Wsprry Pi Documentation

Release 1.2.0-2-g0f89a8d

Lee C. Bussy (and others)

Mar 07, 2024

CONTENTS

This project is for amateur (ham) radio enthusiasts, using an inexpensive single-board computer (Raspberry Pi) to experiment with signal propagation using WSPR. This implementation of WSPR, called “wspr” here to differentiate the application from the protocol, has been available for quite some time as an application that utilizes the specific capabilities of the Raspberry Pi.

Wsprry Pi (whispery pie) was developed to enable ham enthusiasts to enter this realm with relatively low-cost equipment.

CHAPTER

ONE

TABLE OF CONTENTS

1.1 About WSPR

WSPR, “Weak Signal Propagation Reporter,” is a protocol for low-power, digital communication on amateur radio frequencies. WSPR is designed to be a highly efficient way of transmitting and receiving information over long distances, even under challenging radio propagation conditions.

WSPR operates by encoding information in a series of tones, then transmitted in short, controlled bursts using a highly efficient modulation scheme. The resulting signal is often too weak to be heard by human ears but can be decoded by specialized software on a receiving station, which can extract the encoded information from the noise.

WSPR is transmitted on a schedule, allowing multiple stations to share a single frequency. This will enable WSPR for various applications, including monitoring and analyzing radio propagation conditions, testing equipment and antennas, and conducting experiments in low-power, long-range communication.

One of the critical features of WSPR is its ability to operate in very low signal-to-noise ratios, which allows for reliable communication over long distances, even under poor propagation conditions. This makes it a popular choice among amateur radio enthusiasts and researchers interested in studying long-range radio propagation and other related phenomena.

WSPR is the software that implements MEPT-JT, which started life in 2008 as a suggestion by Murray ZL1BPU. Joe K1JT took up the challenge, designed the software, and added many improvements. The name ‘MEPT-JT’ is derived from ‘MEPT’ (Manned Experimental Propagation Transmission), a generic term for a range of low-powered techniques, and ‘JT’ is Joe Taylor’s initials. It is one of a general family of D-MEPT (digital MEPT) modes.

The idea was to take a minimal message, code them as tightly as possible, add forward error correction, and then combine this with a known pseudo-random sequence to generate a two-minute-long message to be sent using 4-FSK modulation and a low baud rate. Joe estimated that the theoretical sensitivity of the system would be -30dB S/N.

Transmissions carry a station’s callsign, Maidenhead grid locator, and transmitter power indicated in dBm. Receiving stations can decode signals with a signal-to-noise ratio as low as 28 dB in a 2500 Hz bandwidth. Stations with internet access can automatically upload their reception reports to a central database called WSPRnet, which includes a mapping facility.

The basic specifications of the MEPT_JT mode are as follows:

- Transmitted message: callsign + 4-character-locator + dBm Example: “K1JT FN20 30”
- Message length after lossless compression: 28 bits for callsign, 15 for locator, 7 for power level ==> 50 bits total.
- Forward error correction (FEC): long-constraint convolutional code, K=32, r=1/2.
- Number of channel symbols: nsym = (50+K-1)*2 = 162.
- Keying rate: 12000/8192 = 1.46 baud.
- Modulation: continuous phase 4-FSK. Tone separation 1.46 Hz.

- Synchronization: 162-bit pseudo-random sync vector.
- Data structure: each channel symbol conveys one sync bit and one data bit.
- Duration of transmission: 162*8192/12000 = 110.6 s.
- Transmissions start two seconds into an even UTC minute: i.e., at hh:00:02, hh:02:02, …
- Occupied bandwidth: about 6 Hz
- Minimum S/N for reception: around -27 dB on the WSJT scale (2500 Hz reference bandwidth).

Like JT65, MEPT_JT includes efficient data compression and strong forward error correction. Received messages are nearly always the same as the transmitted message or are left blank.

1.2 About Wsprry Pi

WsprryPi creates a very simple WSPR beacon on your Raspberry Pi by generating a Pulse-Width Modulation (PWM) square-wave signal through a General-Purpose Input/Output (GPIO) pin on a Raspberry Pi. This is connected through a Low-Pass Filter to remove harmonics and then to an appropriate antenna. It operates on LF, MF, HF, and VHF bands from 0 to 250 MHz.

This image shows a square waveform, with an overlay showing how a waveform might look through successive low-pass filters:

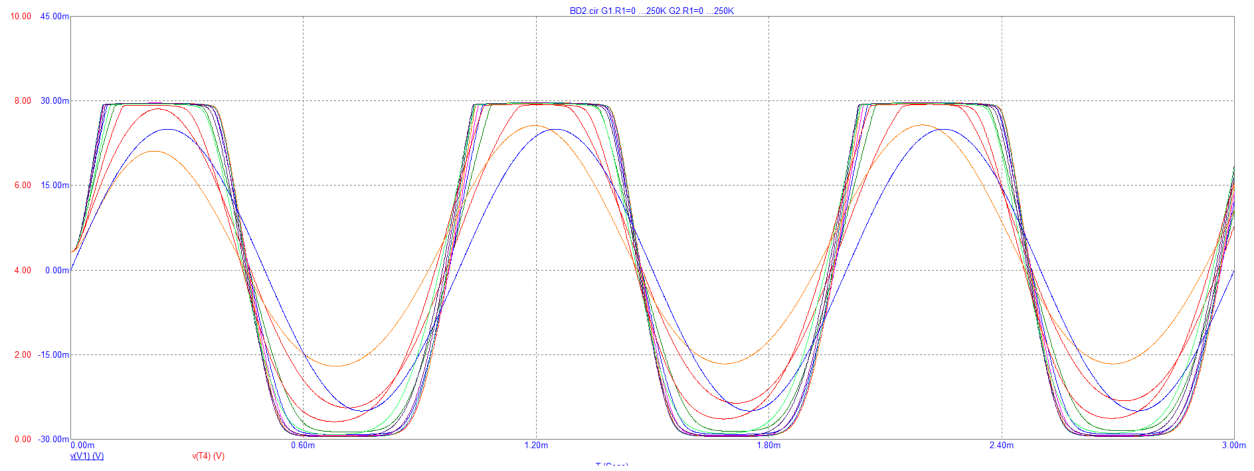


Image from https://analogisnotdead.com/article25/circuit-analysis-the-boss-bd2

You should not use Wsprry Pi without a low pass filter as it will create interference from harmonics on other bands.

