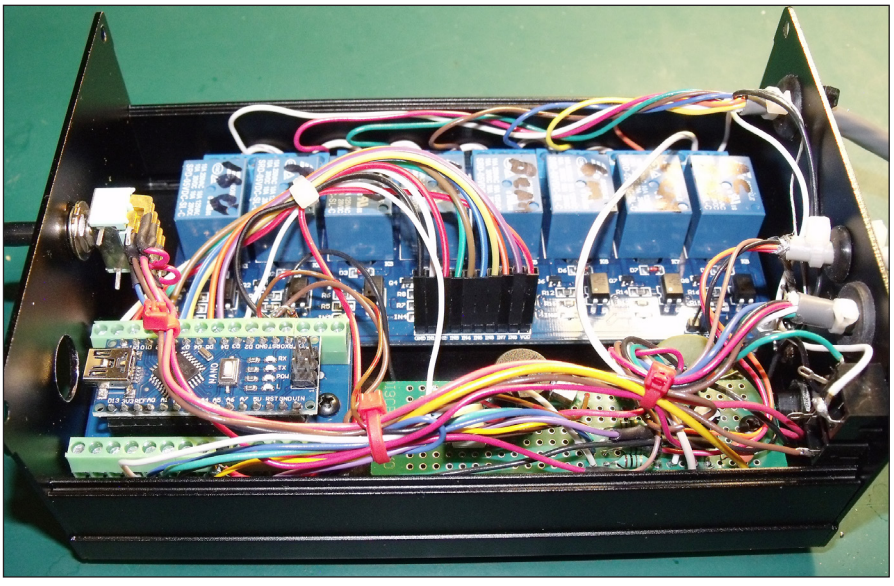

Figure 3 — Antenna switch with an 8th relay that toggles 12 V to enable a 120 V ac SSR to turn on ac power to my rotator, powered speakers, and power supply.

Arduino as a Switching Device

This project uses an Arduino Nano, which is quite small. Arduino clones are available for about $6 within the

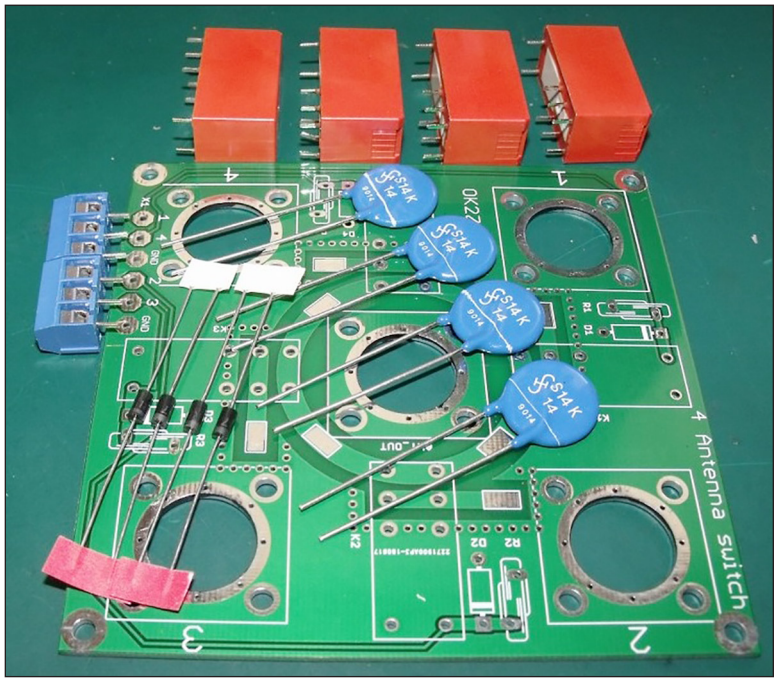

Figure 4 — Antenna switch as a typical relay kit. Normally, these kits do not include SO-239 connectors.

