NanoWSPR
A Low-Cost Multiband Transmitter

Build this microcontrolled transmitter from available modules without performing any Arduino programming.

George R. Steber, WB9LVI

Weak-signal propagation reporter (WSPR) is one of today's highly sophisticated radio modes that permits communication to the far corners of the world using very little power. What makes this possible is a combination of special coding techniques and software that can reliably decode signals near the noise threshold. It was described in detail in the article, "WSPRing Around the World," by Joe Taylor, K1JT, and Bruce Walker, W1BW, from the November 2010 issue of *QST*. It has proved to be useful, providing practical information on radio propagation paths for all to see on **WSPRnet.org**.

I built an ultra-low-cost WSPR transmitter, which I dubbed "NanoWSPR" because of its Arduino Nano heritage. No Arduino programming is required for this project. For less than $20, it should make an attractive weekend project. You need a laptop or a PC with a USB port to make it work.

WSPR Operation

Ordinarily, you'd need a computer running WSPR software, a sound card, and an SSB transceiver to participate in WSPR communications and view real-time results on the WSPRnet world map. The transceiver is connected to the sound card, which is used by the software to both transmit and receive WSPR signals. In actuality, you don't need a transceiver — usually a receiver or a transmitter will do, depending on your goal.

Teachers, students, and shortwave listeners (SWLs) routinely report receptions using just a receiver. Receivers allow monitoring the entire WSPR band at one time without tuning for individual stations. Students can learn a lot about radio theory by participating in this effort. There are even lesson plans for teachers, thanks to Professor Lynne Reynolds, PhD, available at **stellarworldwide.bravesites.com**.

Many of us enjoy building transmitters. For this project, I used readily available modules to build a low-cost, WSPR transmitter with multiband capability (10 to 160 meters). This transmitter produces signals in the 12.5 to 125 mW range. Even with such miniscule power, the results are astonishing. Figure 1 shows a WSPR map from VK7JJ (**wsprd.vk7jj.com/**) of all the stations that reported receiving my 30-meter signal over a period of just a few hours. In one startling case, not shown, my signal was received by OE9GHV 7,050 kilometers away. The amazing part is that my transmitter was running only 125 mW into a quarter-wavelength wire antenna raised about 8 feet above the floor.

"This NanoWSPR transmitter employs only two modules — an Arduino Nano and a Si5351 clock generator."

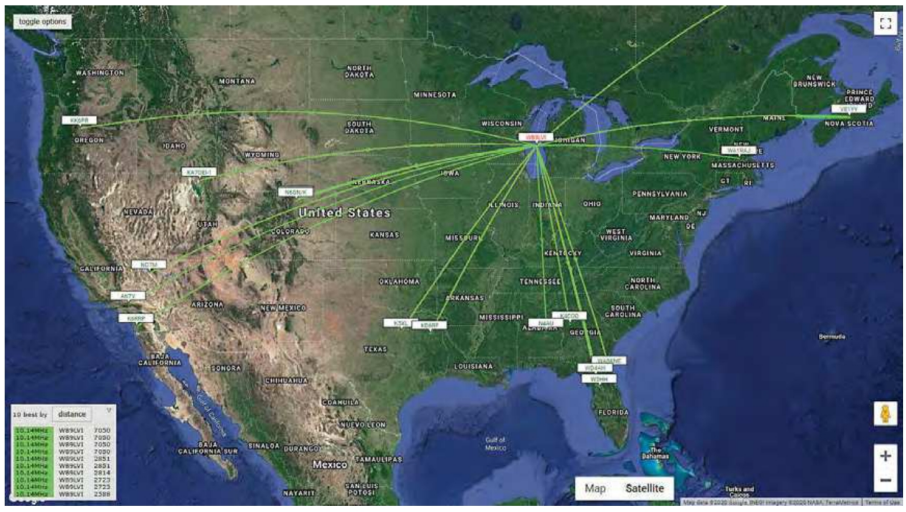

Figure 1 — This WSPR map shows WB9LVI receptions on the 30-meter band using the NanoWSPR transmitter.

This NanoWSPR transmitter employs only two modules — an Arduino Nano and a Si5351 clock generator. Power is supplied by the USB port. An optional chip can be added if you want to increase the transmitted power. Although not absolutely needed, an oscilloscope and frequency counter would be handy to monitor the signal and measure the frequency. Finally, you must assemble an RF filter and an antenna for the band of interest.

So, if you want to get started with the worldwide WSPR network of low-power stations probing radio propagation paths, build this NanoWSPR transmitter.

