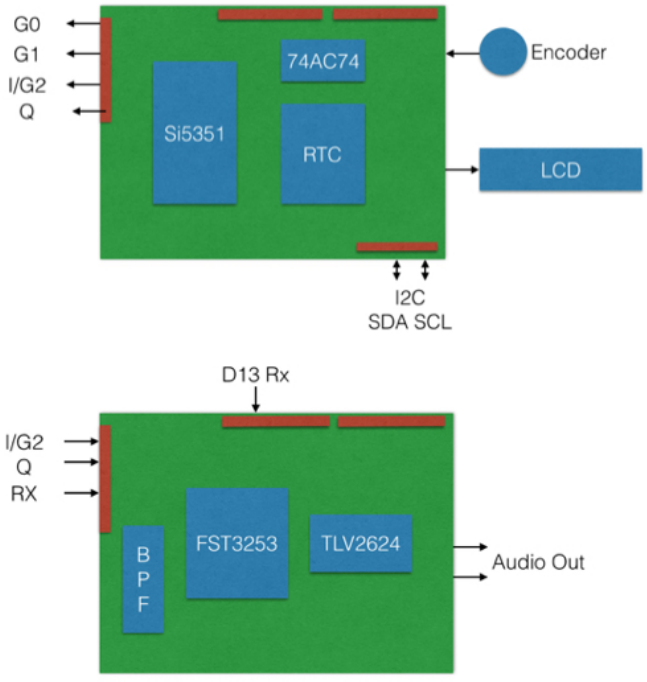
Concept Session 2 - VFO prototype
Course Notes

It is time to remind ourselves what Concept is all about.

	WSPR	QRSS	CW	SDRX	SDTX	DC RX
UNO	x	x	x	x	x	x
*RTC	x			[x]		
*VFO	x	x	x	x	x	x
*SDRX				x		
SDTX					X	
DC RX						X
PA	X	X	X		X	
LPF	x	x	x	x	x	x

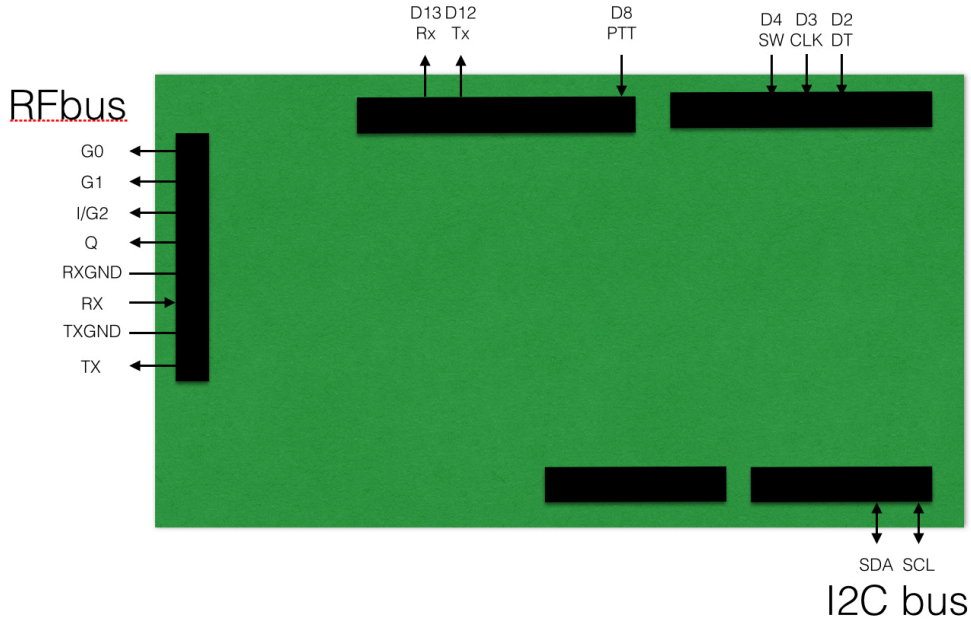
The whole Concept program is to build a serial of shields that plug in to the Arduino UNO and which provide the hardware for Amateurs to implement various modes of operating using programmable digital VFO and SDR techniques, for example SSB, CW, WSPR, QRSS...

