BARSicle - the learn, code, build project

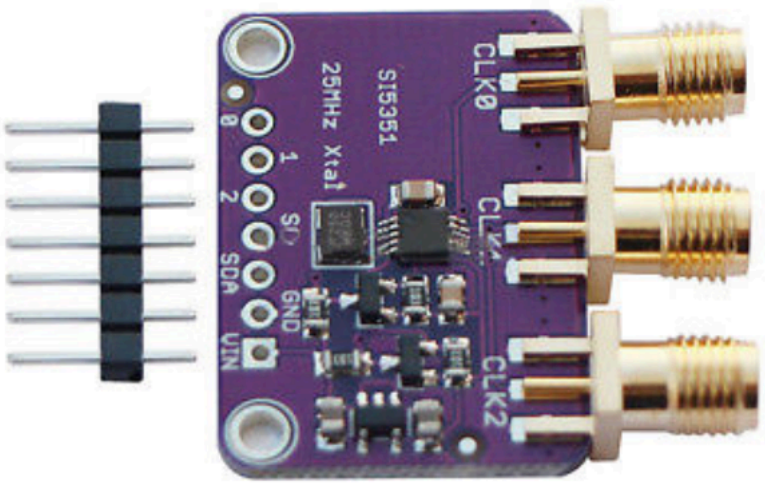
The BARScile project is a beginners learning, coding and building project of the Banbury Amateur Radio Society. It will cover,

* the use of the Arduino Nano
* the build of a digital Signal Generator / VFO, an RF Volt/power meter, a Direct Conversion Receiver.

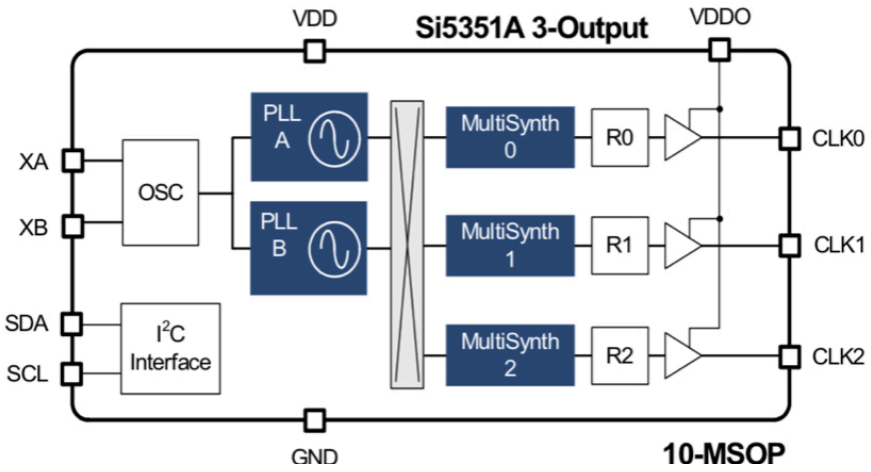
An extension of the project will cover an SSB Exciter, Power Amplifier with Low Pass Filter and an SWR/Antenna Tuning Unit

3. ARDUINO
DDS

This is the final part of the build for ur Signal Generator. It is base on an Si5351 Digital Frequency Synthesiser.. We will use it generate 3 individual outputs of +10dBm from 100kHz to 150MHz to an accuracy of a few centi-hertz. It is programmed by an I2C bus. We will use an excellent Arduino library called Ethernet_Si5351.
\

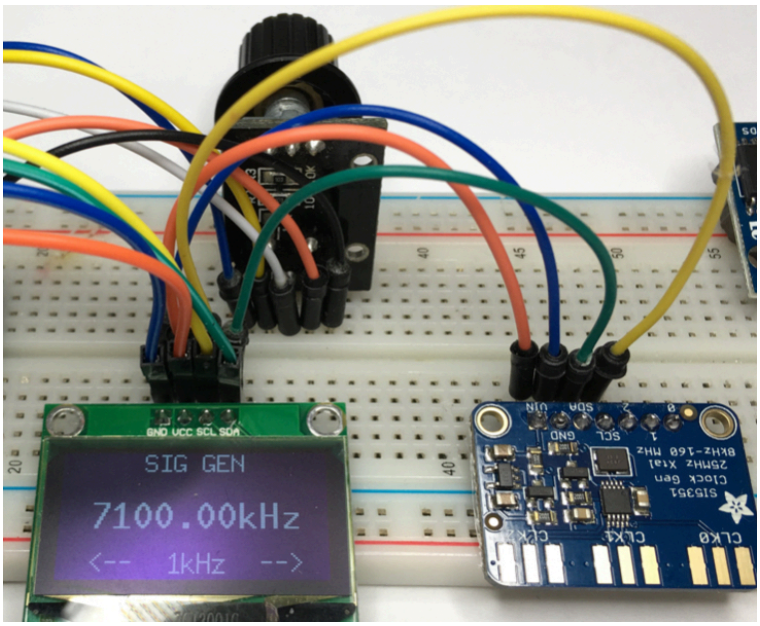


How does it work?



The module uses a 25MHz xtal which is used by two PLLs to look them to high frequency. This is then divided down by a "Multi Synthesiser", the output can be taken directly or further divided to give an output as low as 8kHz. The chip is a 3.3v part, but the module has 5V <-> 3.3V convertors for power and the I2C bus. When set to an output current of 8mA, it will deliver +10dBm into a 50r load. But take note, the outputs are square waves, and so have considerable harmonic content.