Design Philosophy

There are WSPR transmitter designs that can be built using Arduino or other microcontrollers. They often involve attaching real-time clocks, GPS modules, or OLED displays to make them portable. In order to use these designs, you must learn how to modify the firmware using a development system. This is great if you want to learn more about how software is created and used in a device. But the downside is that you must download a lot of integrated development software (IDE) onto your computer. In addition, many of these software environments require having Microsoft Framework installed on your computer. Windows 10 usually comes with Framework installed, but laptops running older versions often do not.

Many of us just want to build a project, not learn how to be an expert programmer. With that in mind, I decided to create a project that does not involve programming. However, you'll need to know how to install a driver and run a program on your computer to load a hex file into the Arduino Nano. In some cases, you may need to register a file on your PC. You should be able to run it on an old PC with a USB port, running any version of Windows.

You can now concentrate on designing and constructing your antenna and RF filters. You can tailor other aspects of the project to what you want, such as arranging the modules and designing your own enclosure.

Arduino Nano and Si5351 Clock Generator Modules

Figure 2 shows the Arduino Nano and the Si5351 modules used here. The Si5351A Clock Generator Breakout Board (8 kHz to 160 MHz) works well with the Arduino Nano and is available from multiple online sources for $8 to $10. It uses the onboard precision clock to drive multiple PLLs and clock dividers using I2C instructions. By setting up the PLL and dividers, you can create precise arbitrary frequencies. There are three independent outputs, and each one can have a different frequency. Outputs are 3 V_{p-p} square waves, either through a breadboard-friendly header or, an optional SMA connector. For this project, the module should have an onboard 25 MHz crystal.

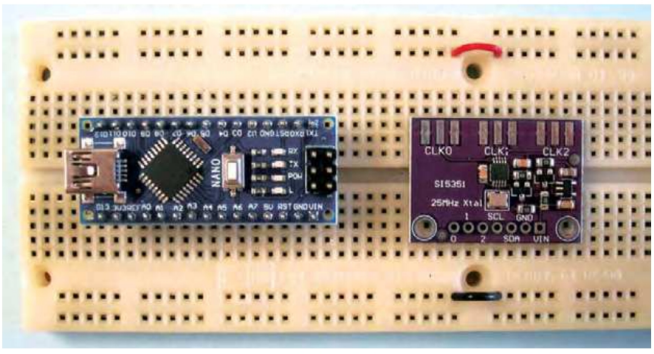

Figure 2 — The Arduino Nano and Si5351 modules used in this project.

Characteristics of WSPR Signals

A WSPR signal is basically a frequency-shift keyed (FSK) tone and does not require amplitude linearity. We can utilize the Si5351 clock with square wave output. A filter is then added to remove higher order harmonics inherent in the square wave.

WSPR signals have a bandwidth of only 6 Hz and fit into a tiny 200 Hz segment on a given band. This permits several tens of stations to coexist with minimal interference. For example, the 30-meter band is centered at 10.140200 MHz. Such a narrow band requires a very accurate and stable oscillator reference for the transmitter. Because the Si5351 clock depends on the onboard 25 MHz crystal, we will likely have some frequency error in our transmitted signal due to crystal tolerance. This error must be corrected to avoid transmitting outside the WSPR band.

NanoWSPR Transmitter Design

My minimalistic transmitter achieved several key features. It was very low cost, had no Arduino programming, had multiband capability, and had the ability to enter call signs, maidenhead grid coordinates, and power levels. It produces an output of 12.5 mW and can be upgradable to 125 mW or higher.

"By setting up the PLL and dividers, you can create precise arbitrary frequencies."