In the first course we will design and build two shields which will create an SDR receiver



The digital VFO_RTC_IQ and the Software Defined Radio SDR_40M (40m SDR).

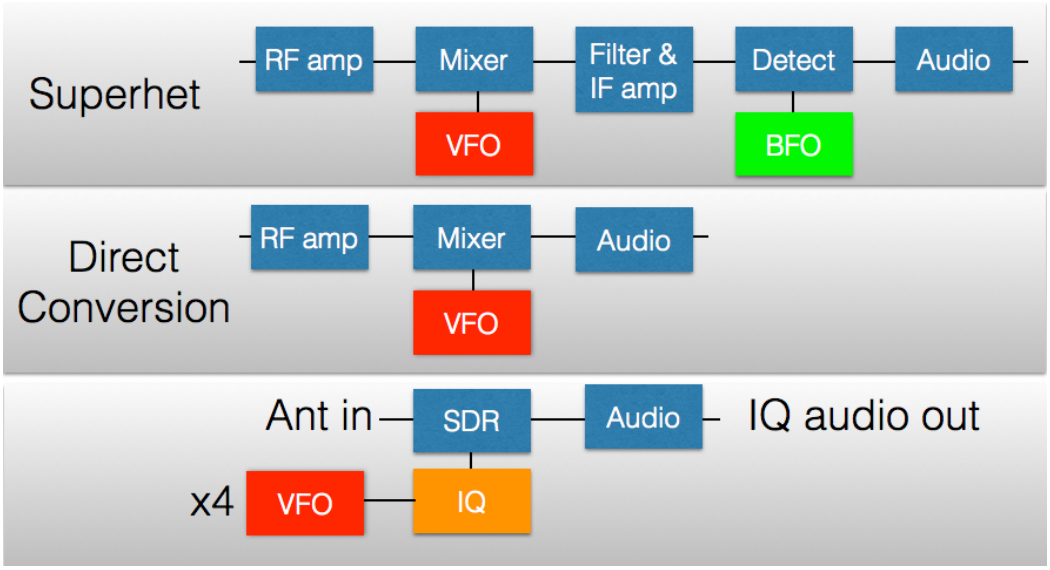
You should note that the interfacing between shields has been defined so that they are compatible:

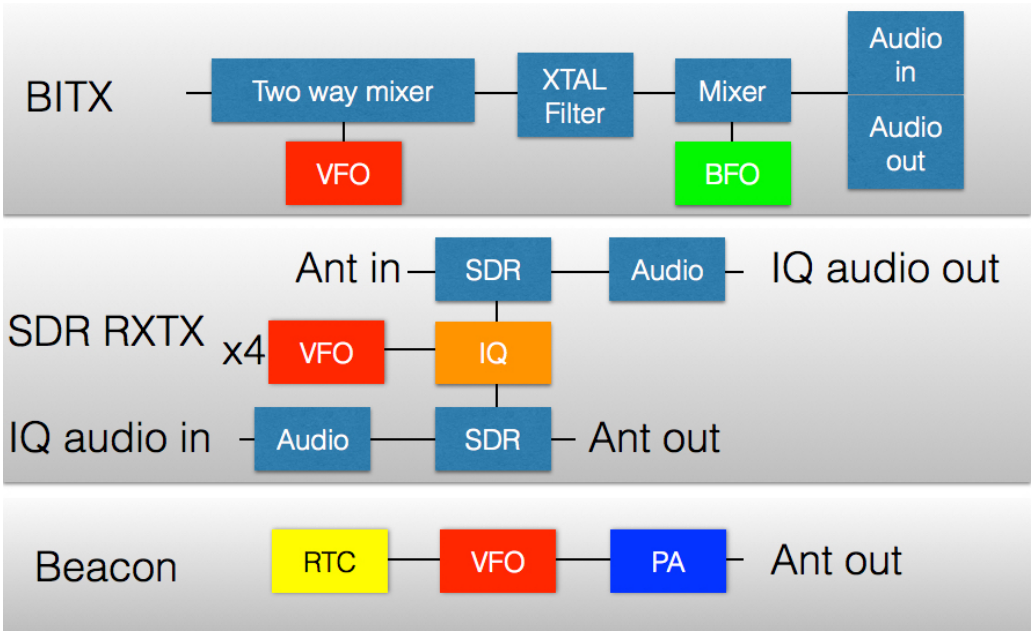


A new connector has been added to the basic board on the left side, to carry RF signals. Some of the digital pins at the top have pre-defined uses, and the analog pins A4/5 are used for I2C bus communication.

