Auto-Calibrated 40 MHz WSPR & FT8 MEPT

Gene Marcus W3PM GM4YRE

This Manned Experimental Propagation Transmitter (MEPT) uses the very popular Arduino Nano and Si5351A clock generator board to generate FT8 or WSPR signals. Frequency and time accuracy are maintained by using an inexpensive GPS receiver board. Other frequencies below 40 MHz may also be used.

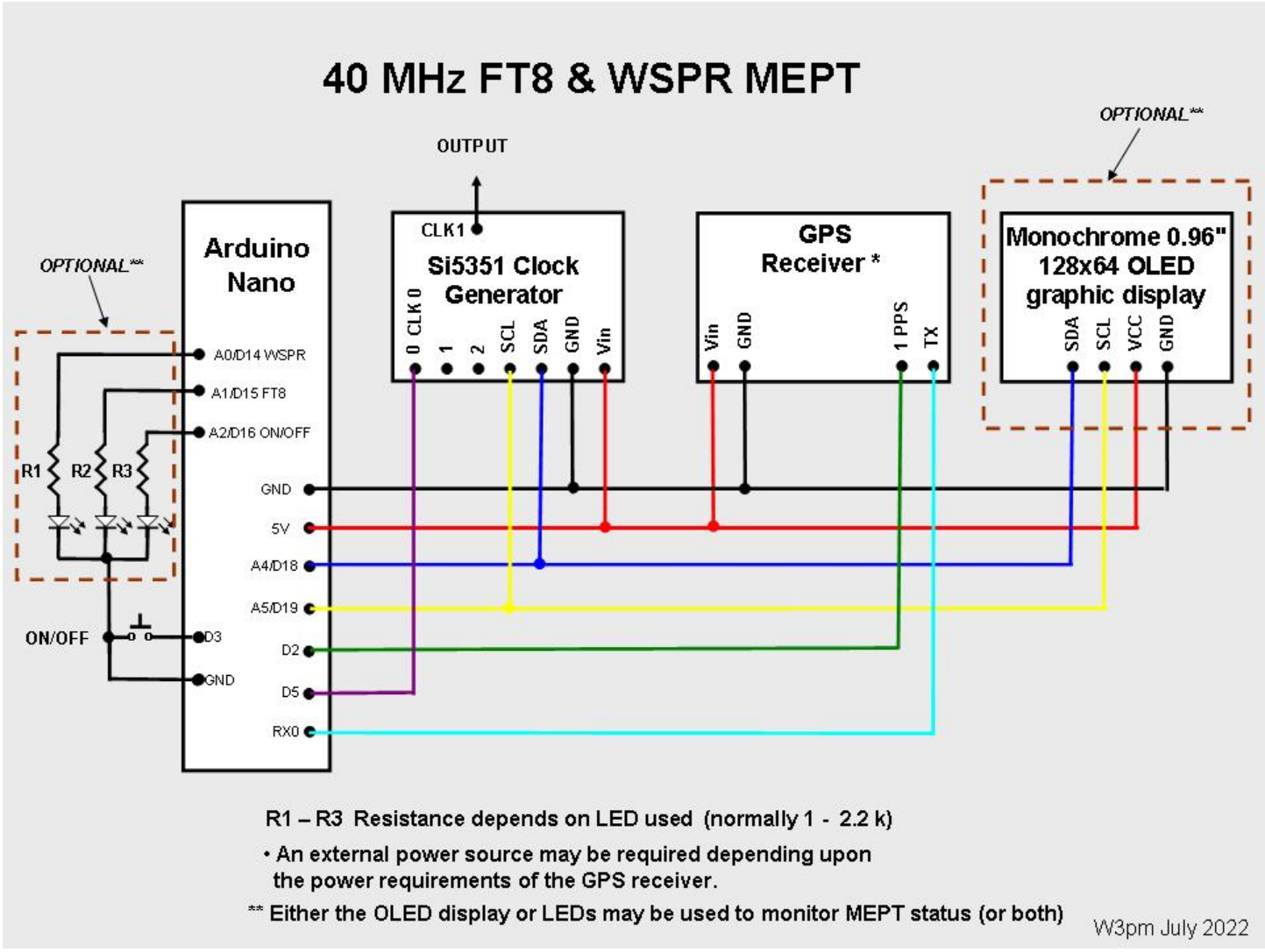

Figure 1 8 Meter GPS WSPR/FT8 MEPT

Introduction

This project was intended to be used as a very low power experimental license-free transmitter on the 40 MHz ISM band as defined by ITU Radio Regulations. The project may also be used as an exciter to a power amplifier by individuals holding experimental licenses for the 40 MHz band. It will also transmit on other frequencies below 40 MHz by including them in the “TXdialFreq” variable as outlined in the Transmit Time, Mode, and Band Scheduling section below.