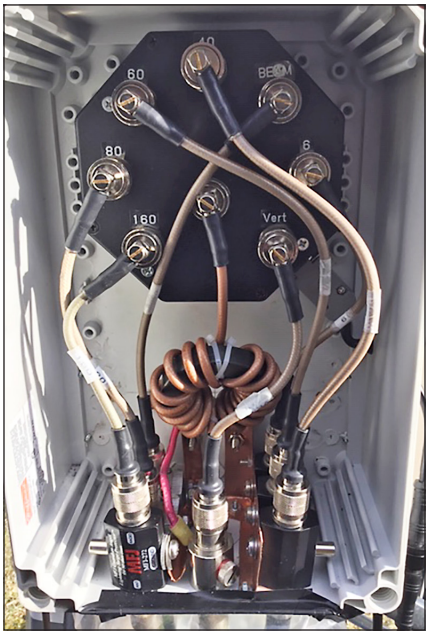

Figure 5 — Tower band switch mounted on my tower. Note the common-mode choke mounted in the output line to the shack and the lightning arrestors at the bottom for each antenna input.

US; a genuine Arduino Nano will run about $20. The only downside to the Arduino clone is that you need to load the CH340 chip driver for the USB interface in your computer before you connect the Nano to your PC.

Make sure that the Nano that you purchase has a USB connector. Less-expensive "Pro Mini" versions that are available require adapters to program.

A 5-V relay board is required, along with the Arduino as an interface between the Arduino outputs to drive the external 12-V relays. You cannot turn on the remote 12-V relays directly from the Arduino. These 5-V relay boards are available on eBay that interface with the Arduino. They cost about $1 per relay and are available in up to 16-relay versions. Power for the Arduino can be taken from the radio's 12-V accessory jack, so it will turn on when the radio is powered on. Current draw from the 12-V source is less than 100 mA. Figure 6 depicts a diagram for a multi-radio band switch. The circuit can be built using any number of relays — up to 8 as shown and depending on your individual needs. The unused Arduino relay outputs may be left unconnected.

The internal 5-V regulator on the Arduino cannot handle the heat of the higher voltage, so an LM7807 (or LM7808 or LM7809) voltage regulator is required to reduce the voltage into the Arduino from the 13.8 volt to the "V_{IN}" input.