This software is compatible with the original Raspberry Pi, the Raspberry Pi 2/3, the Raspberry Pi 4, and the Pi Zero.

1.2.1 Attribution

The original WsprryPi was written by Dan Ankers and released on GitHub. This work was, in turn, based on Dan's PiFmDma where Dan further expanded (with DMA) upon previous work. That earlier work was Oliver Mattos' FM Improved.

If I need to correct any of this history, please let me know by opening an issue.

threeme3 adapted Dan's repo, and then I forked threeme3's work here. Bruce Raymond from TAPR (who sell the WPSR Without Tears hat for the Raspberry Pi) incorporated a few things in some complete Pi images that TAPR makes available, and I wanted to incorporate those here as well. Of course, the Internet being what it is, the libraries Bruce used are no longer available, so I had to go in a different direction.

Where do I come in? Well, Bruce was friendly enough to answer my email. I'm a new ham in 2022 (KF0LDK originally, now AA0NT), still trying to find my way, but I have some minor success in the Raspberry Pi ecosystem, so I decided to take on the gap which might exist between a ham and a new Pi owner. I want to be VERY clear. I am not your Elmer; I cannot help with ham radio. This project stops where WSPR executes on the Pi, and you are responsible for your actions, your hardware, and your level of understanding.

My goal, and where success will be determined, is to allow you to execute one command on your Pi to install and run the Wsprry Pi software. If you are lucky and have been living right, a radio wave will hit the cosmos and be received somewhere else.

1.3 Installing Wsprry Pi

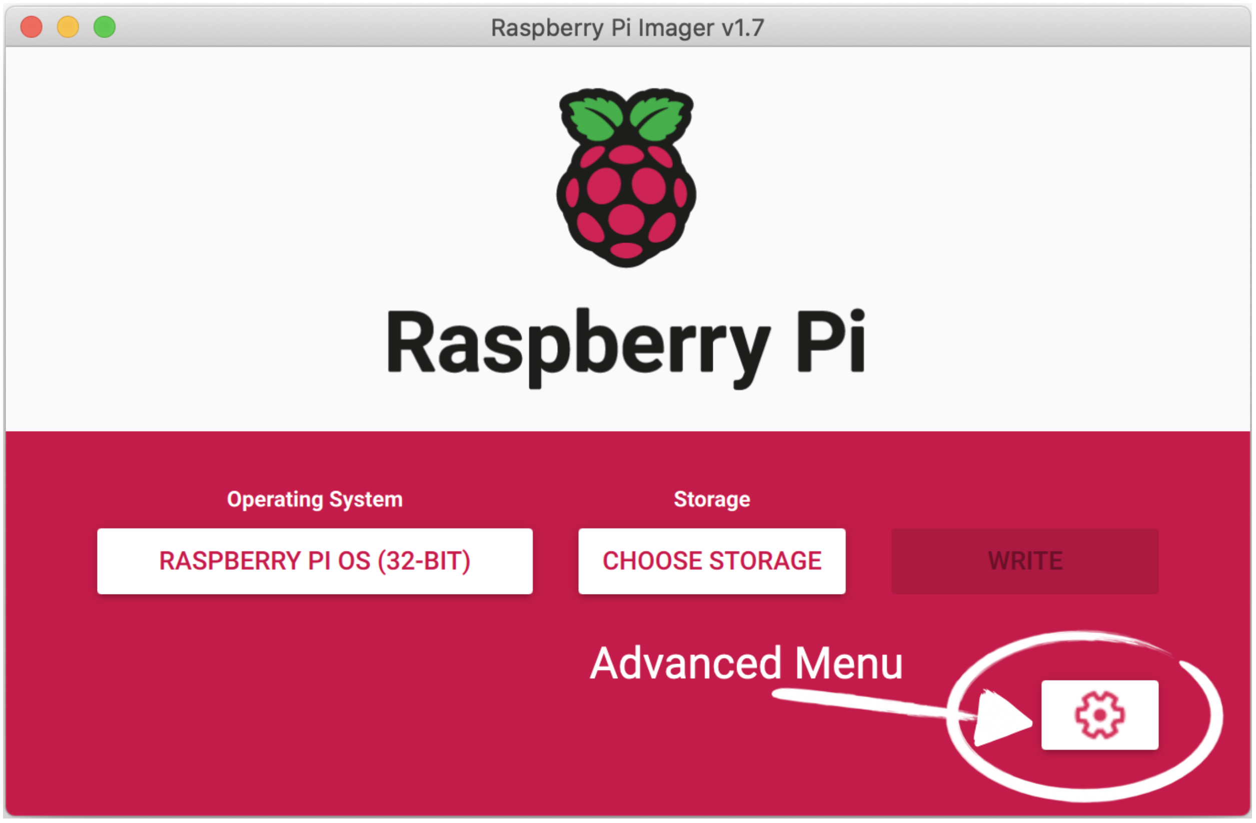
1.3.1 Gather Hardware

You will need the following:

- A Raspberry Pi of any type
- An SD card for the OS image
- A power supply for the Pi. Pay attention here to potentially noisy power supplies. You will benefit from a well-regulated supply with sufficient ripple suppression. You may see supply ripple as mixing products centered around the transmit carrier, typically at 100/120Hz.

1.3.2 Prerequisites

This may be the most challenging part of the whole installation. You must have a working Raspberry Pi with Internet access. It can be hard-wired or on Wi-Fi. There is no better place to learn how to set up your new Pi than the people who make it themselves. Go here, and learn how to install the operating system with the Raspberry Pi Imager. You can pre-config your image with your local/time zone, Wi-Fi credentials, and a different hostname and enable SSH access from the little cog wheel on the Imager's control page.



Select any Raspbian image you like. You can use a full-featured desktop version with all the bells and whistles, or wspr will run just fine on the lite version on an SD card as small as 2 GB (although a minimum of 8 GB seems to be more comfortable these days.) If you use the cog wheel to configure your Pi first, as described above, you can even run it headless without a keyboard, mouse, or monitor. Provided you enabled SSH, you can use your command line from Windows 10/11, MacOS, or another Pi.

Whatever you do, you will need command line access to your Pi to proceed. Once you are up and running and connected to the Internet, you may proceed with Wsprry Pi installation.

