Auto-Calibrated Si5351A Quadrature VFO Project

Gene Marcus W3PM GM4YRE

This is a VFO using the very popular Si5351A clock generator board to generate 80 – 6 meter I/Q signals at outputs CLK 1 and CLK 2. Frequency and time accuracy are maintained by a highly accurate temperature compensated DS3231 real time clock (RTC) board.

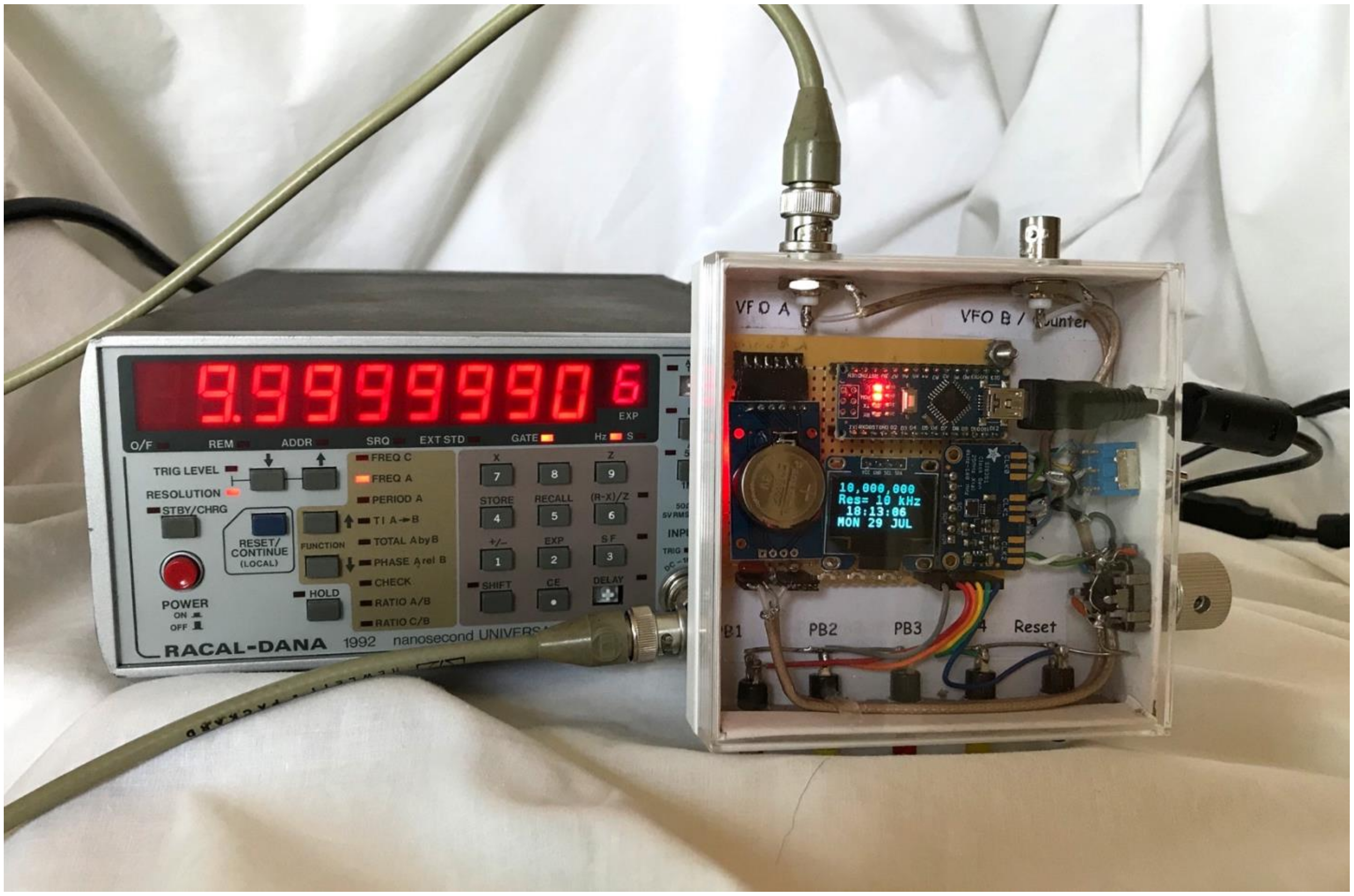

Figure 1

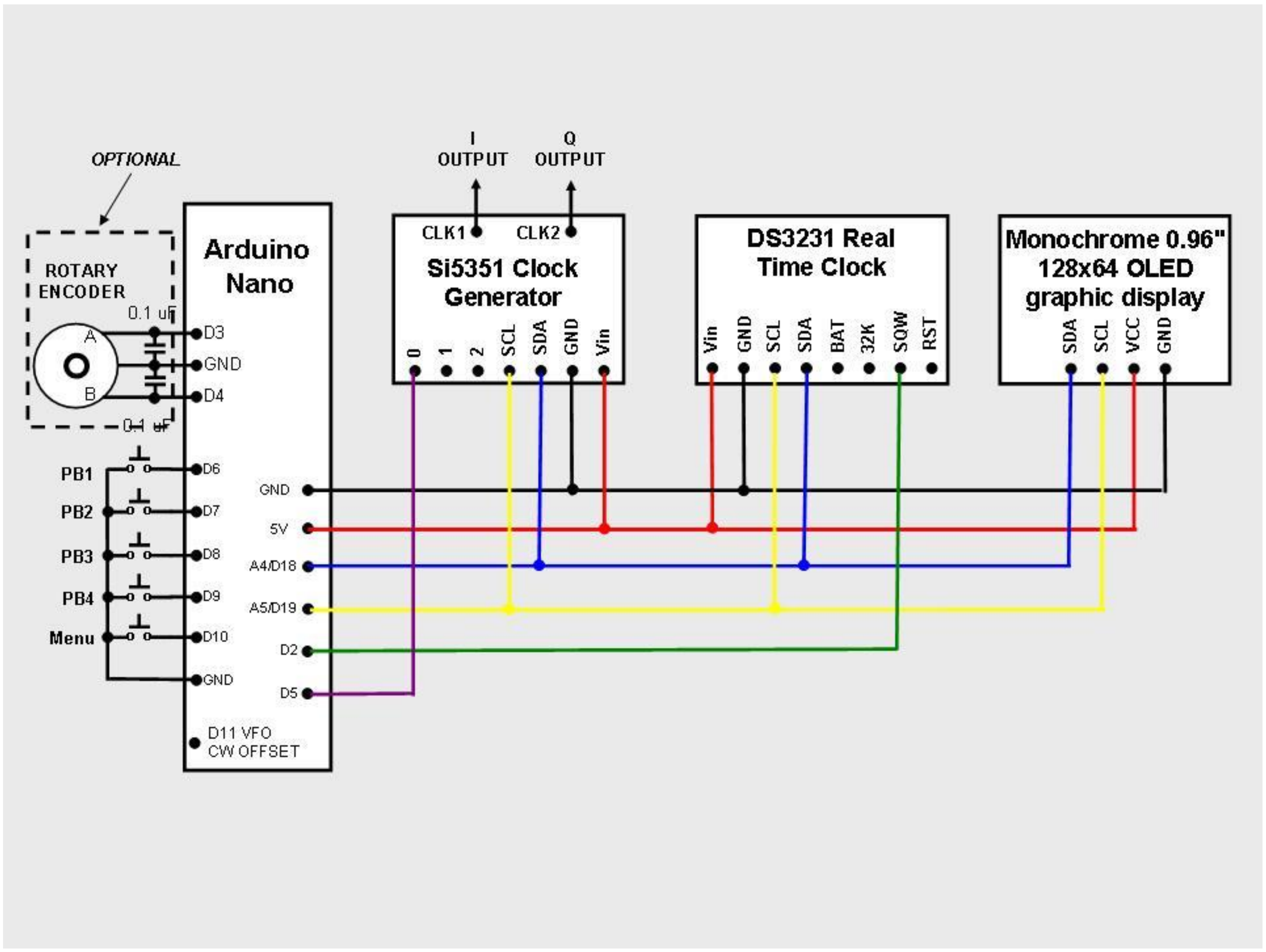

Figure 2

Introduction

Figures 1 and 2 may look familiar to builders of the Multifunction Project. This is because the I/Q VFO project shares the same hardware. If you built the Mutifunction Project, simply upload the Arduino sketch or vice versa.

