

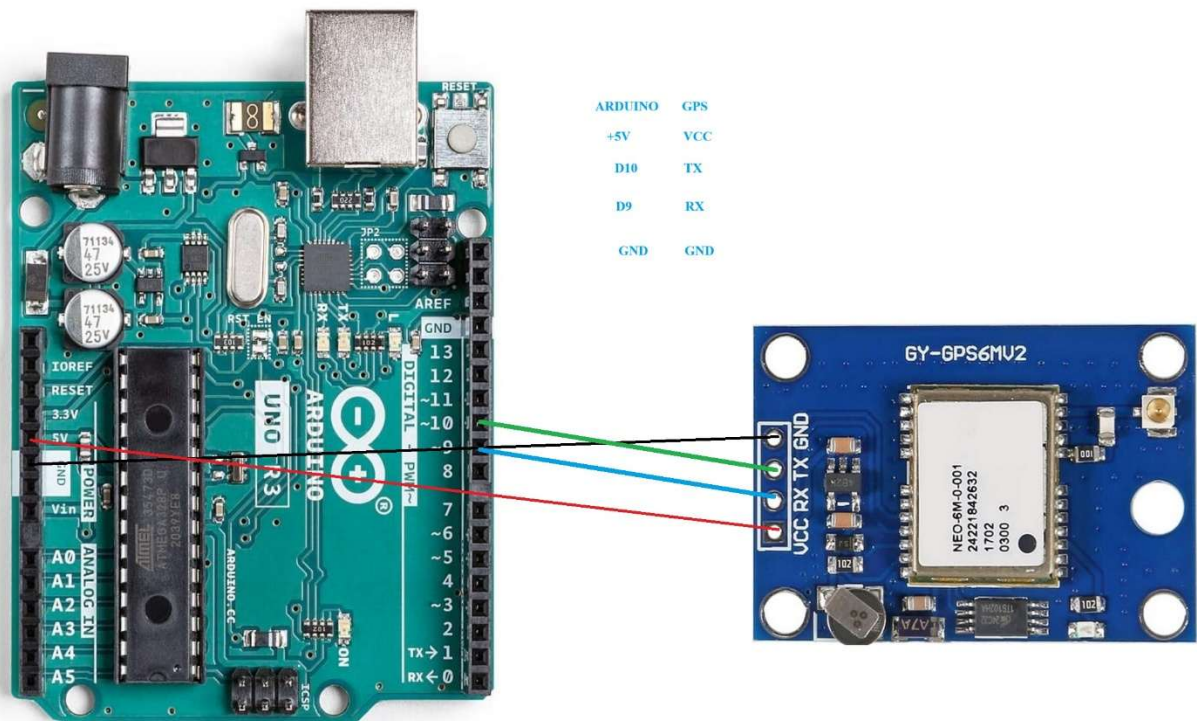
## HOW TO CONVERT DX UnO TRANSCEIVER INTO 5 BAND STANDALONE WSPR TRANSCEIVER

By only adding a GPS MODULE and replacing firmware of DX UnO with Standalone WSPR firmware we can convert DX UnO to a Stand Alone WSPR transceiver that can transmit on 20m,17m,15m,12m and 10m Bands WSPR frequency back-to-back in 2 minutes intervals.

For this modification we need to do:

- 1- HARDWARE MODIFICATION
- 2- FIRMWARE MODIFICATION

### 1- HARDWARE MODIFICATION:



For HARDWARE MODIFICATION a GPS Module need to be added. This GPS MODULE helps in two points. Getting the maiden grid location and Time synchronization.

Any GPS MODULE will work as long as it's a +5V module and we know the baud rate of GPS UART.

For adding GPS Module there needs to be 4 main connections between GPS Module and Arduino Uno. These are:

- 1- GPS GND to ARDUINO UNO GND
- 2- GPS VCC to ARDUINO UNO +5V
- 3- GPS TX to Arduino Uno D10 Pin
- 4- GPS RX to Arduino Uno D9 Pin

## 2- FIRMWARE MODIFICATION:

To achieve standalone DX UnO WSPR TRANSCEIVER, we need to replace DX UnO Firmware with Standalone WSPR firmware.

