Troubleshooting

I Built It, But It Doesn't Work

Well, I'd hoped that you wouldn't ever have to get to this page, but …

I've tried very hard to make sure that everything is correct and that you're working with good parts. However, sometimes stuff happens ;-) The good news is it's all fixable (except for soldering that 40 pin connector on the wrong side of the circuit board). Follow the steps in the order given. I'm assuming that something isn't working right at this point, so let's find out what it is.

1. Check that the components are assembled on the pcb correctly and that the pcb is plugged into the rPi correctly.
2. Check that you've got the rPi Ethernet cable plugged into the rPi and that the other end of the Ethernet cable is going to your router (not your laptop).
3. Check that the red LED on the rPi turns on when power is applied.
4. Verify that the software is working by bringing up the web page (type the IP address into the address line of your browser on your PC/Mac/Linux box: 192.168.aaa.bbb). If the web page displays then the software is probably working okay. The rPi should have power for this test.

   You may have to run the WSPR_Locater program to find the IP address of the rPi and then enter the IP address into the address bar of the browser.

   If the web page doesn't display or WSPR_Locater can't find the rPi then press and hold the shutdown button for 2 sec and unplug the power supply. Wait for 30 sec and plug in the power supply. Wait for the LEDs to stop blinking and try to access the web page again. If this fails to work there may be a problem with the software load on the SD card.
5. Press the pushbutton on the AFB and hold for two seconds. Wait until the LEDs on the rPi stop flashing and unplug the power to the rPi.
6. Unplug the power supply and remove the AFB from the rPi. Look at the board. Is anything odd looking or out of place? Look at both sides of the board.
7. Check component values. Check the values of all of the resistors and capacitors.

   Resistors generally have four color bands. The first two are the resistor value. The third is the multiplier. The fourth is the tolerance. So you read, say, white-brown-red-gold as:

   white = 9, brown = 1, red = 2, gold = 5

   This is interpreted as 91 followed by 2 zeros, or 9100 ohms. The gold means the tolerance is 5%. It's really easy to look at a resistor backwards and see gold as yellow and white as silver. In that case you would read this as

   yellow = 4, red = 2, brown = 1, silver = 10

which would be interpreted as 42 followed by 1 zero, or 410 ohms and have a 10% tolerance. As you might have guessed, interpreting a 9100 ohm resistor as a 410 ohm resistor might cause a problem.

Also, as a matter of convention, we tend to avoid writing out lots of zeros. The convention for that if you're unfamiliar is a suffix. The useful suffixes are M (mega/million), K (kilo/thousand), m (milli/one one thousandth), u (micro/one one millionth), n (nano/one one billionth), p (pico/one one trillionth). Think of the suffix as a multiplier.

For example, using our 9100 ohm resistor, we would list that as 9.1K. 9.1 thousand is the same as 9100.

Second example, 0.000,000,000,680 would be written as 680p (you can see the utility in using the multiplier notation).

Make sure that you haven't read any resistor values backwards.

Capacitor values are shown as three digits. The first two digits are the value and the third digit is the multiplier (number of zeros to add). Unfortunately, you have to sorta know what multiplier to use. Capacitors are typically listed in either uF (micro farads) or pF (pico farads). If the value is given as a three digit number then it's in pF (pico farads). If it's given as a small number (typically less than 1.0), then it's in uF (micro farads).

So a capacitor shown as 681 would have a value of 680 pF (68 plus one zero) and a capacitor with a value of 0.01 would be 0.01 uF.

There are three inductors and unfortunately, they look just like resistors. The color coding is even the same as used for resistors. The units are micro henries (uH). The other unfortunate part is that the 1.2 uH inductor color coding appears to be mismarked. It's marked as a 1.0 uH inductor. It does in fact measure 1.2 uH. So the two inductors that look alike are 1.0 uH. The inductor that looks more like a dumbbell is actually 1.2 uH.