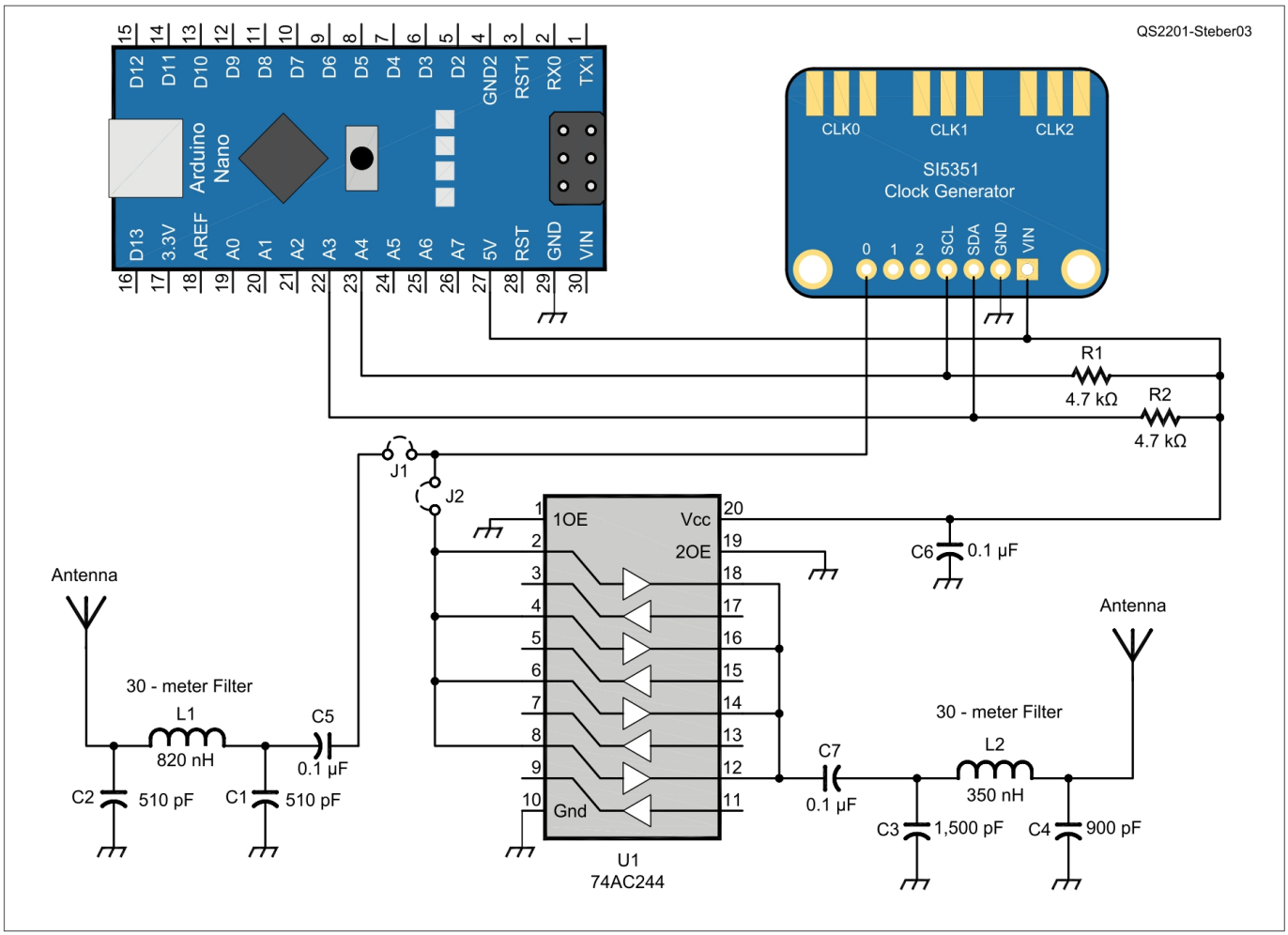

Figure 3 — A schematic diagram of the NanoWSPR multiband transmitter. All parts from Mouser Electronics or eBay.

A schematic for the transmitter is shown in Figure 3. There is very little wiring between the modules, and power is derived from the USB port. You need two 4.7 kΩ pullup resistors (R1, R2) on the SCL and SDA lines.

Various techniques may be employed to build this transmitter. As a first pass, I suggest that the two modules be mounted on a solder-less breadboard strip. This is good for loading the firmware and testing the software. Be very careful when installing the module pins in the strip, as the modules can be damaged by excessive force. Later on, you can decide on a more permanent arrangement of the modules, perhaps on a PCB and in an enclosure with connectors.

There are only a few steps to perform to get your NanoWSPR running, including using the NanoWSPR project software. There are three folders in the project folder on **arrl.org/qst-in-depth** web page.

Loading Firmware into Arduino Nano

Firmware must be loaded into the Arduino Nano for it to operate. This is done just once. First connect the Arduino Nano to your PC with a USB cable. If it is recognized, a COM port will be generated and will appear in Device Manager under Ports (COM & LPT), such as COM1, COM2, etc.

If a new COM port is not found, then the serial driver for the Arduino Nano is probably not installed. In this case, go to the project software and install the CH340 serial port driver.

When this is done successfully, a new COM port will appear in Device Manager. Make a note of this COM port, as it will be needed to install the firmware. Let's assume that COM12 was assigned by your PC for the Arduino Nano. (Of course, it will be different for every PC.)