DX UnO Standalone WSPR firmware is modified WSPR Multiband firmware by Roel Kroes, <https://github.com/RoelKroes/wsprbeacon>

I modified slightly this Arduino sketch to work with DX UnO.

### 1- Download:

DX\_UnO\_WSPR\_MULTIBAND\_V1.0.ino,  
GPS.ino  
Messages.ino  
Settings.h

files from DX UnO github page and save all in the same directory

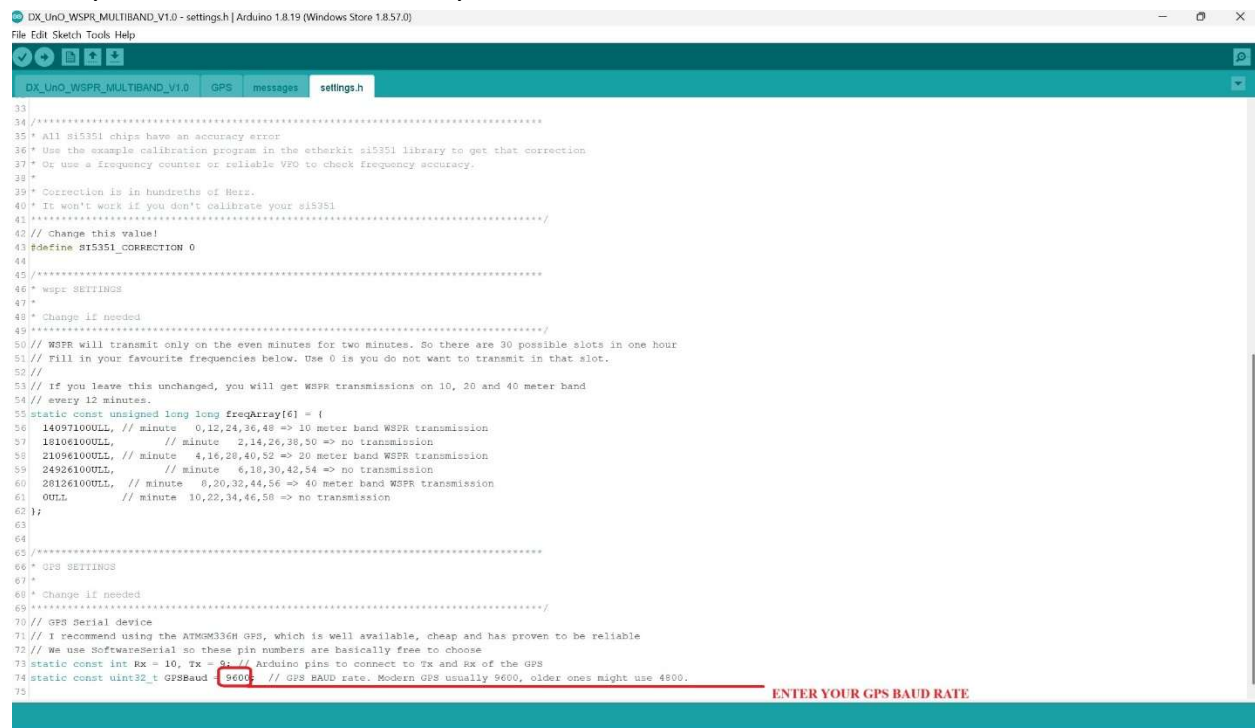
- 2- Modify settings.h after opening all files on Arduino IDE like in below photos:
- Enter Your Callsign
  - Enable Debug mode if you wish to monitor what is Standalone WSPR doing with Serial monitor. If not needed just comment it.

```
1/* *****
2 * Use serial console output for debugging and development
3 *
4 * Change if needed
5*****
6// Comment out if you do not want Serial console output
7// Comment out DEVMODE in a production environment as it will degrade performance!
8// #define DEVMODE
9
10
11// Your own HAM call. Change it
12#define MYCALL "XXXXXX" // REPLACE XXXXX with your CALL SIGN
13
14// The power of your transmission in dbm.
15// for the DX Uno this should be set to 24 (=400 milliwatts).
16#define DBMPOWER 24
17
18
19// If you have a valid GPS lock, the software will make this HIGH twice for 20ms
20// You could connect a LED to it.
21// Set LED_PIN to 0 if you do not want to use a pin
22#define LED_PIN 11
23
24// If you have an Arduino on 10MHz, set this to 10672
25// If you have an Arduino on 8MHz, set this to 5336
26#define WSPR_CTC 10672
27
28// Frequency of the SI5351 Oscillator in Hertz
29// for example #define SI5351FREQ 26000000 if you have a 26MHz XO
30// Use 0 if you have a 25MHz Oscillator
31#define SI5351FREQ 26000000
32
33
34/* *****
35 * All SI5351 chips have an accuracy error
36 * Use the example calibration program in the etherkit si5351 library to get that correction
37 * Or use a frequency counter or reliable VFO to check frequency accuracy.
38 *
39 * Correction is in hundredths of Hertz.
40 * It won't work if you don't calibrate your si5351
41*****
42// Change this value!
43#define SI5351_CORRECTION 0
```