1.3.3 System Changes

Aside from the obvious installation of Wsprry Pi, the install script will do the following:

- Install Apache2, a popular open-source, cross-platform web server that is the most popular web server by the numbers. The Apache Software Foundation maintains Apache. Apache is used to control wspr from an easy-to-use web page.
- Optionally install support for TAPR's shutdown button.
- Disable the Raspberry Pi's built-in sound card. Wsprry Pi uses the RPi PWM peripheral to time the frequency transitions of the output clock. The Pi's sound system also uses this peripheral; any sound events during a WSPR transmission will interfere with WSPR transmissions.

1.3.4 Install WSPR

You may use this command to install Wsprry Pi (one line):

`curl -L installwspr.aa0nt.net | sudo bash`

This install command is idempotent; running it additional times will not have any negative impact. If an update is released, re-run the installer to take advantage of the new release.

The first screen will welcome you and give you instructions:

```
You will be presented with some choices during the install.
Most frequently you will see a 'yes or no' choice, with the
default choice capitalized like so: [y/N]. Default means if
you hit <enter> without typing anything, you will make the
capitalized choice, i.e. hitting <enter> when you see [Y/n]
will default to 'yes.'

Yes/no choices are not case sensitive. However; passwords,
system names and install paths are. Be aware of this. There
is generally no difference between 'y', 'yes', 'YES', 'Yes'.

Press any key when you are ready to proceed.
```

Take special note about default choices, `[Y/n]` means that hitting `Enter` will default to "Yes," should you want to select no, you should choose "N."

The first choice you make will be related to the system time zone and time:

```
The time is currently set to Sun 19 Feb 12:05:48 CST 2023.
Is this correct? [Y/n]:
```

Please be sure this is correct. If you select "No," the script will provide you with the Raspbian configuration screens to set this correctly.

The next choice you make will be whether or not to add support for the system shutdown button:

```
Support system shutdown button (TAPR)? [y/N]:
```

This button is included on the TAPR hat and allows one to press the button on the hat to initiate a clean system shutdown. It is best to shut your Pi down correctly to avoid corrupting the SD card; the button allows that without logging in, typically shutting the Pi down in a few seconds.

Following this choice, some system packages, including Apache2, an open-source web server, will be installed.

When complete, the script displays the final screen:

```
The WSPR daemon has started.
 - WSPR frontend URL   : http://192.168.1.24/wspr
                  -or- : http://wspr.local/wspr
 - Release version     : 0.0.1

Happy DXing!
```

At this point, Wsprry Pi is installed and running.

Note the URL for the configuration UI listed as a {name}.local and IP address choice. Using your system name is a Zeroconfiguration service capability enabled by default with Apple products. It allows accessing your home systems with their local names without remembering the IP Address. The .local name is convenient for automatically-assigned

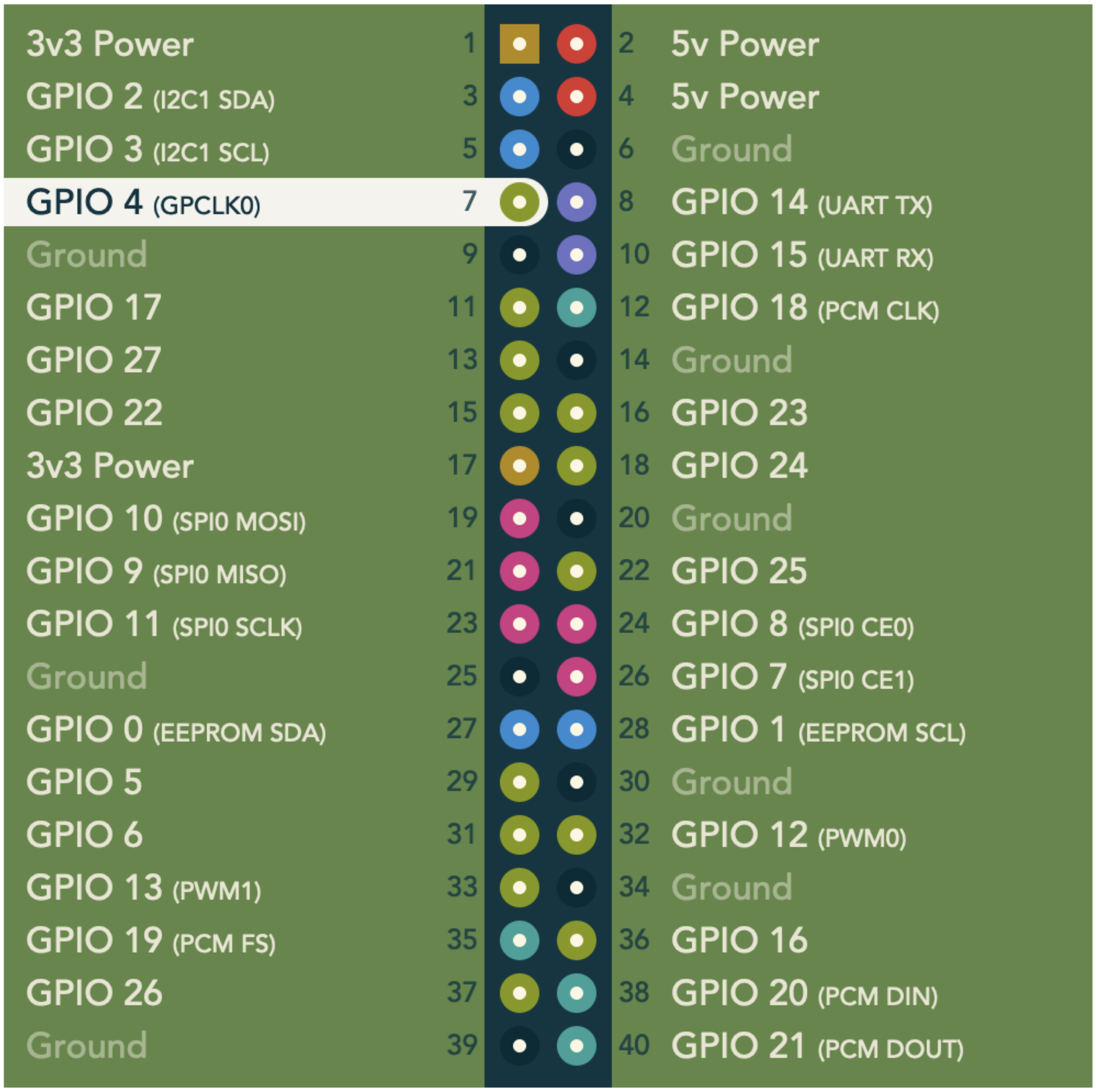
IP addresses in most home networks. The IP address of your Raspberry Pi may change over time, but the name will not.

