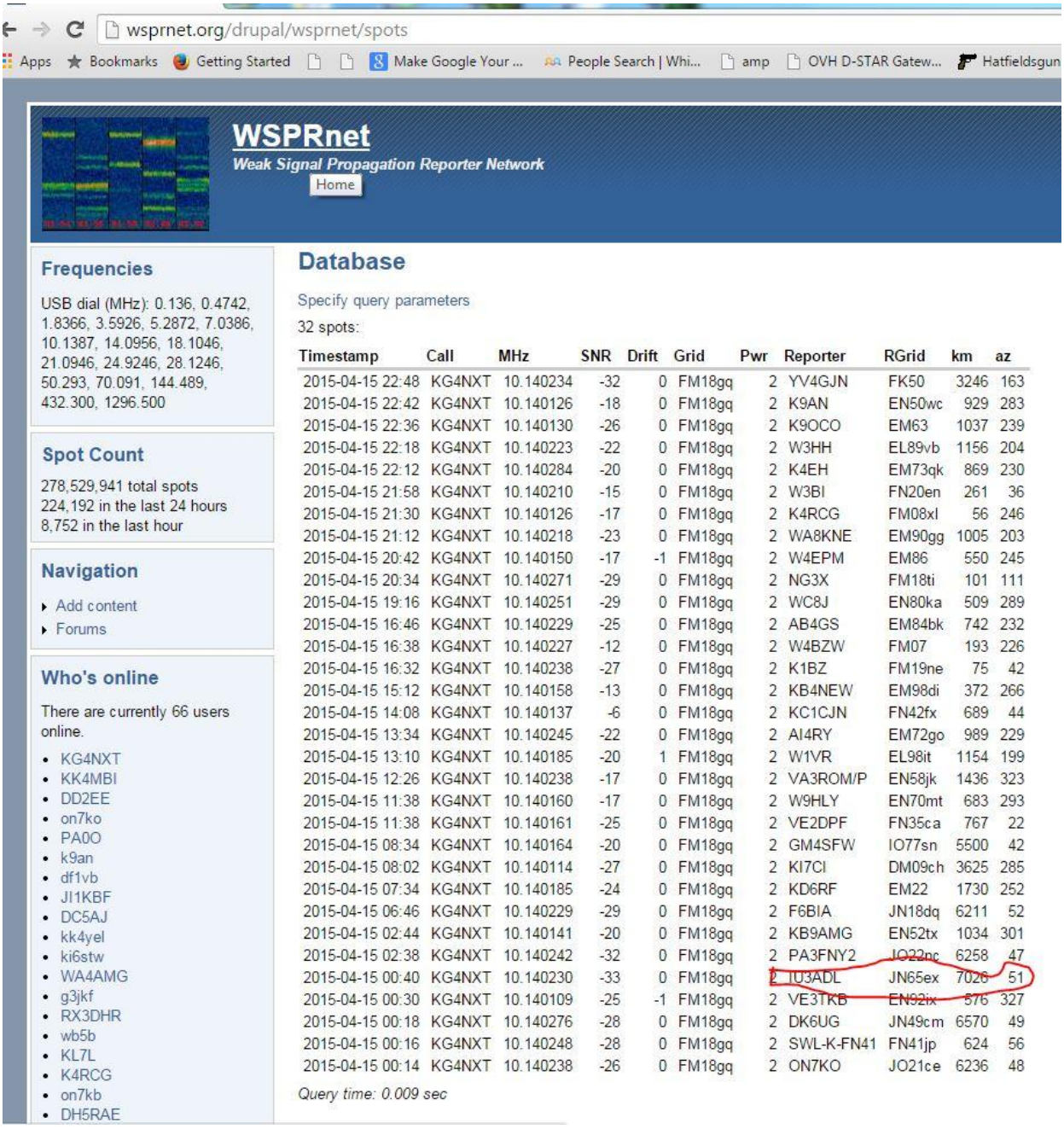

wsprnet.org/drupal/wsprnet/spots
Apps
Bookmarks
Getting Started
Make Google Your ...
People Search | Whi...
amp
OVH D-STAR Gatew...
Hatfieldsgun
WSPRnet
Weak Signal Propagation Reporter Network
Home
Frequencies
USB dial (MHz): 0.136, 0.4742, 1.8366, 3.5926, 5.2872, 7.0386, 10.1387, 14.0956, 18.1046, 21.0946, 24.9246, 28.1246, 50.293, 70.091, 144.489, 432.300, 1296.500
Spot Count
278,529,941 total spots
224,192 in the last 24 hours
8,752 in the last hour
Navigation
Add content
Forums
Who's online
There are currently 66 users online.
KG4NXT
KK4MBI
DD2EE
on7ko
PA0O
k9an
df1vb
JI1KBF
DC5AJ
kk4yel
ki6stw
WA4AMG
g3jkf
RX3DHR
wb5b
KL7L
K4RCG
on7kb
DH5RAE
Database
Specify query parameters
32 spots:
Timestamp Call MHz SNR Drift Grid Pwr Reporter RGrid km az
2015-04-15 22:48 KG4NXT 10.140234 -32 0 FM18gq 2 YV4GJN FK50 3246 163
2015-04-15 22:42 KG4NXT 10.140126 -18 0 FM18gq 2 K9AN EN50wc 929 283
2015-04-15 22:36 KG4NXT 10.140130 -26 0 FM18gq 2 K9OCO EM63 1037 239
2015-04-15 22:18 KG4NXT 10.140223 -22 0 FM18gq 2 W3HH EL89vb 1156 204
2015-04-15 22:12 KG4NXT 10.140284 -20 0 FM18gq 2 K4EH EM73qk 869 230
2015-04-15 21:58 KG4NXT 10.140210 -15 0 FM18gq 2 W3BI FN20en 261 36
2015-04-15 21:30 KG4NXT 10.140126 -17 0 FM18gq 2 K4RCG FM08xl 56 246
2015-04-15 21:12 KG4NXT 10.140218 -23 0 FM18gq 2 WA8KNE EM90gg 1005 203
2015-04-15 20:42 KG4NXT 10.140150 -17 -1 FM18gq 2 W4EPM EM86 550 245
2015-04-15 20:34 KG4NXT 10.140271 -29 0 FM18gq 2 NG3X FM18ti 101 111
2015-04-15 19:16 KG4NXT 10.140251 -29 0 FM18gq 2 WC8J EN80ka 509 289