Needs of a VFO

Various receiver and transceiver designs have different needs from a VFO depending on the architecture of the circuit.





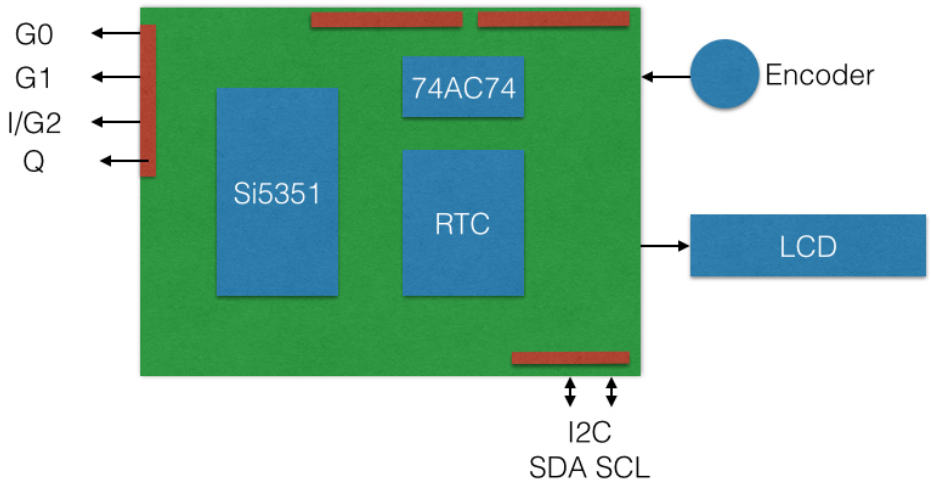
A VFO design to meet all these needs must have

1. Several outputs, each programmable to different frequencies

2. Generation of Quadrature outputs for SDR techniques

3. A real time clock to provide time signals for beacons

So our VFO shield block diagram is like this



The shield contains a Si5351 digital synthesiser chip with three outputs, two are brought out externally, G0 & G1, the third G2 feeds a SN74AC74 used as a Johnson Counter to generate the quadrature I/Q outputs I/G2 & Q.

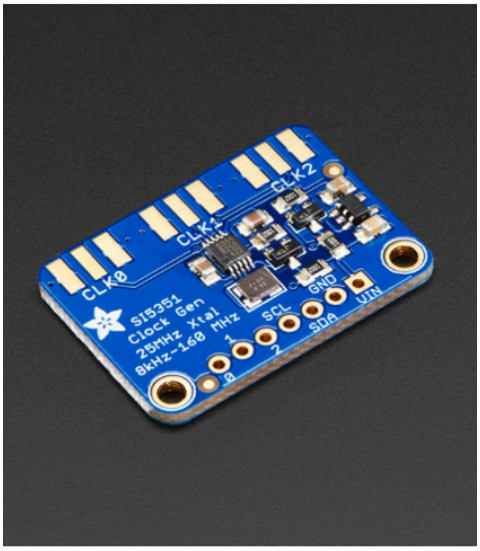
The board also carries a real time DS3231 module with a backup battery. The A4/5 signals control both VFO and RTC chips on the shield, and the external LCD display.

A rotary encoder connects through to the Arduino D2/3/4 pins for tuning.