As shown in Figure 1 the building blocks for this project consist of four boards: an Arduino Nano (*or Uno*) microcontroller, a Si5351A clock generator breakout board, GPS receiver, and an optional 0.96 inch 128x64 serial I2C OLED display board.

The Si535a, DS3231, and OLED boards communicate with the Nano microcontroller over an I2C bus using only three wires. The boards were chosen because they are highly capable, inexpensive, well documented, and available over the internet from a wide variety of vendors. The completed unit draws little power and can be powered by the Nano microcontroller through the USB cable connected to a computer, a USB charger, or by some 5V USB battery backup chargers.

Unlike many other Si5251A based projects, this project does not require calibration. Both time and frequency accuracy are derived from the GPS receiver..

Construction

Construction of the unit is not critical provided adequate RF techniques are used. Do not use long unshielded wires for RF and 1pps connections. I discovered that The GPS reception will be adversely affected if the GPS receiver and/or GPS antenna is mounted too close to the Si5351. I first built and tested the system using a solderless breadboard without any problems. The circuit was then transferred to a piece of perfboard and placed in a plastic project box. Mounting the unit in a metal cabinet with an external GPS antenna is the preferred way to house this project. To improve short and long term accuracy the unit should be kept at a constant temperature away from any drafts.

The Si5351A, GPS receiver, and the display modules are powered from the 5 volt pin of the Arduino. You can use the USB programming cable, a separate 5VDC source connected to the “+5V” pin, or 7-12VDC source connected to the Arduino Nano’s “VIN” pin to power the unit. Builders may require an external 5 V power supply if a different GPS receiver is used. I used the popular and inexpensive NEO-6M GPS module selling for about $10 USD on many internet sites. Ensure any substituted GPS receiver outputs the "$GPRMC" sentence at 9600 baud.

The builder may choose to use either the OLED display or the LED’s to indicate system operation (or both). Using the OLED display allows the operator to monitor the transmit frequency.

Software Installation and Setup

NOTE: The sketch contains two variables which hold unique data symbols pertaining to your callsign, gridsqare, and power level. The variables are named “WSPRsymbols” and “WSPRsymbols”. Examples containing data for “WZ9ZZZ AA99 37dBm” are shown below.

```
const int  WSPRsymbols[162] = {
  3,3,0,0,0,2,0,2,3,2,2,0,3,3,1,2,2,0,1,2,0,1,2,1,1,1,1,2,2,2,0,0,
  0,2,1,2,0,1,0,1,2,0,0,2,0,2,3,2,1,3,0,0,3,1,2,1,0,2,2,1,3,2,1,2,
  2,2,0,1,3,2,1,2,1,2,1,0,3,0,2,1,0,2,3,0,3,3,0,0,0,1,3,2,3,0,1,2,
  0,0,1,2,2,0,0,2,1,2,2,1,2,0,3,1,1,0,3,3,0,0,1,1,2,1,0,0,0,3,3,3,
  2,2,2,0,2,1,2,1,0,2,1,1,0,0,0,2,0,2,2,3,1,2,1,2,1,1,2,0,0,3,3,2,
  0,0,
};

// Load FT8 channel symbols for "CQ WZ9ZZZ AA99"
const byte FT8symbols[79] = {
  3, 1, 4, 0, 6, 5, 2, 0, 0, 0,
  0, 0, 0, 0, 0, 1, 1, 4, 6, 4,
  7, 3, 1, 1, 3, 7, 0, 0, 0, 2,
  0, 4, 3, 2, 1, 7, 3, 1, 4, 0,
  6, 5, 2, 0, 5, 7, 5, 1, 4, 0,
  7, 1, 7, 0, 0, 7, 4, 3, 5, 7,
  3, 0, 5, 4, 7, 4, 1, 7, 2, 0,
  0, 7, 3, 1, 4, 0, 6, 5, 2
};
```

The file to generate the data for “WSPRsymbols” is called WSPRMSG.exe and is found at <https://github.com/W3PM>. Download this file into a convenient directory. Open the file and enter your callsign, grid locator, and power (dBm) as prompted. "WSPRsymbols" will create a file named "WSPRMSG.txt" in the same directory that "WSPRsymbols" resides. Cut and paste the symbol message data to replace the variable’s data.

The file to generate the data for “FT8symbols” is part of the WSJT suite of programs. Open Windows Command Prompt. Navigate to directory C:\WSJTX\bin. Enter the following command using your own callsign and grid square enclosed in quotes as follows:

```
C:\WSJTX\bin> ft8code "CQ WZ9ZZZ  AA99"  > FT8_message.txt
```

A file named “FT8_message.txt” will be created in C:\WSJTX\bin. Open the FT8_message.txt file into a text editor (use “Channel symbols (79 tones)” data). Separate the "Channel symbols:" data with commas. Cut and paste the symbol message data to replace the variable’s data.