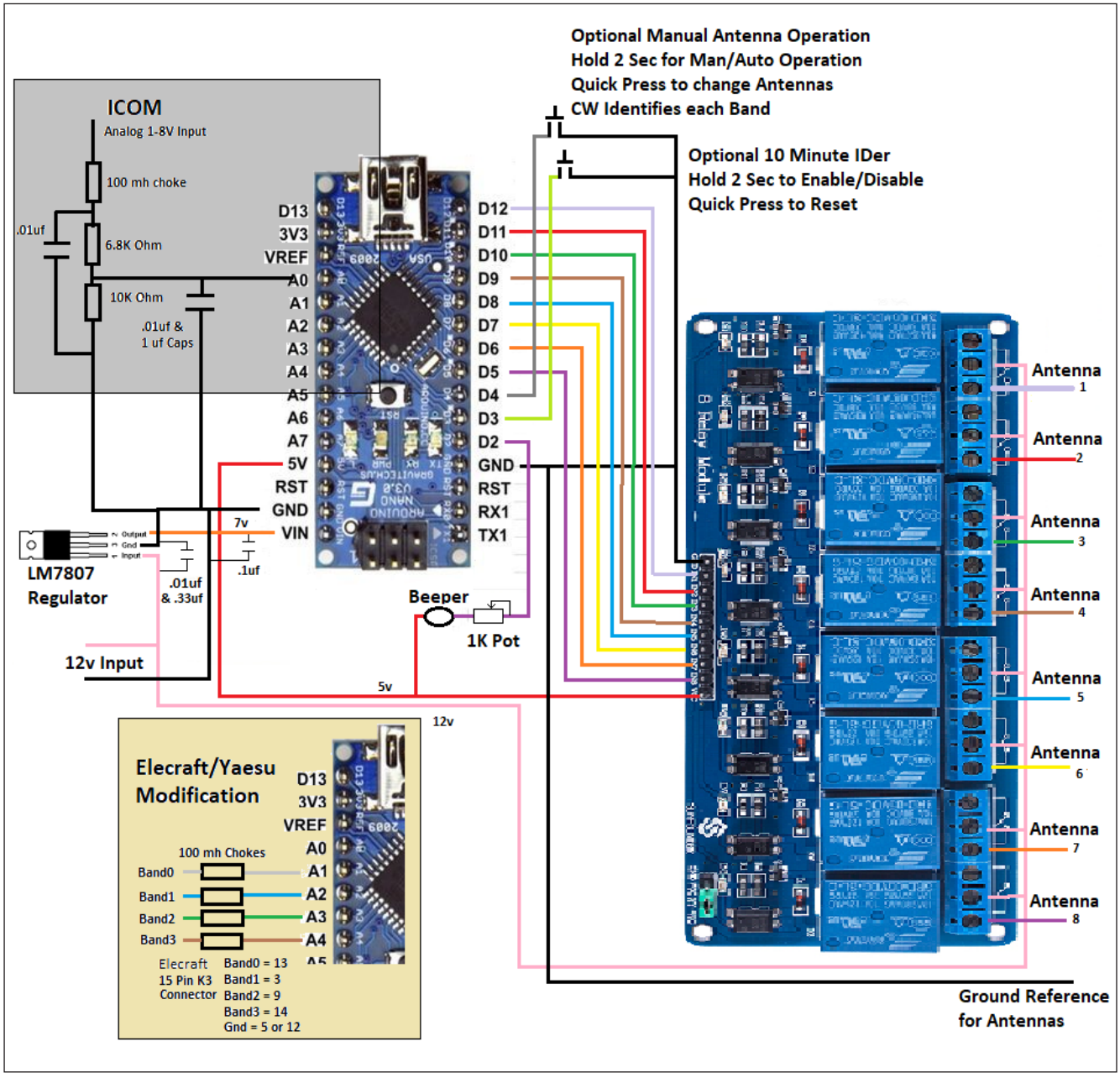

Figure 6 — MultiRadioBandSwitch schematic.

Two optional push buttons may be added. The first allows you to switch through the antennas manually. When the button is held for 2 seconds, the unit switches to manual mode and announces the antenna number in CW. If you press the button again briefly, it will switch to the next antenna, again announcing the antenna in CW. Holding the button for more than 2 seconds will return the unit to auto mode, announcing "AUTO" in CW. If the antenna button is pressed quickly while in auto mode, the relay number will be announced in CW.

The second button is used to enable a 10-minute ID timer. Hold the button for 2 seconds and it will announce "ID" in CW. At the end of 10 minutes, it will announce "ID" again. If the button is pressed briefly, the timer will reset to restart the ID timer. Holding the button again for more than 2 seconds will disable the timer function.

Figure 7 — The case for an Elecraft switch with two momentary push buttons and hole in the front panel to access the Arduino USB programming port. RCA connectors for P3 power. The 15-pin connector is for the Elecraft K3 I/O, and the 9-pin D-sub connector is for the relay antenna output drive.

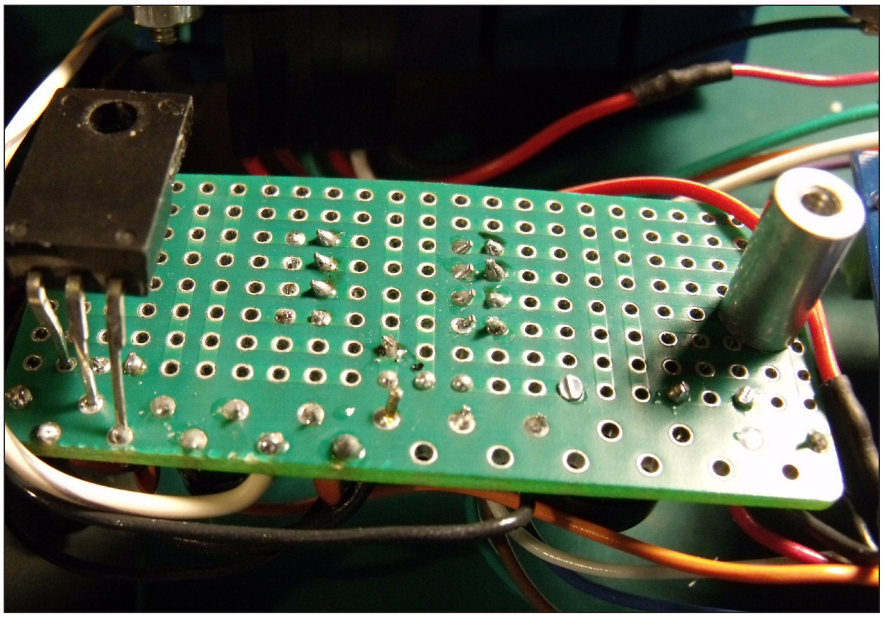

Figure 8 — Regulator mounted below the circuit board and used as a second stand-off. The nut is glued to the top side of the regulator to assist in easy screw attachment.