Connections
Connecting the module is very simple, it uses the 5V supply and the I2C SCL & SDA signals, the same as the OLED display.



The module has three outputs CLK0, 1 & 2.

Ethernet_Si5351 library
This library allows easy and complete control of the synthesiser. The main fucnitons we will use are

```
#include "si5351.h"

Si5351 dds;

void setup() {

  dds.init(SI5351_CRYSTAL_LOAD_8PF, 0, CORRECTION);

  dds.drive_strength(SI5351_CLK0, SI5351_DRIVE_2MA);

  dds.output_enable(SI5351_CLK0, 1);

  dds.set_freq(frequency, SI5351_CLK0);
}
```

The si5351 library is included at the top of our sketch. And a `"DDS"` object is created. Then in "setup( )" the DDS is initialised, with the correct setting for our xtal (8pF load), it is set for a 25MHz xtal (`"0"`) and any correction is applied to get the correct output frequencies.

The drive output strength is set to the maximum of 8mA, which will give us +10dBm int a 50R load. When 3 outputs are used they must be set individually. Next the outputs used are enabled, the inhibit/enable is very fast and can be used for digital signals.

Finally a frequency can be set fo each of the outputs individually. This is programmed in centi-Hertz using a 64bit variable to handle the numbers needed.

Testing
A simple sketch is `File > Sketchbook > My_FREQ` which uses the code above to generate a 7.1MHz signal. Upload this and listen on a nearby receiver to check it is working. You can change the frequency by changing the variable `myFreq.` There is nothing in the `loop( ).`

Signal Generator
Open `File > Sketchbook > My_SIGGEN.` In the loop( ) the button of the encoder is checked to see if it is pressed. This is the button function which returns `true` or `false`.

```
bool button() {                                          // check button
  if (digitalRead(SW) == LOW) {                          // debouce
    while (!digitalRead(SW));
    return true;                                         // pressed
  }
  else {
    return false;                                        // not pressed
  }
}
```

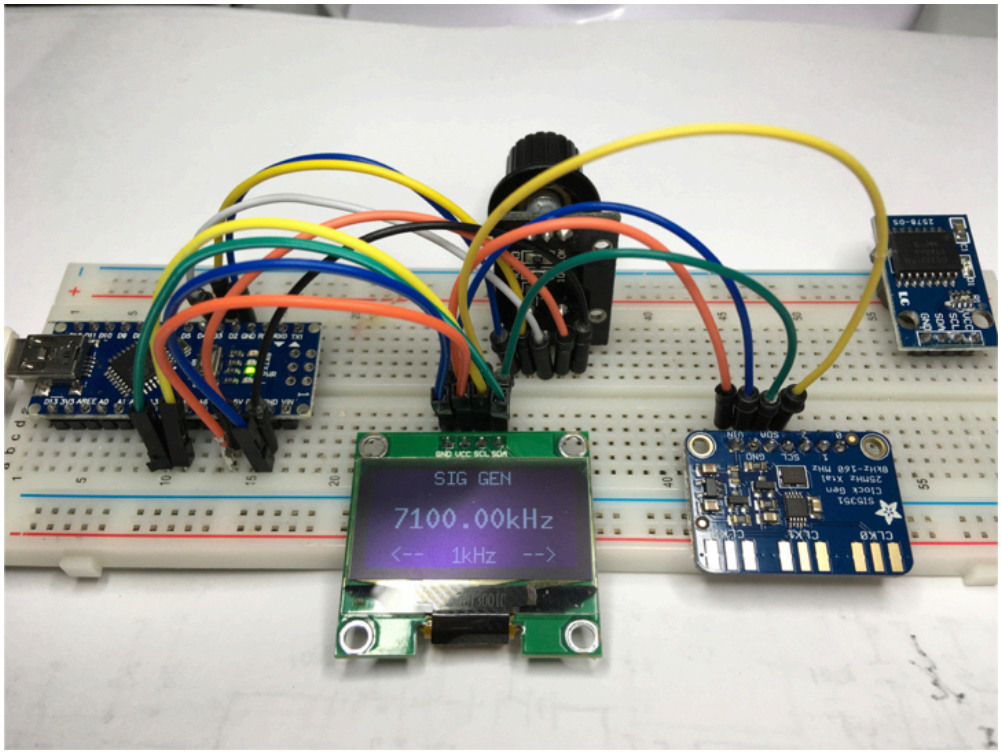
this function is called during the loop( )

```
  if (button()) {                                        // button pressed?
    if (freqStep == 100000000) freqStep = 1000;          // update step
    else freqStep = freqStep * 10;
    dispUpdate();                                        // display
  }
```

The button allows the frequency step to be cycled from 10Hz to 1MHz.

Next the encoder is read to control the tuning. The code is the same as we saw in section 4. Arduino. And need not be explained again here.

This is the completed Signal Generator Breadboard



Calibration.
If you open `File > Examples > Ethernet_Si5351 > si5351_calibration` there is a calibration sketch provided with the library.

The instructions are in the sketch. When you have run this and the module is outputting the correct frequency. Set the value of `CORRECTION` in your code. You should be able to calibrate to better than 1Hz.