CONCEPT

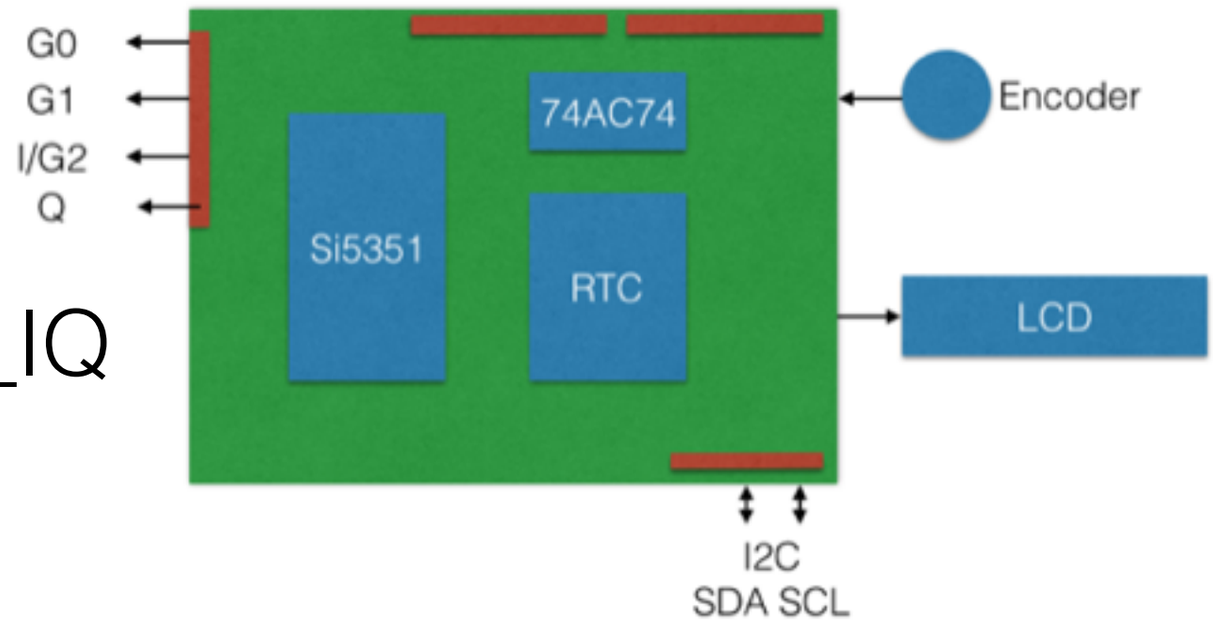
Extend the VFO with tuning & display

Last time

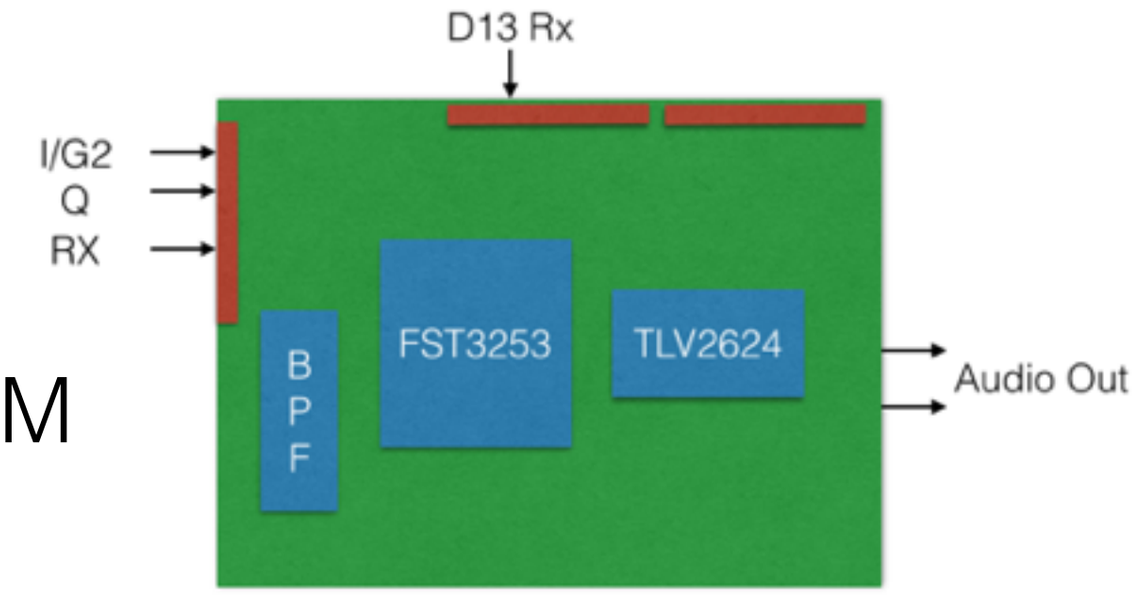
- We learned about the Eagle PCB design software
- We had a quick look at LTspice
- And previously we
  - Looked at Concept
  - Requirements for a VFO
  - Built a VFO and an RTC on the breadboard and tested sketches