I developed the I/Q VFO to replace an analog I/Q VFO in a binaural receiver that I built many years ago. The quadrature outputs may also be used for SDR and phasing rig projects.

As shown in Figure 2 the building blocks for this project consist of four boards: an Arduino Nano (*or Uno*) microcontroller, a Si5351A clock generator breakout board, a DS3231 real time clock (RTC) board, and a 0.96 inch 128x64 serial I2C OLED display board.

The 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 VFO projects, this project does not require calibration. Both time and frequency accuracy are derived from a DS3231 real time clock (RTC) which has an uncertainty of about 2 parts per million from 0 – 40 degrees C (32 to 104 degrees F). Additional accuracy can be attained by adjusting the DS3231 aging offset.

Construction

NOTE: Frequency and timing accuracy is provided by the DS3231SN RTC. Be sure your project uses the DS3231SN, some vendors substitute the DS3231M version which is not as accurate as the "N" version.

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 first built 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 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. Figure 1 shows the frequency accuracy of my completed unit when set to 10 MHz .

The Si5351A, DS3231 RTC, 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.

Software Installation and Setup

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, OLED display, and RTC 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. The DS3231SN RTC board may be checked by running a time initialization sketch. A search of the internet will provide a number of methods to initialize the board. Adafruit Industries provides a very good tutorial to confirm the board is operational. Upon initialization, the time will be a few seconds slow. This is because the time set is the time the sketch was compiled, not the current time. The time may be updated using the "Clock Set" push buttons after the project is completed. Provided a battery is used with the DS3231, the date will not require setting for a year or more.

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. The internal DS3231 temperature compensation algorithm concurrently calculates its correction factor every 64 seconds. The correction factors are continuously updated, therefore you may see small frequency jumps for the first few minutes until the unit stabilizes.

The function selection menu is first displayed beginning with the VFO function. Depress pushbutton 3 to scroll through the other functions. Depress pushbutton 2 to select the desired function.

Once selected, all functions are controlled by pushbuttons. An outline of pushbutton operation follows:

	VFO	**Clock**	**Set Clock**	**Set Date**
PB1	Decrease Frequency	N/A	Time sync* / Set Hour	Set Day
PB2	Increase Frequency	N/A	Set Minute	Set Month
PB3	Band Select	N/A	N/A	Set Year
PB4	Resolution Select	N/A	Hold to change time	Hold to change date

Depress the **MENU** pushbutton to exit and return to the original function selection.

The time is set by selecting the "CLOCK SET" function. Hold down pushbutton 4 while depressing pushbutton 1 to set hours or pushbutton 2 to set minutes. Release pushbutton 4 at the top of the minute.

* A shortcut to synchronize the time is available provided the displayed time is correct to within +/- 30 seconds. Simply depress pushbutton 1 at the top of the minute while in the "CLOCK SET" function to synchronize the time.

The CW offset variable (line 87 "CWoffset") controls the frequency offset. When the Arduino pin D11 is held low, the programmed offset in hertz is added or subtracted. Any positive or negative value of hertz can be used to program the frequency offset.

Options

The unit may be built without the RTC module and used as a stand-alone VFO. Set the "AutoCalibrate" variable on line 78 to "false". If "AutoCalibrate" is set to "false" the stand-alone calibration factor should be used. Instructions to determine the calibration factor begin on line 90 of the sketch. If used in the stand-alone mode, you can expect a drift rate of approximately 1.6 Hz / °F (2.8 Hz / °C). Operating the VFO in stand-alone mode also provides an excellent means to test the project prior to connecting the RTC module.

The frequency is controlled either by the rotary encoder or the frequency up/down pushbuttons. You may want to include either the encoder or pushbuttons, or both.

DS3231 Aging Offset Adjustment

While the default RTC is already very accurate, its accuracy can be pushed even higher by adjusting its aging offset register. The documentation located at the end of the sketch provides methods to determine the correction factor and upload it to the aging offset register.

For normal operation this adjustment is not required. The default 2 parts per million accuracy of the RTC will result in an uncertainty of less than +/- 30 Hz on 20 meters.

Gene Marcus W3PM GM4YRE
113 Wickerberry Lane
Madison, AL 35756
w3pm(at)arrl(dot)net