8. Verify that the components that are supposed to be aligned in a certain way are actually aligned that way. This includes the LED, diode D1, and the MOSFET Q2.

9. Okay, it's time to reassemble the circuit board to the rPi header, get our your voltmeter and turn on the power. The voltmeter ground lead should be connected to the ground test (*Gnd*) point next to the push button.

   I've listed voltages in the schematic (below) as *transmit/idle*, so for example, the voltage on the power supply rail (top wire in the schematic) is shown as (4.51/4.67). This means that I measured 4.51 volts between the power supply rail and ground (*Gnd*) while the unit was transmitting, and 4.69 volts when is was not transmitting (idle).

   Keep in mind that I'm using a cheap DVM, you're probably using a cheap DVM, and component tolerances could be all over the place. This means that if your readings are within, say 10%-20% of mine that they're probably okay.

   (See the Amplifier/Filter Board Schematic, Figure).

> I'd suggest printing out this page and measuring and recording the voltages listed. Actually, it's probably simpler to just measure the voltages (and write them down) on both sides of R5, both sides of L1, and both sides of R2.

The higher voltage across R5 is the power supply rail. It should be between 4.5-5.0 volts. The voltages on both sides of L1 should be about the same when not transmitting and one significantly lower than the other when transmitting (LED on). The voltages on both sides of R2 should be about the same whether transmitting or not. The voltages at R2 will vary depending on how pot R1 is set. The important part is that they're above 0.0 volts and have pretty much the same voltage on either side of R2.

If you find voltages that differ significantly from what's listed on the schematic, start by verifying component values and looking for bad solder joints (resolder the connections of all of the components in line with the bad voltage). If that fails, let me know because we're probably looking at a bad component, which I will be happy to replace.

Getting SSID and Passkey

Start by looking on the router for login information. Also, try the Internet for a router manual (I know, I hate to read directions, too).

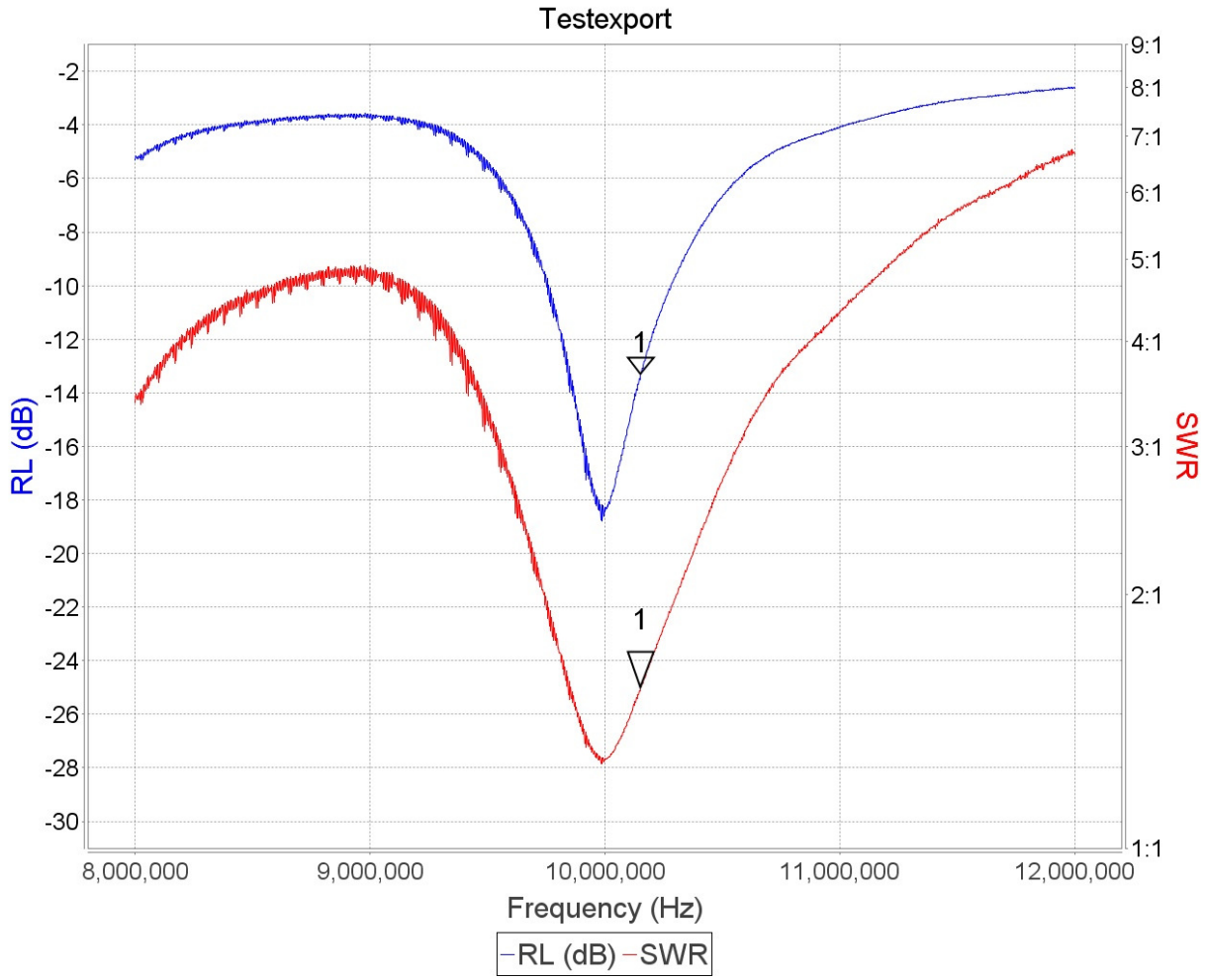
Next, you're looking for your router's web page. Open a browser and type the IP address *192.168.0.1* into the address textbox at the top (just as if you were typing www.google.com). If nothing happens try *192.168.1.1* and *192.168.2.1*.