2015-04-15 16:46 KG4NXT 10.140229 -25 0 FM18gq 2 AB4GS EM84bk 742 232
2015-04-15 16:38 KG4NXT 10.140227 -12 0 FM18gq 2 W4BZW FM07 193 226
2015-04-15 16:32 KG4NXT 10.140238 -27 0 FM18gq 2 K1BZ FM19ne 75 42
2015-04-15 15:12 KG4NXT 10.140158 -13 0 FM18gq 2 KB4NEW EM98di 372 266
2015-04-15 14:08 KG4NXT 10.140137 -6 0 FM18gq 2 KC1CJN FN42fx 689 44
2015-04-15 13:34 KG4NXT 10.140245 -22 0 FM18gq 2 AI4RY EM72go 989 229
2015-04-15 13:10 KG4NXT 10.140185 -20 1 FM18gq 2 W1VR EL98it 1154 199
2015-04-15 12:26 KG4NXT 10.140238 -17 0 FM18gq 2 VA3ROM/P EN58jk 1436 323
2015-04-15 11:38 KG4NXT 10.140160 -17 0 FM18gq 2 W9HLY EN70mt 683 293
2015-04-15 11:38 KG4NXT 10.140161 -25 0 FM18gq 2 VE2DPF FN35ca 767 22
2015-04-15 08:34 KG4NXT 10.140164 -20 0 FM18gq 2 GM4SFW IO77sn 5500 42
2015-04-15 08:02 KG4NXT 10.140114 -27 0 FM18gq 2 KI7CI DM09ch 3625 285
2015-04-15 07:34 KG4NXT 10.140185 -24 0 FM18gq 2 KD6RF EM22 1730 252
2015-04-15 06:46 KG4NXT 10.140229 -29 0 FM18gq 2 F6BIA JN18dq 6211 52
2015-04-15 02:44 KG4NXT 10.140141 -20 0 FM18gq 2 KB9AMG EN52tx 1034 301
2015-04-15 02:38 KG4NXT 10.140242 -32 0 FM18gq 2 PA3FNY2 JO22nc 6258 47
2015-04-15 00:40 KG4NXT 10.140230 -33 0 FM18gq 2 IU3ADL JN65ex 7026 51
2015-04-15 00:30 KG4NXT 10.140109 -25 -1 FM18gq 2 VE3TKB EN92ix 576 327
2015-04-15 00:18 KG4NXT 10.140276 -28 0 FM18gq 2 DK6UG JN49cm 6570 49
2015-04-15 00:16 KG4NXT 10.140248 -28 0 FM18gq 2 SWL-K-FN41 FN41jp 624 56
2015-04-15 00:14 KG4NXT 10.140238 -26 0 FM18gq 2 ON7KO JO21ce 6236 48
Query time: 0.009 sec