Build a prototype

We will build, step-by-step this VFO as a prototype on our experiment board. To do this you need Kits 1 & 2 (see Session 1 notes which describes the kits)

The Si5351



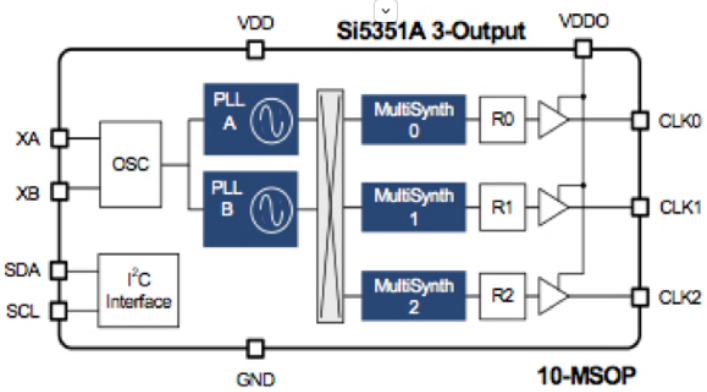
The Si5351 is mounted on a module. It includes level shifter from 5V to 3V3 for the I2C bus and a 3V3 regulator, allowing the module to interface directly with the 5V signals of the Arduino UNO.

Amateurs have in the last few years experimented with a variety of digital synthesisers:

- the Si570 - widely used in the famous Softrock SDRs, single output

- the AD9850 - used by many as it became available as a low cost module from Chinese suppliers, single output.

- the Si5351 - now used in many designs as it has an excellent supporting library for the Arduino. And provides three outputs.