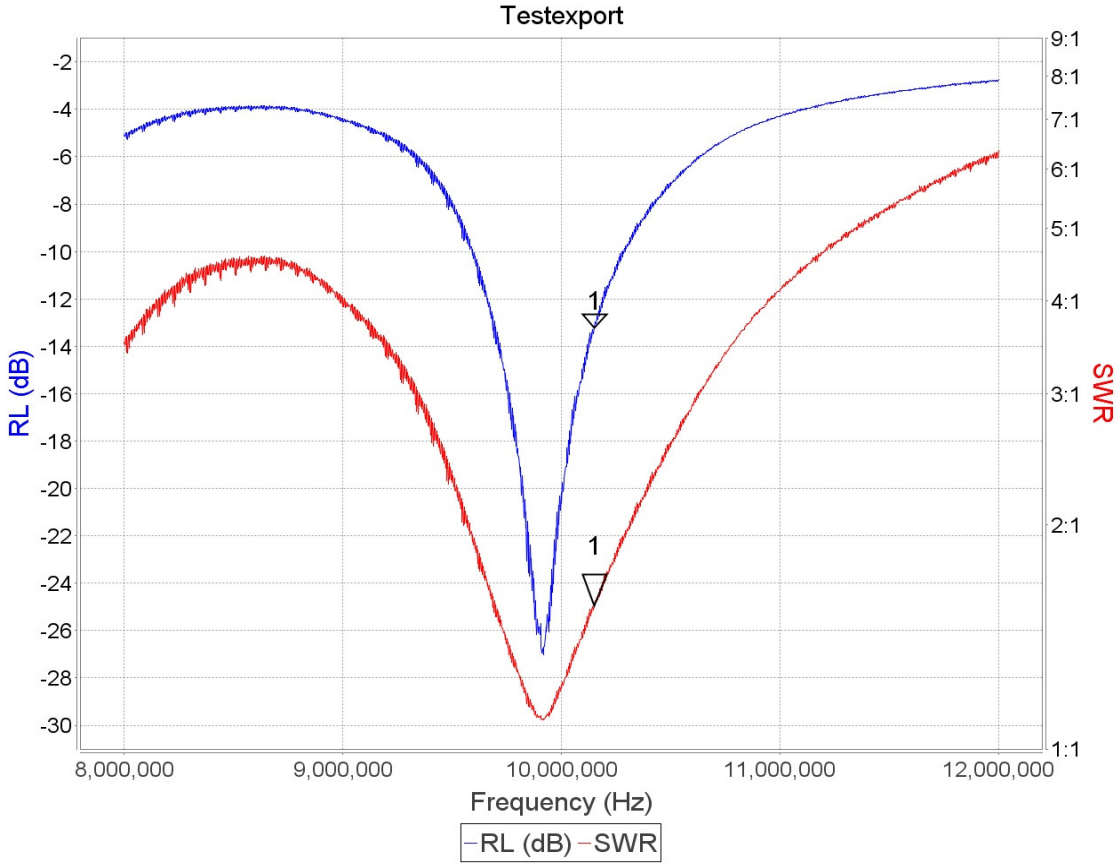
Log into the router using the username and password printed on the side of the router or in the manual.