Done Saving.

Arduino Pro or Pro Mini, ATmega328P (1.3V, 8 MHz) on COM6

## Modify GPS Baud Rate inline with your GPS baud rate.



```
33
34 /*****
35 * All Si5351 chips have an accuracy error
36 * Use the example calibration program in the etherkit si5351 library to get that correction.
37 * Or use a frequency counter or reliable VFO to check frequency accuracy.
38 *
39 * Correction is in hundredths of Hertz.
40 * It won't work if you don't calibrate your Si5351
41 *****/
42 // Change this value!
43 #define SI5351_CORRECTION 0
44
45 /*****
46 * WSPR SETTINGS
47 *
48 * Change if needed
49 *****/
50 // WSPR will transmit only on the even minutes for two minutes. So there are 30 possible slots in one hour.
51 // Fill in your favourite frequencies below. Use 0 is you do not want to transmit in that slot.
52 //
53 // If you leave this unchanged, you will get WSPR transmissions on 10, 20 and 40 meter band
54 // every 12 minutes.
55 static const unsigned long long freqArray[] = {
56   14097100ULL, // minute 0,12,24,36,48 => 10 meter band WSPR transmission
57   18106100ULL, // minute 2,14,26,38,50 => no transmission
58   21096100ULL, // minute 4,16,28,40,52 => 20 meter band WSPR transmission
59   24926100ULL, // minute 6,18,30,42,54 => no transmission
60   28126100ULL, // minute 8,20,32,44,56 => 40 meter band WSPR transmission
61   0ULL, // minute 10,22,34,46,58 => no transmission
62 };
63
64
65 /*****
66 * GPS SETTINGS
67 *
68 * Change if needed
69 *****/
70 // GPS Serial device
71 // I recommend using the ATWGM336H GPS, which is well available, cheap and has proven to be reliable
72 // We use SoftwareSerial so these pin numbers are basically free to choose
73 static const int Rx = 10, Tx = 2; // Arduino pins to connect to Tx and Rx of the GPS
74 static const uint32_t GPSbaud = 9600; // GPS BAUD rate. Modern GPS usually 9600, older ones might use 4800.
75
```

ENTER YOUR GPS BAUD RATE

Now upload your modified and saved DX UnO standalone WSPR firmware to your DX UnO as you upload to any Arduino Uno via Arduino IDE.

That's all needed to run DX Uno Standalone WSPR.

When you first power on DX UnO WSPR nothing will happen until GPS receives data.

When GPS starts getting GPS data Blue CAT led blinks intermittently. This means GPS data is received and valid.

When TX times comes Blue LED will be off and Red TX led will be on for 2 minutes.