Some Windows systems can use this naming standard without additional work; some may need Apple's Bonjour Print Services installed to enable this helpful tool.

Connect to your new web page from your favorite computer or cell phone with the IP address or the {hostname}.local name, and you can begin to set things up. See Operations for more information.

1.3.5 Additional Hardware

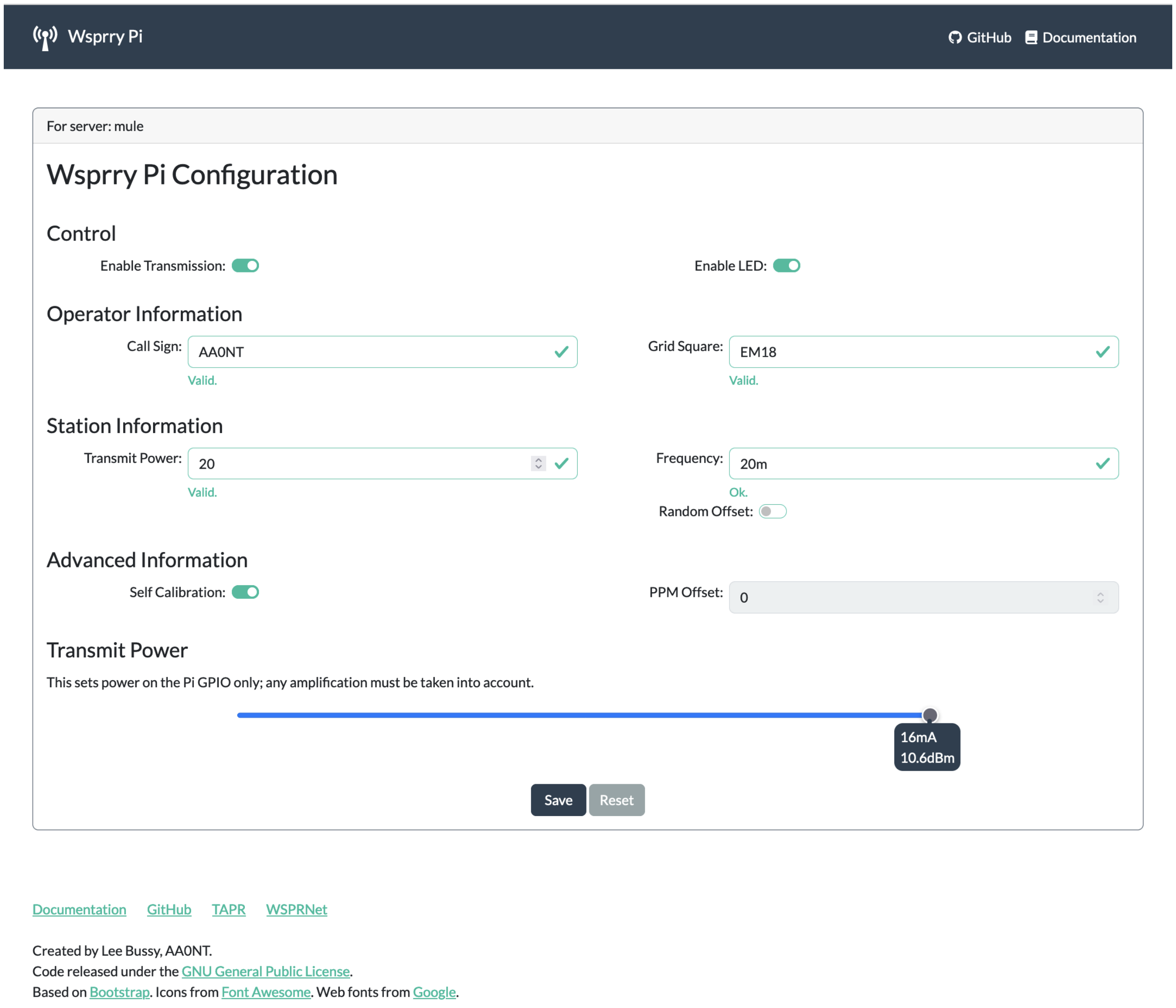
While the TAPR Hat is optional, an antenna is not. Choosing an antenna is beyond the scope of this documentation, but you can use something as simple as a random wire connected to the GPIO4 (GPCLK0) pin, which is numbered 7 on the header.


3v3 Power 1 2 5v Power
GPIO 2 (I2C1 SDA) 3 4 5v Power
GPIO 3 (I2C1 SCL) 5 6 Ground
GPIO 4 (GPCLK0) 7 8 GPIO 14 (UART TX)
Ground 9 10 GPIO 15 (UART RX)
GPIO 17 11 12 GPIO 18 (PCM CLK)
GPIO 27 13 14 Ground
GPIO 22 15 16 GPIO 23
3v3 Power 17 18 GPIO 24
GPIO 10 (SPI0 MOSI) 19 20 Ground
GPIO 9 (SPI0 MISO) 21 22 GPIO 25
GPIO 11 (SPI0 SCLK) 23 24 GPIO 8 (SPI0 CE0)
Ground 25 26 GPIO 7 (SPI0 CE1)
GPIO 0 (EEPROM SDA) 27 28 GPIO 1 (EEPROM SCL)
GPIO 5 29 30 Ground
GPIO 6 31 32 GPIO 12 (PWM0)
GPIO 13 (PWM1) 33 34 Ground
GPIO 19 (PCM FS) 35 36 GPIO 16
GPIO 26 37 38 GPIO 20 (PCM DIN)
Ground 39 40 GPIO 21 (PCM DOUT)

1.4 Web UI Operations

All normal control operations revolve around the Wsprry Pi's configuration file. This file is manipulated by a web UI and accessed by the `wspr` executable. If wspr is running, a change to this file will take effect at the next transmission period, which is the top of an even minute for normal WSPR, or at the top of the quarter hour for -15 operations. If wspr is not actively transmitting and you change the transmit selection, the system will apply the changes at the next proper WSPR window.

The web interface is designed to be a simple-to-use yet comprehensive interface to the wspr program and configuration.



For server: mule

Wsprry Pi Configuration

Control

Enable Transmission: On

Enable LED: On

Operator Information

Call Sign: AA0NT ✓ Valid.