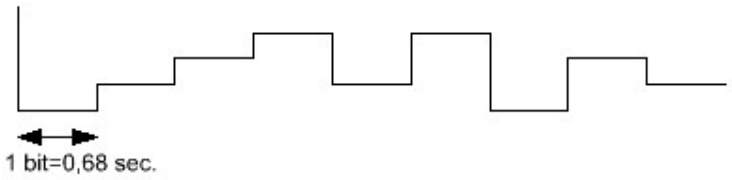
WSPR

How does WSPR work?

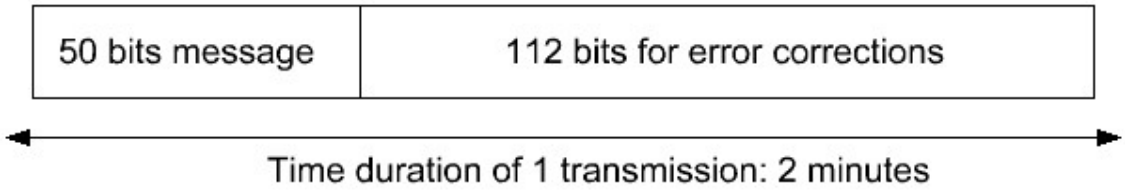
WSPR does work with tones with a length of 0,68 seconds. There are 4 different tone frequencies. (4 frequency shifts), but the difference is minimal, only 1,46 Hz! You do not hear those differences. A WSPR signal sounds like a continuous tone. A standard WSPR beacon transmission consists a call, a QTH locator and an indication for the transmit power. This message has a length of 50 bits.That it is possible to receive weak WSPR signals, is not only due to the long tone lengths. The message with a length of 50 bits is namely extended with even 112 bits, so even more than 2 times the number of bits of the original message. Totally no 50, but 162 bits are transmitted. And with these extra bits it is possible to detect and correct all kinds of errors in the 50 bits of the standard message. Similar techniques are used for the correction of errors in the data of memories and harddisks of our PC. And the bitorder is also changed to improve the error correction for interferences of the signal due to long fading periods. One long, uncorrectable error is converted in many short, correctable errors of the signal.

For every band, a piece of the spectrum of 200 Hz width is assigned. A WSPR signal is only 6 Hz wide, so 33 WSPR signals do fit in this 200 Hz band. And because a WSPR station does transmit only 20% of the time and receives 80% of the time, the total number of active transmitters can be 5x as much or 165.

WSPR SIGNAL



4 frequency shifts
1 shift is 1,46 Hz



Raspberry Pi bareback LF/MF/HF/VHF WSPR transmitter

Makes a very simple WSPR beacon from your RasberryPi by connecting GPIO
port to Antenna (and LPF), operates on LF, MF, HF and VHF bands from
0 to 250 MHz.