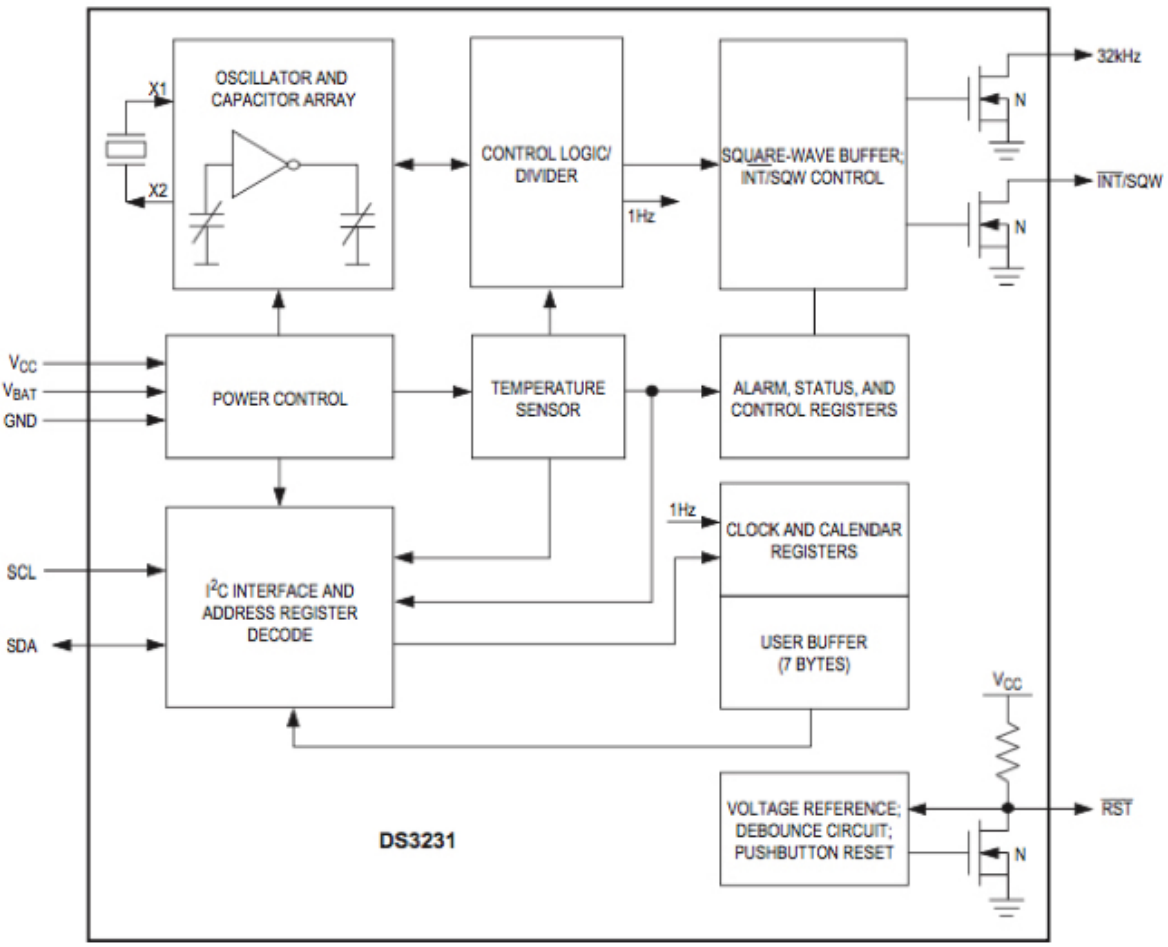


The Si5351 chip has an oscillator at 25MHz using an external Xtal. This is used to drive two programmable PLLs which run at 600-900MHz. The outputs of the PLLs go to three

programmable Multi-Synth dividers to derive the output required. The chip is controlled by instructions over the I2C bus.

For more information about the Si5351 look at the help files copied from the USB stick.

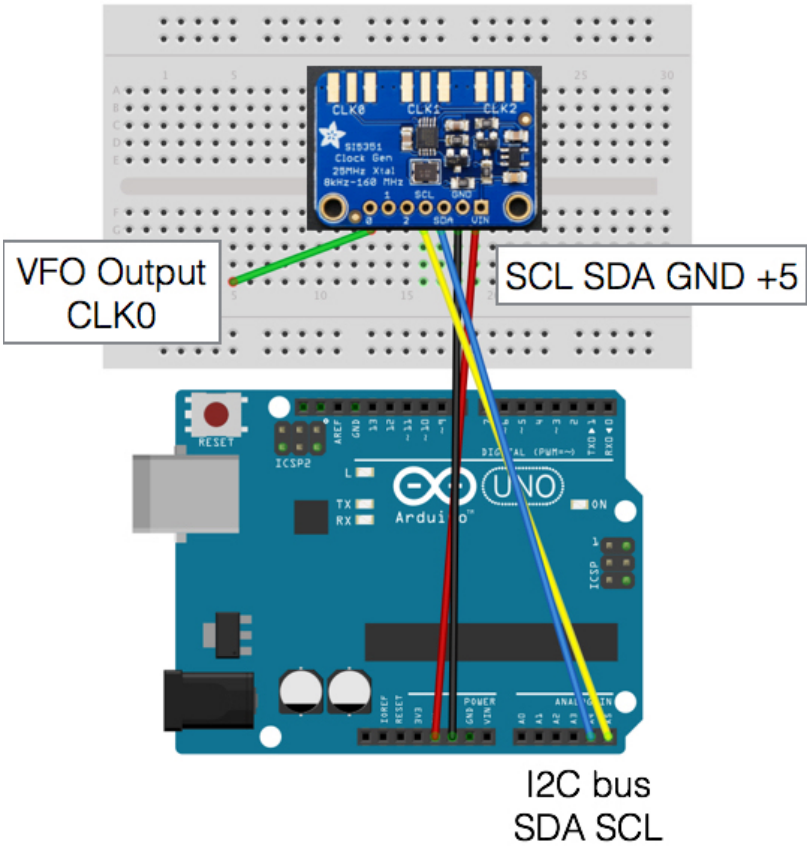
The DS3231 RTC



The integrated DS3231 includes a temperature controlled Xtal oscillator and all the divider chains , It is a very accurate clock device, with readout of years, months, day, day-of-the-week, hour, minutes and seconds. Like the Si5351 it is controlled and the time is read over the I2C bus.

Build a VFO

Let's start with the Si5351 module. Wire it up like this



Take care to get the connections correct as the wire cross each other. You can add a short wire to the output CLK0 to act as an aerial, as this is the output used in the sketch.