Icom Radios

If you have an Icom radio that provides the analog voltage (1 – 8 V) to indicate the radio's band, you'll need to adapt the circuit (see Figure 6). Because Arduino inputs cannot exceed 5 V, a voltage divider circuit using the 6.8 kΩ and 10 kΩ ohm (1/4-W) resistors is needed to limit the input to Arduino pin A0 to less than 4.8 V. These resistors should be 1% tolerance or select values that are close. A single 100 mH choke and capacitors on the input from the radio are used to prevent RF from affecting the analog voltage input. Reference your specific radio documentation for this analog output pin, but it is likely pin 5 of the ACC socket, and the 13.8-V output on the socket is typically pin 8.

Note if you bench test the analog input from the Icom, there is a slight loading effect from the 16.8 kΩ resistors on the analog output from the radio of about 0.2 V.

Elecraft/Yaesu Radios

If you have an Elecraft (K3, K3S, or K4) or a Yaesu that has the BCD band outputs, use the connections shown in the inset at the bottom left of the circuit diagram in Figure 7 in place of the Icom voltage divider. All band data inputs (A1, A2, A3, and A4) should have 100 mH chokes in line to prevent RF from affecting the inputs to the Arduino. Reference your specific radio documentation for the Band0 – Band3 pins and the 12-V output on the accessory connector. For 13.8-V output on the Elecraft radio, you can use the RCA connector that supplies power to the P3 display, add another short pigtail with a female RCA connector to allow connecting

the P3 power. The Yaesu radio uses the LINEAR socket for the band data and 13.8-V outputs. The output pins on the Elecraft and Yaesu radios are all 5-V output, so they won't need any voltage conditioning.

Building the Circuit

I've built a number of different circuits into a variety of cases, and I've learned that a metal case is necessary for noise immunity, with a ground screw to connect the case to your station ground.

When designing the chassis, keep future maintenance in mind. I designed mine so the top and bottom of the chassis can be removed with the ends still in place, so the boards can be accessed.

When constructing the case, cutting openings in the aluminum cases to accommodate connectors can be time consuming. It's much easier to run the cables through a grommet and put the connectors on the end of a short cable.

The version shown in Figures 1 and 7 uses four of the eight available antenna outputs and has an RCA plug to obtain 12 V dc from the Elecraft K3. A second female RCA jack wired in parallel with the male connector on the radio allows an Elecraft P3 display to be plugged in as well. I also mount the Arduino Nano on a screw terminal adapter. This is a small board that the Nano mounts into with push-in connectors. Each pin has a corresponding screw terminal, which is much easier than any other method and cost less than $5. See the Arduino mounts in Figures 1, 2, or 3.

As for mounting the boards into the case, I used 1/4-inch standoffs with 4-40 screw hardware. The LM7807 voltage regulator is fastened, using the case as a heat-sink, beneath the small perf-board. The regulator also acts as a standoff for one end of the perf-board. See Figure 8. For the relay card, I used the slotted edge on the aluminum case for the one side of the relay board, and standoffs for the other side (see Figure 8). A 1/2-inch hole is cut in the front of the case to allow access to the Arduino USB connector for programming.

I would also suggest drilling a series of five or six 1/4-inch holes across the bottom of the case near the front, and another similar series across the top of the case near the back to create some air flow through the case for cooling.

Loading the Arduino IDE

You can download the code into the Arduino before you have it built up into the chassis. The connections are not required to load the code into your Arduino just to become familiar with the process.

You will need to load the Arduino IDE onto your PC. This can be downloaded from Arduino.cc. Locate the Software and Download Tools. A web editor version is available, but this is only valid for genuine Arduino boards and will *not* work with the clones. Install the Arduino IDE on your PC, Mac, or Linux system using just the default option.

If you are using a clone NANO, load the CH340 driver before connecting the Arduino.