Look through the various wireless setting for *SSID* and passkey (or something similar). I'm sorry, but I can't offer too much more because each router will be different.

Appendix 1 – Antenna Response

I measured the antenna using a miniVNATiny from miniRadioSolutions.com and got the following responses (Illustration 32).


Testexport
RL (dB)
SWR
-2
-4
-6
-8
-10
-12
-14
-16
-18
-20
-22
-24
-26
-28
-30
9:1
8:1
7:1
6:1
5:1
4:1
3:1
2:1
1:1
1
1
8,000,000
9,000,000
10,000,000
11,000,000
12,000,000
Frequency (Hz)
RL (dB)
SWR


Testexport
RL (dB)
SWR
-2
-4
-6
-8
-10
-12
-14
-16
-18
-20
-22
-24
-26
-28
-30
9:1
8:1
7:1
6:1
5:1
4:1
3:1
2:1
1:1
1
1
8,000,000
9,000,000
10,000,000
11,000,000
12,000,000
Frequency (Hz)
RL (dB)
SWR

According to the antenna formula for a dipole the optimal length for the antenna is 46 ft 2 in (468 ft/10.14 MHz = 46 ft 2 in). I chose 41 ft because that length seems to match the suboptimal elevation I'm expecting the antenna to actually be used at. My antenna was draped across several door hinges for the response plot. I'm expecting that the antenna should work adequately at low elevations. Of course, it's still not as good as an dipole at a reasonable elevation.

Appendix 2 – Programming an SD Card

Okay, something happened, and despite your best intentions somehow the SD card got messed up and you're stuck trying to reprogram it. It happens. Linux (what's running on the rPi) is a great operating system, but it is finicky about the file system (on the SD card).

Setup

1. SD Card; at least 4GB, 8GB is better. Class 10.
2. SD Card reader/writer. I've got a Targus Micro SD/T-F. It doesn't really matter. It connects your SD card to a USB port.
3. Win32DiskImager – It's a freebee. Download it from: *sourceforge.net/projects/win32diskimager/.*
4. Install Win32DiskImager.
5. Disk Image – Download it from my website – *WsprWithoutTears.com*

   Click on *WSPR Baseline link*. The screen will display a gibberish line (link to the file on DropBox). Copy this to the clipboard (highlight it with your mouse and type <CTRL>-C while it's highlighted). (<CTRL>-C means hold the control key down while pressing the C key.)

   Click in address bar of your browser (where you'd type *www.google.com*), type <CTRL>-A followed by <CTRL>-V. Press *Enter*.
6. Unzip the file after it's finished downloading (it will take a while – this is a good sized file).