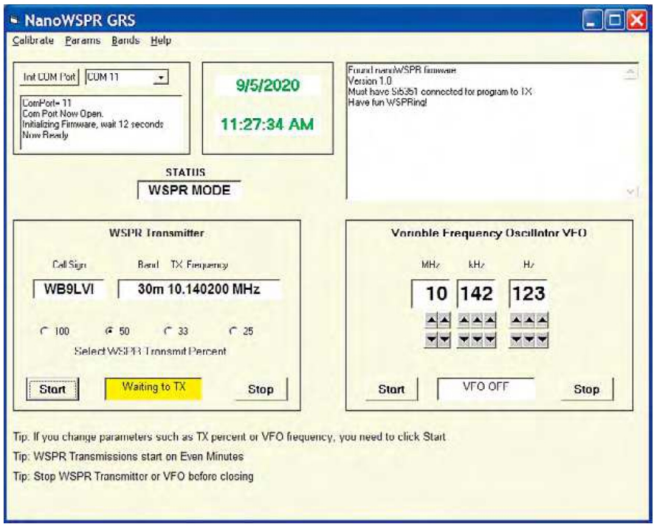

Figure 4 — The main window of the NanoWSPR program.

Now go to the **HEX FILE LOADER** folder in the project folder. Edit the "*bat*" file in the folder with a simple editor like *Notepad*. Do not open it or use it with a sophisticated word processor like Word, as that will add extra characters to the file. Find the following line: "avrdude -C avrdude.conf -v -p atmega328p -c arduino -P COM11 -b57600 -D -U flash:w:WSPR.hex:i"

Change the COM11 to the number of your COM port, say COM12. Save the file. Now, run the *bat* file by clicking on it, and it will load the firmware into your Arduino Nano.

NanoWSPR Software

Software was written in Visual Basic to control the Arduino Nano for WSPR operation. Find the program *NanoWSPR.exe* in the project software and run it. Figure 4 shows the NanoWSPR screen. Here again you must know the COM port assigned to the Arduino Nano. Follow the on-screen instructions by selecting the COM port and initializing it.

If everything is going well, you should see a message saying that the firmware has been found and that it is being initialized — taking up to 12 seconds.

If things did not go well, there are some things to try. Sometimes a necessary file like *mscomm32.ocx* is not registered. Place it in Windows **SYSTEM32** directory or **SYSWOW64** folder, and register it. This file and others are discussed in the project software *readme* file.

We cannot rely on the onboard 25 MHz Si5351 crystal to be correct, so we must calibrate our unit. The easiest way is to set the output frequency to 10 MHz using the NanoWSPR VFO. Now measure it with a frequency counter at Clock 0. Be sure to insert a 470 Ω resistor in the counter test lead to avoid overloading the Si5351.

In my case, the measured frequency was 69 Hz high. The error at 25 MHz would be 69 × 2.5 = 172.5 Hz. Therefore, the corrected crystal frequency is 25,000,000 + 172.5, or 25,000,172 Hz. Enter the correct crystal frequency into the NanoWSPR program. Don't expect absolute accuracy, as the frequency will change with temperature, but it is good enough to hit the WSPR bands without problems.

Filters and Antennas

Each WSPR band transmitter square wave must be filtered and matched to the antenna. Because of the low-output power, you can use small inexpensive components for the filters. Mouser and eBay are good sources.

In Figure 3, there are two RF filters and a 74AC244 integrated circuit (IC). With J1 connected (and J2 open), a 30-meter filter is connected to the Clock 0 output of the Si5351 via capacitor C5. This is the low-power output, with transmitting power in the range of 10 to 20 mW. I designed the filter using RF Tools LC filters design at **https://rf-tools.com/lc-filter**. It is a 50 Ω terminated, third-order, 0.5 dB ripple Chebyshev filter for the 30-meter band.

For higher output power, connect J2 (and open J1) to use the 74AC244 IC. Four drivers are connected in parallel to provide 125 to 150 mW of power. If you want even more power, you can add the four other drivers in the chip in parallel to get up to 300 mW output. In designing a filter for the 74AC244, we want to match the four paralleled outputs of the 74AC244 — roughly 12 Ω — to the 50 Ω antenna. We need a 1:4 impedance matching filter. The 30-meter filter comprises C3, L2, and C4 in Figure 3. Use online filter design tools to design filters for other WSPR bands.

Ordinary resonant half-wave dipoles can be used with good success. You can also use a quarter-wavelength of wire with a long wire counterpoise.

NanoWSPR Setup

WSPR transmits on even-numbered minutes, and the start of transmission must be accurate to within 1 second. Correct UTC can be set on your PC by using internet time. Otherwise, you can use low-cost atomic time clocks tuned to WWV.