Grid Square: EM18 ✓ Valid.

Station Information

Transmit Power: 20 ✓ Valid.

Frequency: 20m ✓ Ok.

Random Offset: Off

Advanced Information

Self Calibration: On

PPM Offset: 0

Transmit Power

This sets power on the Pi GPIO only; any amplification must be taken into account.

16mA
10.6dBm

Documentation GitHub TAPR WSPRNet

Created by Lee Bussy, AA0NT.
Code released under the GNU General Public License.
Based on Bootstrap. Icons from Font Awesome. Web fonts from Google.

Since the wspr application runs as a system daemon, the web page saves the configuration and is picked up immediately by the wspr application.

Configuration items:

- Control
  - Enable Transmission
    - Off: While the transmission is disabled, the wspr program will still run and monitor the configuration for changes.

    * On: The wspr program will transmit based on the saved configuration.

  - Enable LED: The TAPR Pi Hat includes a red LED attached to Pin 12 (GPIO18, BCM18). When enabled, this LED will light during transmission.

- Operator Information
  - Call Sign: This should be your registered callsign. Please do not make things up or use another person's callsign. It is illegal and immoral, and you will eventually annoy someone enough for them to find and report you.
  - Grid Square: The Maidenhead Locator System coordinates, also called QTH locators, grid locators, or grid squares. Use the first four digits only; two letters followed by two numbers.
- Station Information
  - Transmit Power: Transmit power in dBm. This entry is informational for the WSPR packet and does not affect the energy used to transmit the message. Typically, this value is 10, representing the Raspberry Pi's signal power capability. If you are using the TAPR hat, it includes an amplifier raising this dBm level to 20.
  - Frequency: Frequencies can be specified as an absolute TX carrier frequency (e.g., 14097100 for 20-meter WSPR) or using one of the strings in the table below. If you use a string, the transmission will happen in the middle of the WSPR region of the selected band. You may use multiple frequencies separated by spaces; wspr will iterate through them in order. A 0 as a frequency is used in a frequency list to establish a transmission window gap. The web page will validate the entries and attempt to disallow an improper entry.

Frequency Designator	Frequency in Hz
LF	137500
LF-15*	137612.5
MF	475700
MF-15	475812.5
160m	1838100
160m-15*	1838212.5
80m	3570100
60m	5288700
40m	7040100
30m	10140200
20m	14097100
17m	18106100
15m	21096100
12m	24926100
10m	28126100
6m	50294500
4m	70092500
2m	144490500

*The `-15` suffix indicates the WSPR-15 region of the band.

- Random Offset: Add a random frequency offset to each transmission:
  - +/- 80 Hz for WSPR
  - +/- 8 Hz for WSPR-15
- Advanced Information

  - Self Calibration: This is an exclusive setting with PPM Offset below. If enabled (default,) wspr will use NTP to to obtain the PPM error of the Raspberry Pi's crystal.
  - PPM Offset: This is an exclusive setting with Self Calibration above. If Self Calibration is disabled, you may manually enter the configuration value for the Raspberry Pi's crystal offset.

- Transmit Power: This slider will set the transmit power to one of eight levels. This feature is for people running on experimental frequencies which need to limit transmission power. The default is 16mA / 10.6dBm. These levels are estimated at the Raspberry Pi GPIO according to Broadcom documentation. They do not consider any amplification, such as on the TAPR board. The settings are:

  0. 2mA / -3.4dBm
  1. 4mA / 2.1dBm
  2. 6mA / 4.9dBm
  3. 8mA / 6.6dBm
  4. 10mA / 8.2dBm
  5. 12mA / 9.2dBm
  6. 14mA / 10.0dBm
  7. 16mA / 10.6dBm

- Save: Applies the currently displayed configuration. If any fields fail the basic checks, the field will be red and not allow saving. The changes may take time to change what wspr is doing; if wspr is actively transmitting, wspr will use the configuration change after the current transmission window ends.
- Reset: Reloads the configuration page from the system.

1.5 Command Line Operations

The Wsprry Pi executable, wspr, is controlled by the Linux `systemd` controller. It will run in the background as soon as your Pi starts up. It is a singleton application by design, meaning only one `wspr` process may be running. You must stop the daemon if you desire to have some manual control for testing or other reasons. Here are some commands you may use:

- `sudo systemctl status wspr`: Show a status page for the running daemon.
- `sudo systemctl restart wspr`: Restart the daemon and wspr with it.
- `sudo systemctl stop wspr`: Stop the daemon. The daemon will restart again upon reboot.
- `sudo systemctl start wspr`: Start the daemon if it is not running.
- `sudo systemctl disable wspr`: Disable the daemon from restarting on reboot.
- `sudo systemctl enable wspr`: Enable the daemon to start on reboot if it is disabled.

You will control the shutdown button monitor daemon like `wspr`, substituting `shutdown-button` for `wspr` above.

To run wspr from the command line, a complete listing of command line options is available by executing `(sudo) /usr/local/bin/wspr -h`:

```
Usage:
  wspr [options] callsign locator tx_pwr_dBm f1 <f2> <f3> ...
    OR
  wspr [options] --test-tone f

```

(continues on next page)

(continued from previous page)

```
Options:
  -h --help
    Print out this help screen.
  -v --version
    Show the Wsprry Pi version.
  -p --ppm ppm
    Known PPM correction to 19.2MHz RPi nominal crystal frequency.
  -s --self-calibration
    Check NTP before every transmission to obtain the PPM error of the
    crystal (default setting.)
  -f --free-running
    Do not use NTP to correct the frequency error of RPi crystal.
  -r --repeat
    Repeatedly and in order, transmit on all the specified command line
    freqs.
  -x --terminate <n>
    Terminate after completing <n> transmissions.
  -o --offset
    Add a random frequency offset to each transmission:
      +/- 80Hz for WSPR
      +/- 8Hz for WSPR-15
  -t --test-tone freq
    Output a test tone at the specified frequency. Only used for
    debugging and verifying calibration.
  -l --led
    Use LED as a transmit indicator (TAPR board).
  -n --no-delay;
    Transmit immediately without waiting for a WSPR TX window. Used for
    testing only.
  -i --ini-file
    Load parameters from an ini file. Supply path and file name.
  -D --daemon-mode
    Run with terse messaging.
  -d --power_level
    Set actual TX power, 0-7.

Frequencies can be specified either as an absolute TX carrier frequency,
or using one of the following bands:

  LF, LF-15, MF, MF-15, 160m, 160m-15, 80m, 60m, 40m, 30m, 20m,
  17m, 15m, 12m, 10m, 6m, 4m, and 2m

If you specify a band, the transmission will happen in the middle of the
WSPR region of the selected band.

The "-15" suffix indicates the WSPR-15 region of the band.

You may create transmission gaps by specifying a TX frequency of 0.
```