Now open and upload the first VFO sketch

```
File > Sketchbook > My_VFO_KB
```

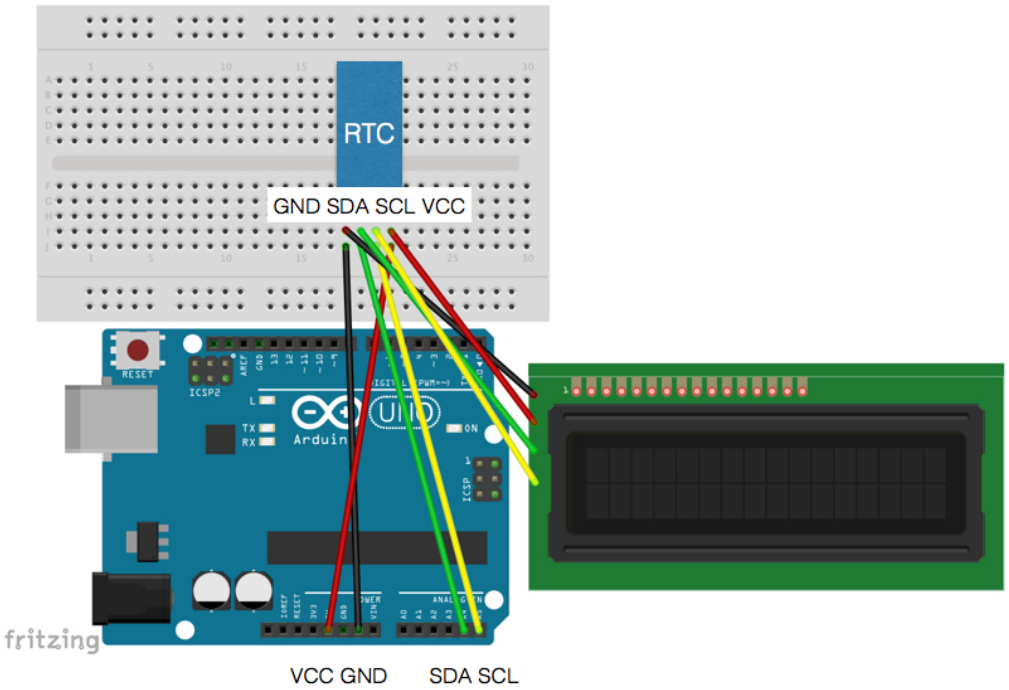
Open the monitor (top right lens button in the Arduino IDE software window), make sure that the bottom drop-downs say "Newline" and "9600".

The VFO will output its frequency "My_VFO = 7000000Hz". Type in any new frequency (from around 100kHz to 30KMHz) and it will tune to this. Check you are getting the correct output by listening on a nearby receiver.

The sketch code is not reproduced her as you can see and read it in the Arduino IDE. The slides provide a quick over view of the code.

Build an RTC

Next we will build an RTC. Wire up this on your experiment board using the plug-in RTC module.



(These diagrams were made using a free software called “Fritzing”).

Take care to get the connections correct as the wires cross. Both the RTC and the LCD connect to the I2C bus on A4/5. (A4 = SDA and A5 = SCL signals).

When the DS3231 is first wired up the time may/will not be correct. Open and upload the sketch

`File > Sketchbook > My_RTC_Set`

The LCD will display the format to enter the time information.

Open the monitor and enter the time in this format

YY = 15, MM = 00 Jan, DD = 1-31, w = 0 Sun, HH = 1-23hr, MM = 0-59, SS = 0-59,

for example:

1506212115500 is “Tues 21 Jul 2015 11:55:00”

Enter a time ahead of the actual, then wait until the exact second and hit return or click “send” to set the RTC. It will now be at that time. For use as a WSPR beacon you MUST align the time (0-1 sec) with that of your PC, which itself must be correct. We could have used a GPS module, but this seemed a bit difficult (you could develop one..)

To simply read the time from your RTC run the sketch

```
File > Sketchbook > My_RTC_LCD
```

Check out the code of both sketches until you understand it. The course slides provide a quick over view of the code.

You have completed session 2.

Next Session

We take a break from Concept and look at the popular Eagle CAD software for schematic capture and PCB design.