The modified sketch may now be downloaded into the Arduino Nano. The Arduino download website <http://arduino.cc/en/Main/Software> outlines installation instructions for the first-time Arduino user.

The sketch requires the open source library; “SSD1306Ascii” by Bill Greiman. The library is located in the Arduino IDE at; Sketch > Include Library > Manage Libraries. (Windows users will find the menu bar on top of the active window. Linux and MAC users will find the menu bar at the top of the primary display screen.) I found that other more robust ASCII/Graphics libraries were not compatible with the functions and timing complexity of this sketch.

In the unlikely event of operational problems, I suggest the builder check the Arduino Nano and OLED display boards individually to ensure proper operation. The Arduino board can be checked using some of the example sketches provided with the open source Arduino software. The simple “blink” example sketch will confirm that a sketch can be loaded and the Arduino board is functioning. The OLED display may be checked by running one of the AvrI2C examples found with the SSD1306Ascii library.

Transmit Time, Mode, and Band Scheduling

Transmit times, modes of operation, and frequency are controlled by a 3 column, 1-60 row multi-dimensional array called “schedule[][3]” The format for “schedule[][3]” is {TIME, MODE, BAND},. Any number of rows up to 60 may be used. Ensure the last line in “schedule[][3]” is (-1,-1,-1),

Column 1: TIME in 2 minute increments past the top of the hour for WSPR and 1 minute increments for FT8

Column 2: MODE is 1 for WSPR and 2 for FT8

Column 3: BAND is the row number (beginning with 0) of the “TXdialFreq []” variable containing the desired band of operation.

Band frequencies are stored in the variable “TXdialFreq []”. As the name implies, this is the dial frequency as used in WSJTX and not the actual transmit frequency. The offset frequency (1400 – 1600 Hz in the case for WSPR) is added to the dial frequency in the sketch to determine the actual transmit frequency.

Band frequencies are stored in Hz i.e 7.04 MHz = 7040000. Any number of bands may be stored. Band frequencies are not limited to the 8 meter band. Any frequency below 40 MHz may be entered.

Example:

```
 const int schedule [][3] = {
 {0, 1, 0},   // WSPR at top of the hour on 40620.000 kHz
 {2, 2, 1},   // FT8 at 2 minutes past the hour on 40680.000 kHz
 {3, 2, 1},   // FT8 at 3 minutes past the hour on 40680.000 kHz
 {20, 1, 1},  // WSPR at 20 minutes past the hour on 40680.000 kHz
 {58, 1, 1}, // WSPR at 58 minutes past the hour on 40680.000 kHz
(-1,-1,-1),  // End of schedule flag
 };

const unsigned long TXdialFreq [] =
{
 40620000,  // band 0
 40680000,  // band 1
 0
};
```

Operation

When the unit is turned on, the auto-calibration algorithm begins to calculate the correction factor for the Si5351A's 25 MHz clock. This takes 40 seconds to complete.

When the GPS receiver obtains valid data, the OLED screen will display the operational parameters. The transmitter arming (ON/OFF) status is displayed in the upper right hand corner. Depress the pushbutton to toggle the transmitter arming status on and off when not transmitting. Allow a few minutes for the MEPT's SI5351 25MHz oscillator and autocalibrate function to stabilize before setting the transmitter to "ON".

Although the auto-calibration function will set the Si5351 to the correct frequency, some inexpensive Si5351 boards will experience some short term drift between calibration updates. The short term drift results from the 25 MHz crystal being mounted too close to the Si5351 IC. The crystal will heat and cool between transmit on/off cycles. This is especially evident on higher frequencies if longer WSPR transmit intervals are used. The 8 meter transmit schedule provided in the sketch leaves little time for the 25 MHz crystal to cool between transmit cycles resulting in very little short term drift.

The sketch provides the option to hop frequency within the 200 Hz WSPR window or 2300 Hz FT8 window with each transmission. Transmit frequency hopping may be beneficial to avoid decoding conflicts with interference or other band users. If frequency hopping is not used the transmit frequency will be the dial frequency plus the selected offset frequency.

The following commands are found near the beginning of the sketch. These may be changed to reflect desired operation prior to compiling and uploading the sketch:

```
// Enter desired WSPR transmit offset from dial frequency below:
// Transmit offset frequency in Hz. Range = 1400-1600 Hz (used to determine TX frequency within WSPR window)
unsigned long TXoffset = 1522UL;

// Set to true if you want the WSPR frequency to change within the 200 Hz WSPR freuency window
bool FreqHopTX = false;

// Enter desired FT8 transmit offset from dial frequency below:
// Transmit offset frequency in Hz. Range = 200-2500 Hz (used to determine FT8 TX frequency within FT8 window)
unsigned long FT8_TXoffset = 1230UL;

// Set to true if you want the FT8 frequency to change within the 2500 Hz FT8 freuency window
bool FT8_FreqHopTX = false;
```

.