1.5.1 Command Line Entries for Testing

You may transmit a constant tone at a specific frequency for testing. In this example, wspr will send a tone at 780 kHz (780000 Hz):

```
wspr --test-tone 780e3
```

1.5.2 Example Usage

Remember that anything that creates a transmission will require you to use `sudo`.

```
wspr --help
```

Display a brief help screen.

```
wspr --test-tone 780e3
```

Transmit a constant test tone at 780 kHz.

```
wspr N9NNN EM10 33 20m
```

Using callsign N9NNN, locator EM10, and TX power 33 dBm, transmit a single WSPR transmission on the 20m band using NTP-based frequency offset calibration.

```
wspr --led N9NNN EM10 33 20m
```

The same as above but using the red LED on transmit (TAPR Hat) to indicate an active transmission. The LED is connected to Pin 12 (GPIO18, BCM18).

```
wspr --free-running N9NNN EM10 33 20m
```

The same as above, but without NTP calibration.

```
wspr --repeat --terminate 7 --ppm 43.17 N9NNN EM10 33 10140210 0 0 0 0
```

Transmit a WSPR transmission slightly off-center on 30m every 10 minutes for seven transmissions, using a fixed PPM correction value.

```
wspr --repeat --offset --self-calibration N9NNN EM10 33 40m
```

Transmit repeatedly on 40m, use NTP-based frequency offset calibration, and add a random frequency offset to each transmission to minimize collisions with other transmitters.

1.6 Advanced Operations

Everyone wants to know hat is under the covers. These topics cover some of the underlying controls and topics related to WSPR and `wspr` operations.

1.6.1 Transmission Timing

This software uses system time to determine the start of WSPR transmissions, so keep the system time synchronized within 1-second precision, i.e., use NTP network time synchronization or set the time manually with the `date` command. A WSPR broadcast starts on an even minute and takes 2 minutes for WSPR-2. Broadcasts will begin at minutes 0, 15, 30, and 45 and take 15 minutes for WSPR-15.

1.6.2 Calibration

The system will use NTP calibration by default and produces a frequency error of about 0.1 PPM after the Pi has temperature stabilized and the NTP loop has converged.

Frequency calibration is required to ensure the WSPR-2 transmission occurs within the narrow 200 Hz band. The reference crystal on your RPi might have a frequency error (which, in addition, is temperature dependent -1.3Hz/deg C @10MHz). You may manually correct the frequency or add a PPM correction on the command line to calibrate the transmission.

NTP Calibration

NTP automatically tracks and calculates a PPM frequency correction. If running NTP on your Pi, use the `--self-calibration` option to have this program query NTP for the latest frequency correction before each WSPR transmission. Some residual frequency errors may still be present due to delays in the NTP measurement loop, and this method works best if your Pi has been on for a long time, the crystal's temperature has stabilized, and the NTP control loop has converged.

AM Calibration

A practical way to calibrate is to tune the transmitter on the same frequency as a medium wave AM broadcast station, keep tuning until zero-beat (the constant audio tone disappears when the transmitter is precisely on the same frequency as the broadcast station), and determine the frequency difference with the broadcast station. This difference is the frequency error that can be applied for correction while tuning on a WSPR frequency.

Suppose your local AM radio station is at 780kHz. Use the `--test-tone` option to produce different tones around 780kHz (e.g., 780100 Hz) until you can successfully zero-beat the AM station. If the zero-beat tone specified on the command line is F, calculate the PPM correction required as `ppm=(F/780000-1)*1e6`. In the future, specify this value as the argument to the `--ppm` option on the command line. You can verify that the ppm value has been set correction by specifying `--test-tone 780000 --ppm \<ppm\>` on the command line and confirming that the Pi is still zero beating the AM station.

1.6.3 INI File

System daemon operations will read the `wspr.ini` file to supply execution parameters. During everyday use, there should be no reason to edit the file directly. The Wsprry Pi installer stores the file in the user data directory:

```
$ ls -al /usr/local/etc
total 12
drwxr-xr-x  2 root root 4096 Feb 18 14:51 .
drwxr-xr-x 10 root root 4096 Sep 21 19:02 ..
-rw-rw-rw-  1 root root  171 Mar  6 19:47 wspr.ini
```

The INI file is a standard INI file with which you may already be familiar. It has three sections, Control, Common, and Extended. The application will ignore blank lines and other whitespaces. The settings are in a key/value pair separated by an equals sign. Comments are allowed and delineated by a semicolon.

The web page configuration will overwrite any comments added to the file.

Here is an example file:

```
[Control]
Transmit = False ; Bool

[Common]
Call Sign = NXXX ; String of 7 or less
Grid Square = ZZ99 ; Four characters
TX Power = 20 ; Power in dBm is always a relatively low integer
Frequency = 20m ; This can be an integer with "m" on end, an integer as in "450000000",␣
↪or an exponential format like "780e3"

[Extended]
PPM = 0.0 ; Double
Self Cal = True ; Bool
Offset = False; Bool
Use LED = False ; Bool
Power Level = 7 ; 0-7, 7 is max
```

- **Control**
  - *Transmit**: A true or false Boolean, indicating whether wspr will transmit. Even when not transmitting, wspr will continue to run and react to changes in the INI.
- **Common**
  - **Call Sign**: Your registered call sign. Your callsign will be alphanumeric, contain no spaces, and be that which is assigned to you by your authorizing entity.
  - **Grid Square**: Your four-digit Maidenhead grid square. It will be two letters followed by two numbers.
  - **TX Power**:
  - **Frequency**: A string representing the frequency or a list of frequencies through which wspr will rotate. The notation can be an integer with "m" as a suffix indicating a standard band plan, an integer as in '450000000' representing the frequency in Hz, or an exponential format like '780e3'. The '-15' indicates it should use a 15-minute window in LF, MF, or 160m.
- **Extended**
  - **PPM*: A double for the PPM offset to use as a specific calibration. The compensation is not honored when in self-calibration mode.
  - **Self Cal**: A true/false Boolean indicating whether wspr should self-calibrate with the NTP clock.
  - **Offset**: = A true/false Boolean indicating whether to add a slight offset to each transmission.
  - **Use LED**: A true/false Boolean to use the LED attached to the TAPR board as a transmission indicator.
  - **Power Level**: A power level indicator, 0-7, the default is 7:
    - *0*: 2mA or -3.4dBm
    - *1*: 4mA or 2.1dBm
    - *2*: 6mA or 4.9dBm
    - *3*: 8mA or 6.6dBm
    - *4*: 10mA or 8.2dBm
    - *5*: 12mA or 9.2dBm
    - *6*: 14mA or 10.0dBm

  - ∗ *7*: 16mA or 10.6dBm

1.6.4 PWM Peripheral

The code uses the RPi PWM peripheral to time the frequency transitions of the output clock. The RPi sound system also uses this peripheral; hence, any system sound events during a WSPR transmission will interfere with WSPR transmissions. Sound can be permanently disabled by editing `/etc/modules` and commenting out the `snd-bcm2835` device.

1.6.5 RF And Electrical Considerations

The output of the Pi's PWM is a square wave, so a low-pass filter is REQUIRED. Knowing why this is required is part of learning to be a Ham. Connect a low-pass filter (via a decoupling capacitor) to GPIO4 (GPCLK0) and the Ground pin of your Raspberry Pi and connect an antenna to the LPF. The GPIO4 and GND pins are found on the main header on pins 7 and 9, respectively; the pin closest to the P1 label is pin 1, and its 3rd and 4th neighbor is pin 7 and 9, respectively. See this link for pin layout.

Examples of low-pass filters can be found here (PDF link). TAPR makes a very nice shield for the Raspberry Pi that is pre-assembled, performs the appropriate filtering for the selected band, and also amplifies the power output to 20dBm. Just connect your antenna, and you're good to go.

The expected power output is 10mW (+10dBm) in a 50 Ohm load without the TAPR hat. This power looks negligible, but when connected to a simple dipole antenna, this may result in reception reports ranging up to several thousand kilometers.

As the Raspberry Pi does not attenuate ripple and noise components from the 5V USB power supply, you should use a regulated supply with sufficient ripple suppression. You may see a supply ripple as mixing products centered around the transmit carrier, typically at 100/120Hz.

Do not expose GPIO4 to voltages or currents above the Absolute Maximum limits. GPIO4 outputs a digital clock in 3V3 logic, with a maximum current of 16mA. As no current protection is available and the system uses a DC component of 1.6V, do not short-circuit or place a resistive (dummy) load straight on the GPIO4 pin, as it may draw too much current. Instead, use a decoupling capacitor to remove the DC component when connecting the output dummy loads, transformers, antennas, etc. Do not expose GPIO4 to electrostatic voltages or voltages exceeding the 0 to 3.3V logic range; connecting an antenna directly to GPIO4 may damage your RPi due to transient voltages such as lightning or static buildup as well as RF from other transmitters operating into nearby antennas. Therefore, adding some form of isolation is recommended, e.g., using an RF transformer, a simple buffer/driver/PA stage, or two Schottky small signal diodes back-to-back.

1.7 Wsprry Pi System Internals

The system consists of the following:

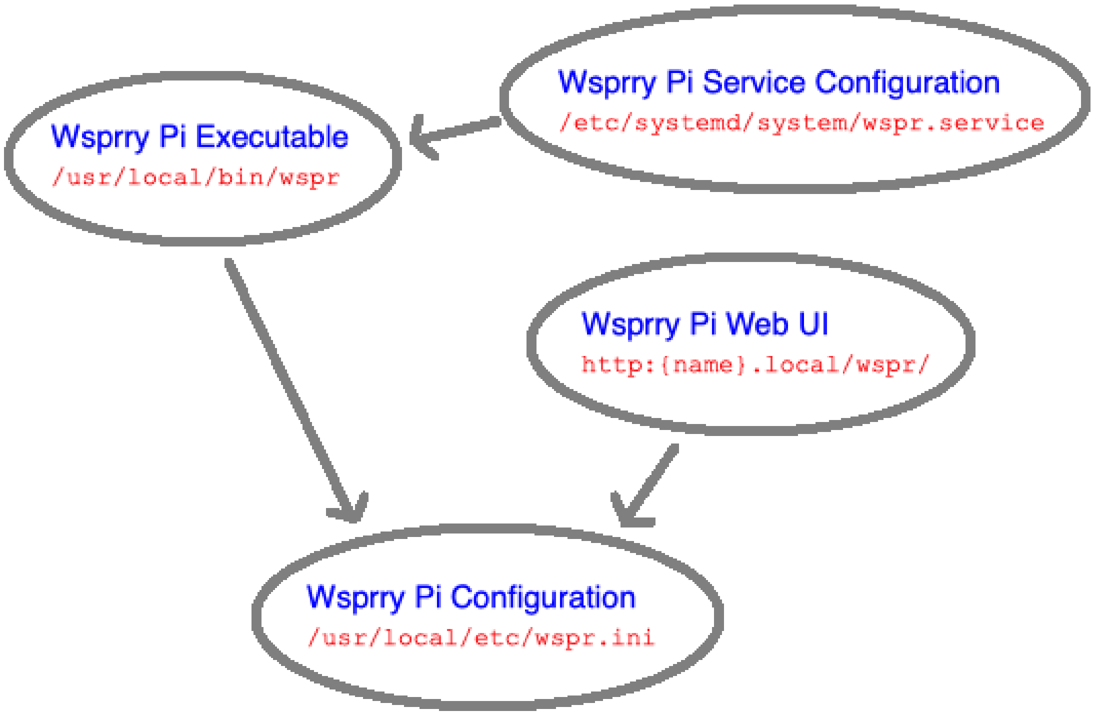
- `wspr` (executable): Installed to `/user/local/bin/wspr`
- `wspr.ini` (configuration): Installed to `/user/local/etc/wspr.ini`
- `wspr.service` (service control file): Installed to `/etc/systemd/system/wspr.service`
- Wsprry Pi Web UI: Installed to `/var/www/html/wspr`

If you choose to use the TAPR board-supplied shutdown button, the following files are also part of the system:

- `shutdown-button.py` (Python script): Installed to `/user/local/bin/shutdown-button.py`

- `shutdown-button.service` (service control file): Installed to `/etc/systemd/system/shutdown-button.service`

The primary data flow is as follows:



1.7.1 Log Files

The `wspr` service will log messages to transmission and error logs. As of version 1.1.0, these log files will move out of the system log and into four separate logs:

- `/var/log/wsprrypi/wspr.transmit.log`: The standard log showing informational messages about the wspr process and transmissions.
- `/var/log/wsprrypi/wspr.error.log`: Any errors or warnings logged by the wspr process.
- `/var/log/wsprrypi/shutdown-button.transmit.log`: If enabled, this log will show any messages from the system shutdown daemon connected to the shutdown button.
- `/var/log/wsprrypi/shutdown-button.error.log`: If enabled, this log will show any errors or warnings from the system shutdown daemon.

You may review these text files with standard Linux tools:

- `cat /var/log/wsprrypi/wspr.transmit.log`: Print the log's contents to the screen.
- `more /var/log/wsprrypi/wspr.transmit.log`: View the log using the system paging feature (`man more` for more details).

- `tail -f /var/log/wsprrypi/wspr.transmit.log`: Show the end of the log and continue to show new lines as they are logged (`man tail` for more information.)

The `logd` daemon rotates the logs on the following schedule:

- `wspr.transmit.log` and `wspr.error.log` are rotated daily, the old log is compressed, and 14 logs are retained (14 days.)
- `shutdown-button.transmit.log` and `shutdown-button.error.log` are rotated monthly (if they have contents,) the old log is compressed, and 12 logs are retained (12 months.)

1.8 Wsprry Pi Development

All current development has been done with g++ (Raspbian 10.2.1-6+rpi1) 10.2.1 20210110.

1.9 FAQ and Known Errors

1.9.1 Install Error `bash: line 1: syntax error near unexpected token `<'`

If this happens, the DNS redirect (vanity URL) I use to make the install command shorter and easier to type may have broken.

Explanation: The installation command line uses an application called `curl` to download the target URL. The pipe operator (`|`) redirects that to whatever follows, in this case, `sudo` (run as root) `bash` (the command interpreter) to make the bash script run as soon as it downloads. A regular HTML page will be sent instead of the bash script if the redirect breaks somehow. Bash doesn't know what to do with HTML (the < in the first position of the first line), so it simply refuses to do anything.

You may use the following longer and more difficult to type, command instead (one line):

```
curl -L https://raw.githubusercontent.com/lbussy/WsprryPi/main/scripts/install.sh | sudo␣
↪bash
```

1.10 Additional Reading

1.10.1 Raspberry Pi and Wsprry Pi Related

- "BCM2835 ARM Peripherals"
- "BCM2835 Audio Clocks"
- "GPIO Pads Control2"
- "Clock Management"
- "pagemap, from the userspace perspective"

1.10.2 WSPR and Other Ham References

- "WSJT: New Software for VHF Meteor-Scatter Communication." (QST, December 2001)
- "JT44: New Digital Mode for Weak Signals." (QST, June 2002)
- "EME with JT65"(QST, June 2005)
- "The JT65 Communications Protocol" (QEX, September-October 2005).
- "WSJT: Meteors, Moonbounce, and More." (Mid-Atlantic States VHF Conference, September 2005)
- "Open Source WSJT: Status, Capabilities, and Future Evolution." (12th International EME Conference, Wurzburg, August 2006)
- "Open Source WSJT." (PowerPoint presentation)
- "How Many Bits are Copied in a JT65 Transmission?"(Dubus, 3/2006)
- "Recommended Procedures for Random Digital EME." (Dubus, 3/2006)
- "MAP65: A Panoramic, Polarization-Matching Receiver for JT65." (Microwave Update, October 2007)
- "Quest for Optimum Coding and Modulation Schemes for EME." (13th International EME Conference, Florence, August 2008)
- Earth-Moon-Earth (EME) Communication (from ARRL Handbook, 2010)
- "Moonbounce from Arecibo Observatory" (QST, August 2010)
- "Frequency-Dependent Characteristics of the EME Path" (14th International EME Conference, Dallas, August 2010)
- "Frequency-Dependent Characteristics of the EME Path" (PowerPoint slides: 14th International EME Conference, Dallas, August 2010)
- "WSPRing Around the World" (QST, November 2010)
- "Meteor Scatter: How Much Antenna is Too Much?", ARRL Antenna Book, 22nd Edition (2011)
- "MAP65 Version 2: A Panoramic, Polarization-Matching Receiver for JT65" (15th International EME Conference, Cambridge, August 2012)
- "MAP65 Version 2: A Panoramic, Polarization-Matching Receiver for JT65" (PowerPoint Slides: 15th International EME Conference, Cambridge, August 2012)
- "Small Station EME at 10 and 24 GHz: GPS Locking, Doppler Correction, and JT4" (Dubus, 2/2013).
- "Optimized Small-Station EME: X-pol at 432 MHz" (16th International EME Conference, Pleumeur-Bodou, France, August 2014)
- "Optimized Small-Station EME: X-pol at 432 MHz" (PowerPoint Slides: 16th International EME Conference, Pleumeur-Bodou, France, August 2014)
- "Moonbounce at W2PU", (Mid-Atlantic States VHF Conference, September 2014).
- "EME with Adaptive Polarization at 432 MHz" (Dubus, 4/2014).
- "Open Source Soft-Decision Decoder for the JT65 (63,12) Reed-Solomon Code" (QEX, May/June 2016)
- "*WSJT-X:* New Codes, Modes, and Tools for Weak-Signal Communication" (PowerPoint Slides: 17th International EME Conference, Venice, August 2016)
- "High-Accuracy Prediction and Measurement of Lunar Echoes" (QEX, November-December 2016, pp. 36-40)
- "The MSK144 Protocol for Meteor-Scatter Communication" (QEX, September/October, 2017, pp. 8-14)
- "Work the World with WSJT-X, Part 1: Operating Capabilities", (QST, October 2017, pp. 30-36)

- “Work the World with WSJT-X, Part 2: Codes, Modes, and Cooperative Software Development”, (QST, November 2017, pp. 34-39)
- “FT8 in the ARRL RTTY Roundup”, (QST, January 2019, p 29)
- “FT4 and FT8 Contesting”, (NCJ, May/June 2020)
- “The FT4 and FT8 Communication Protocols”, (QEX, July/August 2020. pp. 3-13)