You can google "Arduino ch341 driver download" and download and install the appropriate driver for your system platform (Windows 32 bit or 64 bit, Linux, or Mac).

Load the **MultiRadioBandSwitch.zip** files from the *NCJ* website, **www.ncj.com**, and extract the files (double click on the downloaded file). You may choose to move the extracted folder and files to a more permanent location on your computer hard drive first.

With the Arduino IDE already installed on your computer, double click on the **MultiRadioBandSwitch.ino** filename. All of the .ino-extension files in the same directory will load into the editor. Note also that the files *must* be in a directory called "MultiRadioBandSwitch," matching the **MultiRadioBandSwitch.ino** name.

Connect your Arduino Nano to a USB port on your computer. The Nano typically requires a mini-B type USB cable.

In the Arduino IDE, select the Tools and Board menu selections. Click Arduino Nano within this menu.

In the Arduino IDE, select the Tools and Port menu selections. Select the serial port that the Arduino Nano is connected to (you may need to remove the Arduino Nano USB cable and note which ports are still listed to identify the correct port. Re-install the Arduino Nano USB cable and select the USB port that was added to the list.)

The code loaded into the Arduino Editor should load into your Arduino Nano, even if you do not have the complete circuit built up.

♦ Select the Sketch menu and Verify/Compile. This should complete successfully. No other Libraries are required for the Arduino sketch I've provided.

♦ Select the Sketch menu and click on Upload. If this fails, you may need to select the Tools menu, Processor, and select ATmega328P (Old Bootloader).

NOTE: Do not change the Tools/Programmer. It should remain set as AVRSIP mkII.

Modifying the Code

Three settings need to be modified in the code for your specific station needs. These are all near the top of the **MultiRadioBandSwitch.ino** file, which should be the left-most tab in the Arduino editor.

1. Define which radio type you are using. You need to comment out the radio *not* being used with two backslashes (i.e, //).

To set up for Elecraft or Yaesu radios (//#define Icom is commented out):

```
#define Elecraft
//#define Icom
```

To set up for an Icom radio (//#define Elecraft is commented out):

```
//#define Elecraft
#define Icom
```

Note: These lines do *not* end in semicolons (;).

2. Define which relay will be activated for each bands. Each band

needs to have a relay defined in order to configure the system for your particular station setup.

```
//Define The RELAY for Each BAND (Even if it's not a resonant antenna, something needs to be connected.)
  const int i160m = Relay1;
  const int i80m = Relay1;
  const int i60m = Relay1;
  const int i40m = Relay2;
  const int i30m = Relay2;
  const int i20m = Relay3;
  const int i17m = Relay3;
  const int i15m = Relay3;
  const int i12m = Relay4;
  const int i10m = Relay4;
  const int i6m = Relay4;
```

Note: Each line must end with a semicolon.

For the example above, Antenna 1 (Relay1) is used for 160, 80, and 60 meters. Antenna 2 (Relay2) is used for 40 and 30 meters. Antenna 3 (Relay3) is used for 20, 17, 15, 12, and 10 meters. And Antenna 4 (Relay4) is used for 6 meters. These can all be redefined to assign each band to match your specific antennas.

Also note carefully that each band, 160 through 6 meters, is assigned to an antenna relay. You should assign at least one relay for each band, even if you don't have a valid antenna for the band. If no antenna is defined, and you transmit on that band, the radio would transmit into an open coax.

3. Define the highest relay used. There is the line of code, just below the antenna assignments:

```
  const int HighestRelay = 4;
```

This should be updated to define the number of antennas, or relays, that you are using in your circuit. This allows manual mode to cycle *only* through the antennas you are using. This can be the actual number of relays that you are using for your current antenna setup, not necessarily the number of relays you have installed. For instance, if you have four relays built into the circuit but are only using three, you can define the `HighestRelay = 3;` on the line. You can later change this to 4 and re-download the code to the Arduino to update it.

A word of warning: For each line of code you edit, make sure you do not remove the semicolon (;) at the end of each line. These are required.

PDF article:
https://ncjweb.com/bonus-content/W0IH-MultiRadioBandSwitch.pdf

Code/support files:
https://ncjweb.com/bonus-content/W0IH-MultiRadioBandSwitch.zip