******
Installation / update:
******
  Make sure you are using the latest kernel by updating your system. The latest
  kernel includes fixes wich improve NTP ppm measurement accuracy:
    sudo apt-get update
    sudo apt-get dist-upgrade

  Download and compile code:
    sudo apt-get install git
    git clone https://github.com/JamesP6000/WsprryPi.git
    cd WsprryPi
    make

  See the accompanying BUILD file for more details.

******
Usage: (WSPR --help output):
******
  Usage:
    wspr [options] callsign locator tx_pwr_dBm f1 <f2> <f3> ...
      OR
    wspr [options] --test-tone f

  Options:
    -h --help
      Print out this help screen.
    -p --ppm ppm
      Known PPM correction to 19.2MHz RPi nominal crystal frequency.
    -s --self-calibration
      Call ntp_adjtime() before every transmission to obtain the PPM error of the xtal.
    -r --repeat
      Repeatedly, and in order, transmit on all the specified freqs.
    -x --terminate <n>
      Terminate after n transmissions have been completed.
    -o --offset
      Add a random frequency offset to each transmission:
        +/- 80 Hz for WSPR
        +/- 8 Hz for WSPR-15
    -t --test-tone freq
      Simply output a test tone and the specified frequency. Only used
      for debugging and to verify calibration.
    -n --no-delay
      Transmit immediately, do not wait for a WSPR TX window. Used
      for testing only.

  Frequencies can be specified either as an absolute TX carrier frequency, or
  using one of the following strings. If a string is used, the transmission
  will happen in the middle of the WSPR region of the selected band.
    LF LF-15 MF MF-15 160m 160m-15 80m 60m 40m 30m 20m 17m 15m 12m 10m 6m 4m 2m
  <B>-15 indicates the WSPR-15 region of band <B>.

  Transmission gaps can be created by specifying a TX frequency of 0

******
Radio licensing / RF:
******
  In order to transmit legally, a HAM Radio License is REQUIRED for running
  this experiment. The output is a square wave so a low pass filter is REQUIRED.
  Connect a low-pass filter (via decoupling C) to GPIO4 (GPCLK0) and Ground pin
  of your Raspberry Pi, connect an antenna to the LPF. The GPIO4 and GND pins
  are found on header P1 pin 7 and 9 respectively, the pin closest to P1 label
  is pin 1 and its 3rd and 4th neighbour is pin 7 and 9 respectively. See this
  link for pin layout: http://elinux.org/RPi_Low-level_peripherals Examples of
  low-pass filters can be found here: http://www.gqrp.com/harmonic_filters.pdf

  The expected power output is 10mW (+10dBm) in a 50 Ohm load. This looks
  neglible, but when connected to a simple dipole antenna this may result in
  reception reports ranging up to several thousands of kilometers.

  As the Raspberry Pi does not attenuate ripple and noise components from the
  5V USB power supply, it is RECOMMENDED to use a regulated supply that has
  sufficient ripple supression. Supply ripple might be seen as mixing products
  products centered around the transmit carrier typically at 100/120Hz.

  DO NOT expose GPIO4 to voltages or currents that are above the specified
  Absolute Maximum limits. GPIO4 outputs a digital clock in 3V3 logic, with a
  maximum current of 16mA. As there is no current protection available and 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 DC component when connecting the output dummy
  loads, transformers, antennas, etc. DO NOT expose GPIO4 to electro- static
  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 it is RECOMMENDED to add some form
  of isolation, e.g. by using a RF transformer, a simple buffer/driver/PA
  stage, two schottky small signal diodes back to back.