Program SD Card

1. Plug the SD Card reader/writer into a USB port.
2. Plug the SD Card into the reader/writer.
3. Start Win32DiskImager.

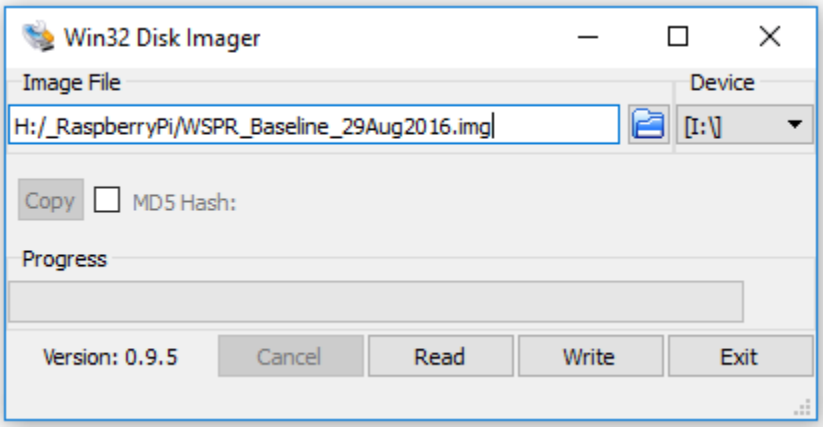


Figure 2: Win32DiskImager

4. Click on the folder icon in the upper right and find the disk image you downloaded and unzipped.
5. Click *Write* and go get a cup of coffee. Note: Win32DiskImager shows a small message box when it's done writing the SD card. Unfortunately, the message box likes to hide behind anything else you have up on the screen so you can't see it. Be forewarned that you might have to go looking for it.
6. Click *Exit*.
7. Remove the SD card and install it into the rPi.

If you haven't annoyed the gods of the Internet too much today you should be in luck and everything will work!

Appendix 3 – Communicating With the rPi over SSH

This sorta seems like the antithesis of what I've been trying to accomplish. I wanted to hide the hard details so that you could concentrate on getting a WSPR signal on the air. Okay, you're not afraid of Linux and you want to talk to your rPi over SSH. Actually, I sympathize – that's my favorite mode for talking to the rPi.

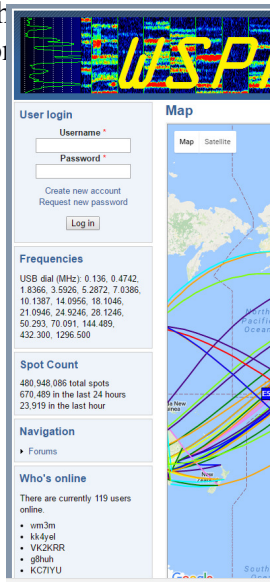
1. Download the program KiTTY.exe (no, not a cat) as part of the *WSPR Executables* from WsprWithoutTears.com or from http://www.9bis.net/kitty/?page=Download.
2. Get the IP address of the rPi from WSPR_Locater.
3. Run the KiTTY and enter the IP address for the rPi.
4. Username: ***pi*** Password: ***wspr***

Appendix 4 – Postprocessing Data

The first thing you're going to want to do once your WSPR kit is up and running is to see how your signal is getting out. Go to the website *www.wsprnet.org* (enter *www.wsprnet.org* in a browser) and click on *Map* in the upper right hand corner.

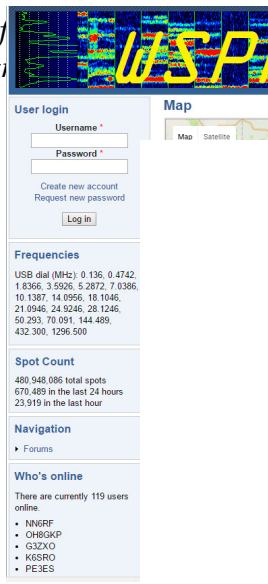
WSPRnet Map

Scroll down to the bottom of the graph and change Band to 30m, insert your *callsign*, change th
Period to 3 hours, click on the *Day/Night* overlay checkbox and then click on the *Update* buttor
can pan and zoom to see how far your signal has gotten.