At the end of TX, Red LED will be off and Blue LED will blink intermittent.

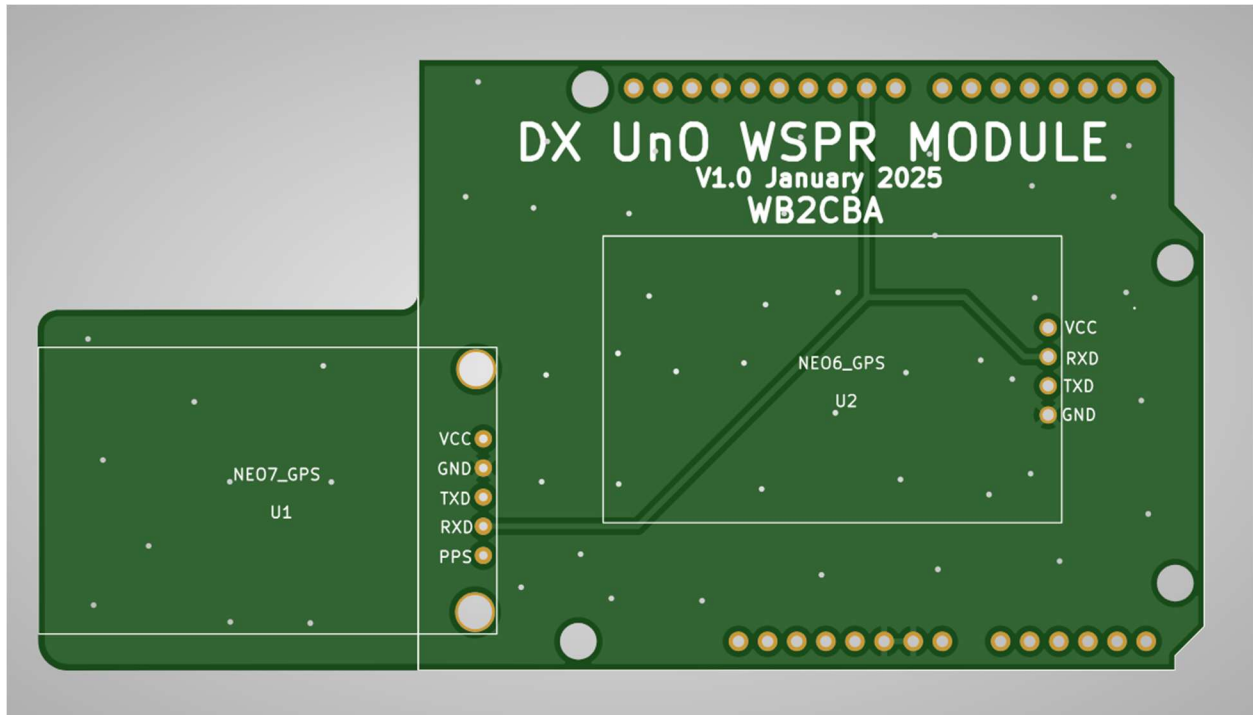
This loop continues like that. In every TX sequence DX Uno WSPR will switch to a different band.

This will give you a nice clue about your antenna efficiency in different bands.

When operating DX UnO with WSPR conversion MIC and SPK sockets will be redundant and not used so no need to connect these. The only cable connection is USB cable which will be used

only for powering up DX UnO WSPR. It can just be connected to a USB power source with +5V through Arduino Uno.

If you want this DX UnO WSPR conversion a bit neater and more elegant you can also use this PCB module which is a GPS Carrier board:



This has two GPS type you can solder on. This pcb stacks between Arduino Uno and DX UnO which creates a three pcb sandwich like rig. [Use long pin Arduino Shield headers.](#)

**Gerber file of this GPS Carrier PCB is in DX UnO Github page:**

[https://github.com/BluQRP/DX\\_UnO](https://github.com/BluQRP/DX_UnO)

Here are couple of photos illustrating how this setup looks like:

