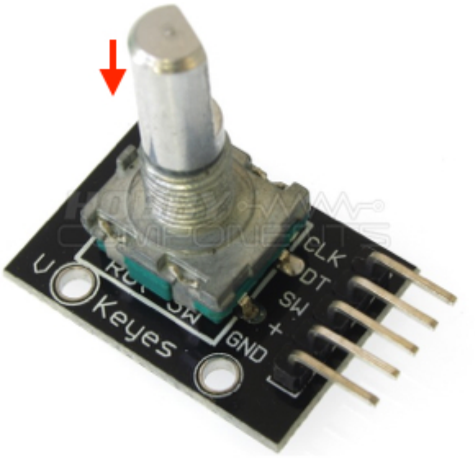
Concept Session 4 - VFO ENC & LCD
Course Notes

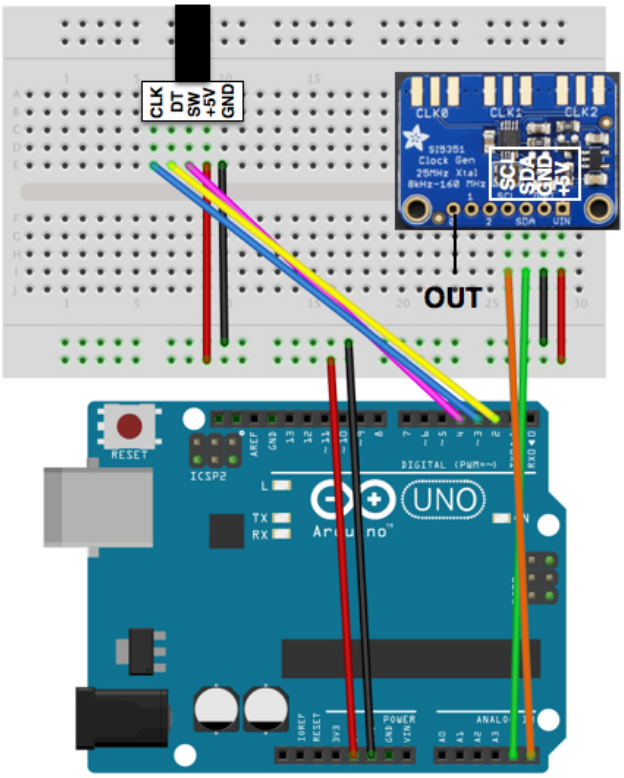
In this session we will add the Rotary Encoder for tuning and an LCD for display to the basic Si5351 VFO

The Rotary Encoder



The encoder generates two outputs which describe the direction of rotation. The output changes in steps. In addition there is a push button which can be use for other actions.

The Encoder connects to the Arduino on pins CLK = 2 & DT = 3, and the button SW = 4.

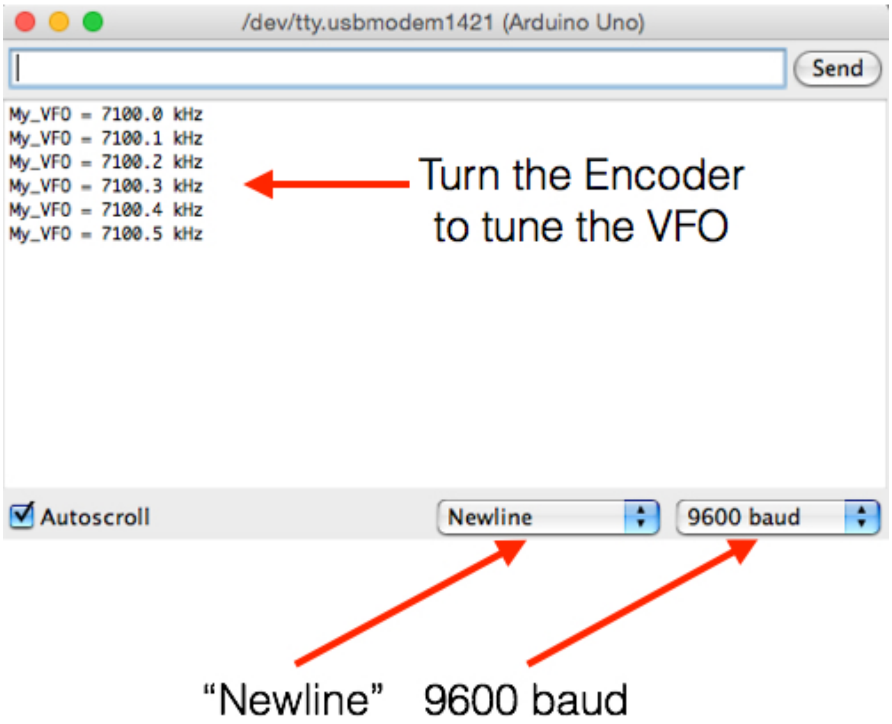


Wire up your Si5351 module and this circuit on your breadboard. Take care to connect 2, 3 and 4 correctly, as the wires cross. The wiring to the Si5351 module is as before.

Now open and upload this sketch

```
File > Sketchbook > My_VFO_ROTARY
```

Then open the monitor window which will be used to display the tuned frequency.



/dev/tty.usbmodem1421 (Arduino Uno)

```
My_VFO = 7100.0 kHz
My_VFO = 7100.1 kHz
My_VFO = 7100.2 kHz
My_VFO = 7100.3 kHz
My_VFO = 7100.4 kHz
My_VFO = 7100.5 kHz
```

☑ Autoscroll   Newline   9600 baud

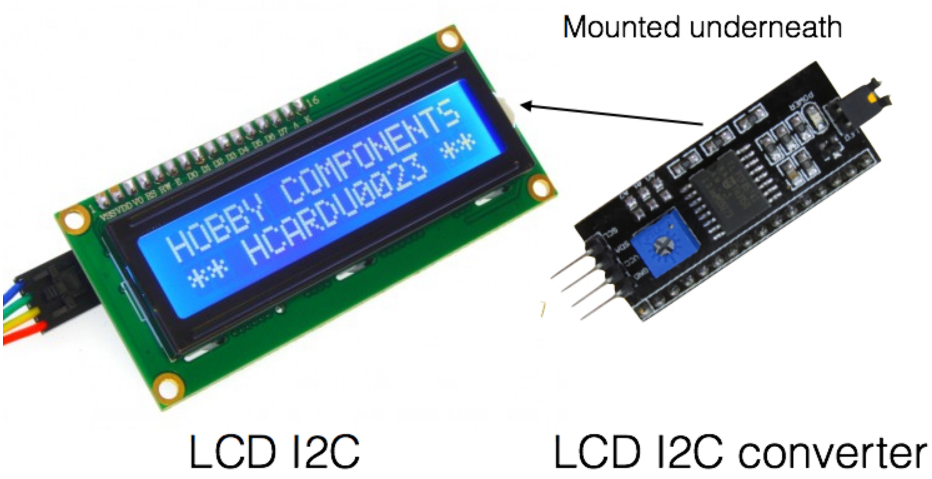
Turn the Encoder to tune the VFO

“Newline”   9600 baud

At the moment the VFO only tunes in 100Hz steps starting from 7100.0kHz. Check it out on a nearby receiver. Many may disagree but I find that 100Hz steps are all that is needed to tune in an SSB station in listenable quality.

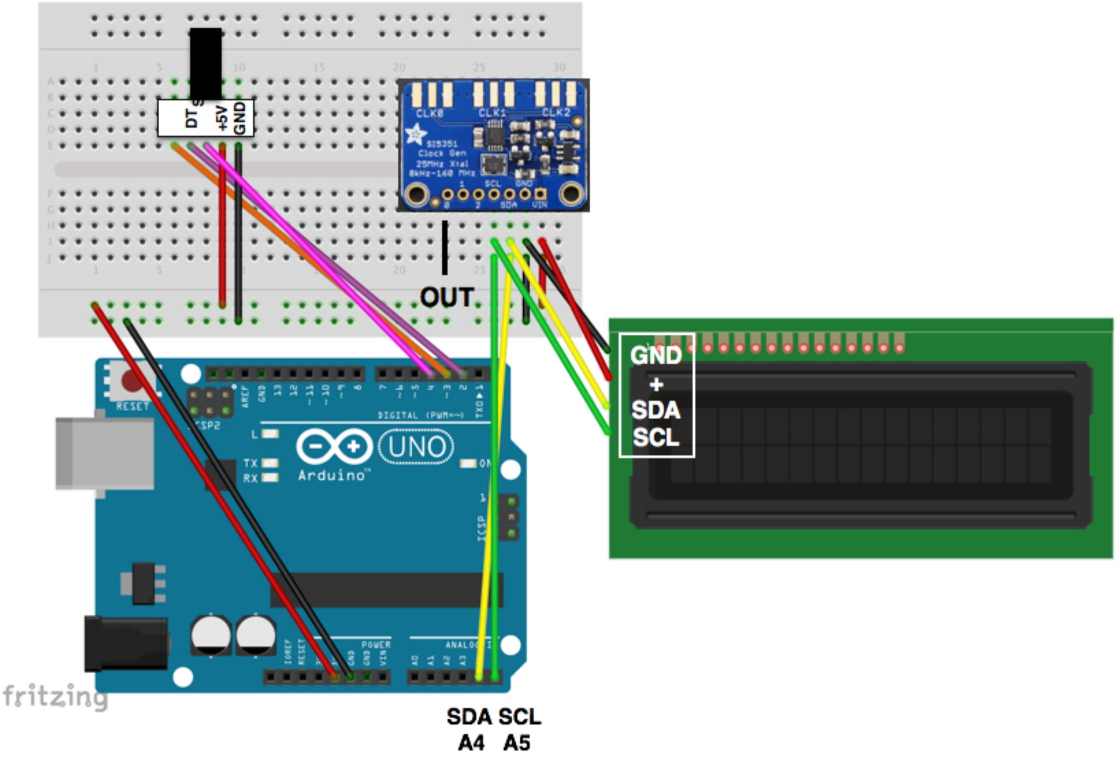
Read and understand the software code of the sketch.

Add an LCD display



The LCD display in Kit 2 is 16 characters on 2 lines. Attached to the back it has an I2C - parallel converter to drive the data and control lines of the display. The I2C converter also has a small potentiometer which adjust the display brightness. There are four connections, I2C (SDA & SCL), VCC & GND.

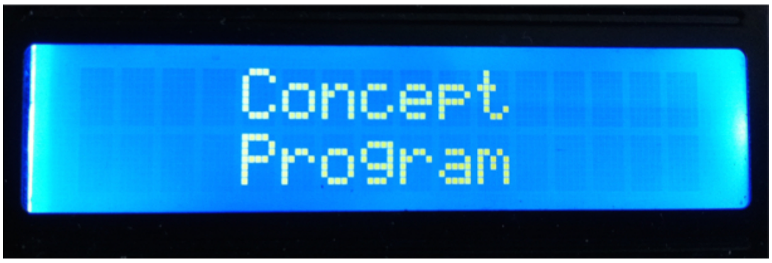
Add this to your breadboard like this:



Again take care to get the wiring correct as some wires do cross.

Test

First thing we will do is to test the display by driving it with some a simple text message.



Perhaps we should look at the code this time as it shows how to use an I2C LCD. This is the first part of the code

```
// include libraries for I2C comms and LCD driver
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// LCD I2C address, cols, rows
#define LCDADDR 0x27
#define LCDCOLS 16
#define LCDROWS 2

// create an LCD object "lcd"
LiquidCrystal_I2C lcd(LCDADDR, LCDCOLS, LCDROWS);
```

The library “Wire.h” implements the I2C protocol on pins A4 & 5. “LiquidCrystal_I2C.h” is used generate characters and to drive the display. The library is on the USB stick copied to your library folder. The “Wire.h” is a standard library which is part of the Arduino IDE installation.

Every device connected to an I2C bus has a unique address, the LCD is at 0x27 or 0x3F depending on the model (if your does not work edit the code to change the address), and the library needs us to define the columns and rows of the display.

Then, as we have seen before, we create an object called “lcd” (or any other name you like!) which then inherits all the class functions in the library. The next part of the code is all in the “setup” which, remember, will run just once.

```
// setup runs once on upload
void setup() {
  // start serial comms to the Arduino IDE monitor window
  Serial.begin(9600);

  // initialise the LCD & switch the backlight on
  lcd.init();
  lcd.backlight();

  // move the cursor to col, row and output text
  lcd.setCursor(3, 0);
  lcd.print(" Concept ");
  // wait 2 sec (2000ms)
  delay(2000);

  // move the cursor to col, row and output text
  lcd.setCursor(3, 1);
  lcd.print(" Program ");
}

// loop does nothing, but must be here
void loop() {

}
```

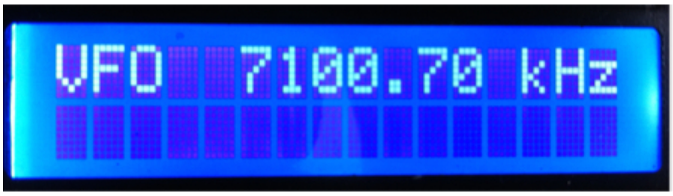
As you can see there are class functions for the “lcd” from the library, for example to set the cursor to a col/row `lcd.setCursor(3, 0);` and to print text `lcd.print(" Concept ");`.

Try the encoder

Now we know the LCD is working try the encoder. Open and upload the sketch

```
File > Sketchbook > My_ROTARY_LCD
```

Again set the LCD address if it is different. This creates another very simple VFO, adding an extra feature, that is to use the button to change the band 40 - 30 - 20m. It only tunes in 100Hz steps.



```
VFO  7100.70 kHz
```

Have a good look at the code, an explanation is on the course slides. Especially study the functions used to control the Si5351, and read the Si5351 help documentation.

Now that you have a complete breadboard of the VFO, but without the RTC, you can try out a number of other sketches that are on the USB stick.

Or maybe you want to write your own?

You have completed session 4.