******

TX Timing:
******
  This software is using system time to determine the start of WSPR
  transmissions, so keep the system time synchronised within 1sec precision,
  i.e. use NTP network time synchronisation or set time manually with date
  command. A WSPR broadcast starts on an even minute and takes 2 minutes for
  WSPR-2 or starts at :00,:15,:30,:45 and takes 15 minutes for WSPR-15. It
  contains a callsign, 4-digit Maidenhead square locator and transmission
  power. Reception reports can be viewed on Weak Signal Propagation Reporter
  Network at: http://wsprnet.org/drupal/wsprnet/spots

******
Calibration:
******
  Frequency calibration is REQUIRED to ensure that the WSPR-2 transmission
  occurs within the narrow 200 Hz band. The reference crystal on your RPi might
  have an frequency error (which in addition is temp. dependent -1.3Hz/degC
  @10MHz). To calibrate, the frequency might be manually corrected on the
  command line or a PPM correction could be specified on the command line.

  NTP calibration:
  NTP automatically tracks and calculates a PPM frequency correction. If you
  are running NTP on your Pi, you can use the --self-calibration option to
  have this program querry NTP for the latest frequency correction before
  each WSPR transmission. Some residual frequency error 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
  of a medium wave AM broadcast station; keep tuning until zero beat (the
  constant audio tone disappears when the transmitter is exactly on the same
  frequency as the broadcast station), and determine the frequency difference
  with the broadcast station. This 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 (eg 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 comman 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.

******
PWM Peripheral:
******
  The code uses the RPi PWM peripheral to time the frequency transitions
  of the output clock. This peripheral is also used by the RPi sound system
  and hence any sound events that occur 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.

******
Example usage:
******
  Brief help screen
    wspr --help

  Transmit a constant test tone at 780 kHz.
    sudo wspr --test-tone 780e3

  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.
    sudo wspr --self-calibration N9NNN EM10 33 20m

  Transmit a WSPR transmission slightly off-center on 30m every 10 minutes for
  a total of 7 transmissions, and using a fixed PPM correction value. sudo
    wspr --repeat --terminate 7 --ppm 43.17 N9NNN EM10 33 10140210 0 0 0 0

  Transmit repeatedly on 40m, use NTP based frequency offset calibration,
  and add a random frequency offset to each transmission to minimize collisions
  with other transmissions.
    sudo wspr --repeat --offset --self-calibration N9NNN EM10 33 40m

******
Reference documentation:
******
  http://www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf
  http://www.scribd.com/doc/127599939/BCM2835-Audio-clocks
  http://www.scribd.com/doc/101830961/GPIO-Pads-Control2
  https://github.com/mgottschlag/vctools/blob/master/vcdb/cm.yaml
  https://www.kernel.org/doc/Documentation/vm/pagemap.txt

******
Credits:
******
  Credits goes to Oliver Mattos and Oskar Weigl who implemented PiFM [1]
  based on the idea of exploiting RPi DPLL as FM transmitter.

  Dan MD1CLV combined this effort with WSPR encoding algorithm from F8CHK,
  resulting in WsprryPi a WSPR beacon for LF and MF bands.

  Guido PE1NNZ <pe1nnz@amsat.org> extended this effort with DMA based PWM
  modulation of fractional divider that was part of PiFM, allowing to operate
  the WSPR beacon also on HF and VHF bands. In addition time-synchronisation
  and double amount of power output was implemented.

James Peroulas <james@peroulas.com> added several command line options, a
makefile, improved frequency generation precision so as to be able to
precisely generate a tone at a fraction of a Hz, and added a self calibration
feature where the code attempts to derrive frequency calibration information
from an installed NTP deamon. Furthermore, the TX length of the WSPR symbols
is more precise and does not vary based on system load or PWM clock
frequency.

Michael Tatarinov for adding a patch to get PPM info directly from the
kernel.

[1] PiFM code from
    http://www.icrobotics.co.uk/wiki/index.php/Turning_the_Raspberry_Pi_Into_an_FM_Transmitter
[2] Original WSPR Pi transmitter code by Dan:
    https://github.com/DanAnkers/WsprryPi
[3] Fork created by Guido:
    https://github.com/threeme3/WsprryPi
[4] This fork created by James:
    https://github.com/JamesP6000/WsprryPi