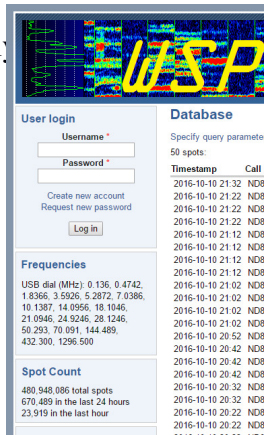


Here's a typical view.

Now click on the *Database* button (next to *Map*) in the upper right hand corner. Click on *Specif parameters* just below Database in the upper left and change *Band* to 30m and add your *callsig* the *Update* button.



This gives the particulars for the contacts – their gridsquares and the distance traveled. (Multipl 0.62 to get miles.)



Downloading Data

I'm fascinated to see how far my signal goes using just a piece of wire strung around the room (that's all I've done with my antenna). It's also interesting to see the effect of daytime vs nighttime on propagation.

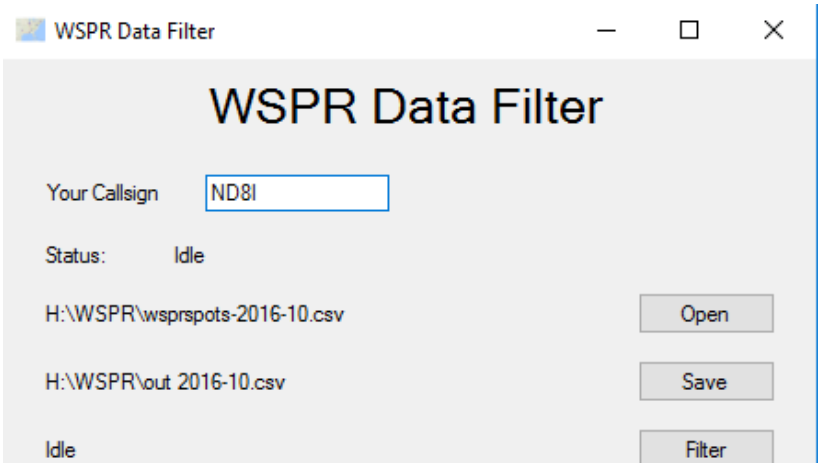
After you've run your beacon for a while you might be tempted to play with the data some more. The WSPRnet people have stored information on all of the transmissions in files that you can download and process. This is under the *Downloads* tab. The files are stored by *year-month*. They are available in gzip or zip format.

Pick a month that you'd like to investigate and click on the word *zip*. I'm amused by what can only be described as wishful thinking on the part of the people who put together the web page. The writeup at the top says “Compressed file sizes range from 1-20MB.” The compressed files will be much closer to 300 MB and uncompressed go to about 1.4 GB.

Most of the entries were not information I cared about – they list *all* of the contacts in a particular month. I only want to see my contacts.

WSPR Data Filter

I wrote a program to extract only my contacts. The program is called *WSPR_Data_Filter.exe* and can be downloaded from the web site. The program removes everything but your contacts from the downloaded data.



Here is how you use it -

1. Download a month's worth of data from WSPRnet.org.
2. Unzip the file (these are big files – it will take some time).
3. Download Run WSPR_Data_Filter (download from http://Wspr.WithoutTearscom/, click on *WSPR Executables* and extract the program from the .Zip file).
4. Run WSPR_Data_Filter.
5. Enter your *callsign*
6. Click the *Open* button, select the file you unpacked, and click *Open*.
7. Click the *Save* button, enter a *filename* to save the output as, and click *Save*.
8. Click the *Filter* button.
9. Click the *Exit* button when processing is done.

The output can be read into any spreadsheet program (e.g. Excel) directly for processing and graphing. I have written some post processing software using Octave to give a broader picture of your data (see below).