Your call sign, Maidenhead grid, power level, and transmit percent also must be set up in the

"Teachers, students, and shortwave listeners (SWLs) routinely report receptions using just a receiver."

NanoWSPR software. To share spectrum, please reduce your transmissions to less than to 50%. This will allow others to share the frequency. Select your output filter, connect your antenna, and you are ready to go.

NanoWSPR Final Words

The NanoWSPR transmitter should make a nice project for those wishing to experiment with the WSPR mode. Results will depend on the band chosen and antenna construction. Even with such low power, don't be surprised if you get a spot from across the ocean.

Note that not all laptop USB ports are the same. If you are having difficulty using the 74AC244, consider running it on its own power supply. Adding an extra bypass capacitor might help too.

I enjoyed designing and using this NanoWSPR transmitter. WSPRing is fun and also educational. In addition to observing which propagation paths are open, it can be used to demonstrate low-power radio technology.

Please let me know when you get your NanoWSPR transmitter working. Propagation permitting, your signal will be copied in the far distant regions of the world.

George R. Steber, PhD, WB9LVI, is Emeritus Professor of Electrical Engineering and Computer Science at the University of Wisconsin-Milwaukee. He is retired, having served over 35 years, and he still lectures occasionally on science and engineering topics. George has an Advanced-class license, is a Life Member of ARRL and IEEE and is a professional engineer. He is involved in cosmic ray research and is developing methods to study them on a global basis. He has also worked for NASA and the USAF. George has written articles for several technical magazines, including *QEX* and *Nuts and Volts*. When not dodging protons, pions, and muons, he enjoys WSPR and JT9 amateur radio, racquet sports, astronomy, and jazz. You may reach him at **steber@execpc.com** with "Nano" in subject line and email mode set to text.

For updates to this article, see the *QST* Feedback page at www.arrl.org/feedback.

The DC2020 Receiver

This basic direct-conversion receiver uses just a few parts and can be modified for other bands.

Harold Smith, KE6TI

This is an ordinary direct-conversion receiver of fairly simple design, comprising only seven transistors. I built it for quick and easy assembly and mounted it in an enclosure. I used only leaded through-hole components, (no integrated circuits — ICs), with all easily available parts. It tunes the bottom 100 kHz of the 7 MHz band for CW reception. Increasing the cutoff frequency of the audio low-pass filter could also allow reception of SSB. A 1 μV signal from a signal generator is clearly audible in my headphones.

Circuitry

The receiver consists of four stages (see Figure 1):

1 A variable oscillator that is voltage tuned via a pair of common rectifier diodes.

2 A product detector with audio output.

3 A low-pass filter that rolls off above roughly 1 kHz.

4 An audio amplifier with more than enough gain to drive headphones.

I chose to tune the variable oscillator with a pair of rectifier diodes serving as varactors. This avoids the expense of an air variable capacitor. You might try diodes from your junk box in place of the 1N4002s that I used.

I chose a Hartley oscillator configuration, because it gave good tuning range with not too extreme tuning voltages. The Hartley uses a tapped inductor that I wound myself on a T50-7 toroid. I also used only NP0 and C0G ceramic capacitors in the oscillator to provide stability that was more than adequate for CW. I did have to tweak the coil a little to put the oscillator range where I wanted it, which allowed me to avoid using a trimmer capacitor. The tweaking consisted of compressing and stretching the turns on the toroid until it tuned the range I wanted. You can also add or subtract turns if needed.

The mixer and product detector is a differential stage, which gives good gain and light loading of the tuned circuit. The tuning range in this version is small enough that I didn't need to provide for peaking the front end to follow the oscillator tuning. You can change the bandwidth of the front end by paralleling a resistor across C1 to lower its Q, or by changing the number of turns on the primary of T1. This will lower the front-end gain.

An active low-pass filter with a cutoff of about 1 kHz follows the detector stage. It should be scalable to other frequencies by changing the values of the low-pass filter resistors. For a 3 kHz cutoff, make R11 and R12 each 510 Ω and change C17 to 100 pF. This will increase the audio bandwidth without affecting the gain.

The two-transistor audio amplifier is based on a Sziklai Pair. It produces plenty of audio to drive headphones. Mount the audio volume control, R15, close to Q5 and Q6. I used a volume control with a built-in switch to turn the power to the radio **ON** and **OFF**. The very large decoupling capacitor, C19, is used for hum suppression.

The low-frequency response is determined almost entirely by the value of C20 and the impedance of the headphones. It is common homebrew practice to use 32 Ω stereo phones with little radios like this, usually with the phone

"Increasing the cutoff frequency of the audio low-pass filter could also allow reception of SSB."