BARS Concept

VFO_RTC_IQ

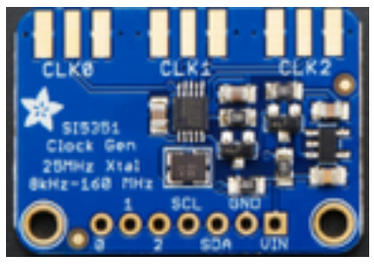


SDR_40M



Kit 1 & 2

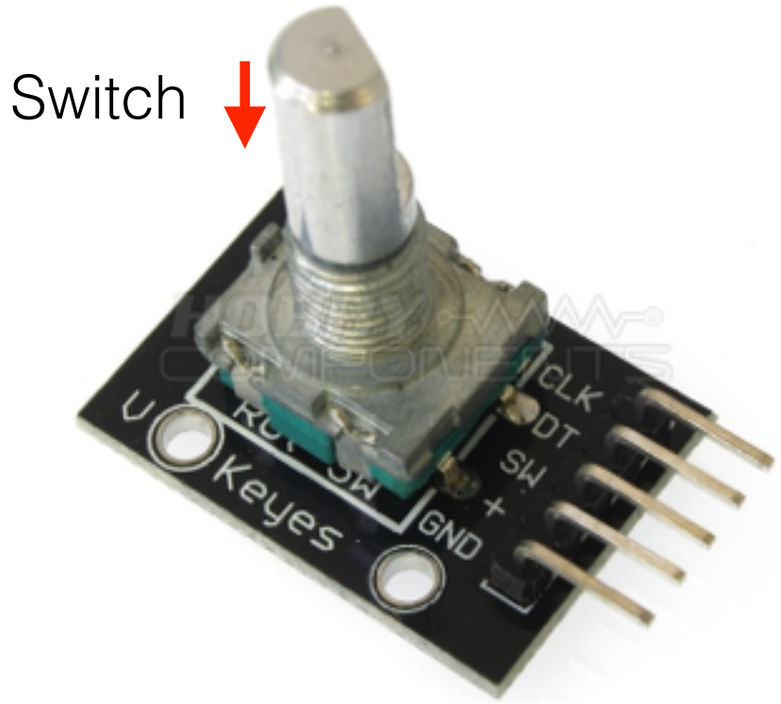
	Starter	Learner	VFO & RTC	VFO+ROT	VFO+ROT+LCD	VFO_RTC_IQ	SDR_40M
Arduino UNO	x						
400 point BB	x						
Jumper wires	x						
LED		x					
220R		x					
Piezo buzzer		x					
Si5351 module			x	o	o		
Rotary Encoder				x	o		
I2C LCD			x		o		
RTC module			x				
CR1220 battery			x				
F - M wires			x		o		

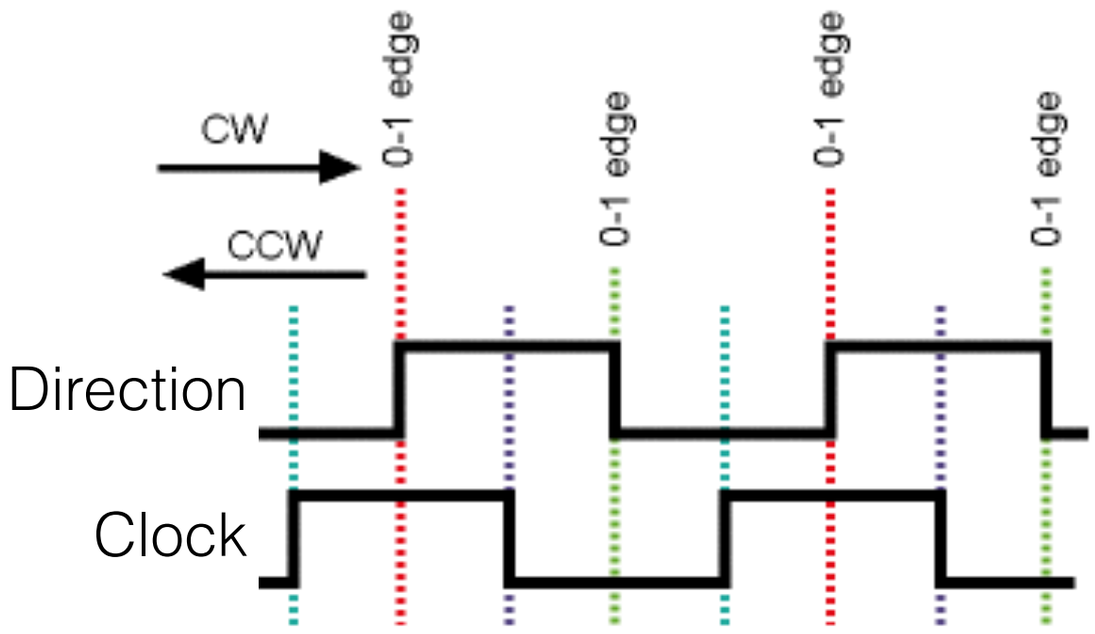


Good News!!! The RTCs have arrived - 7 available

Rotary Encoder tuning

…and push the button for other actions



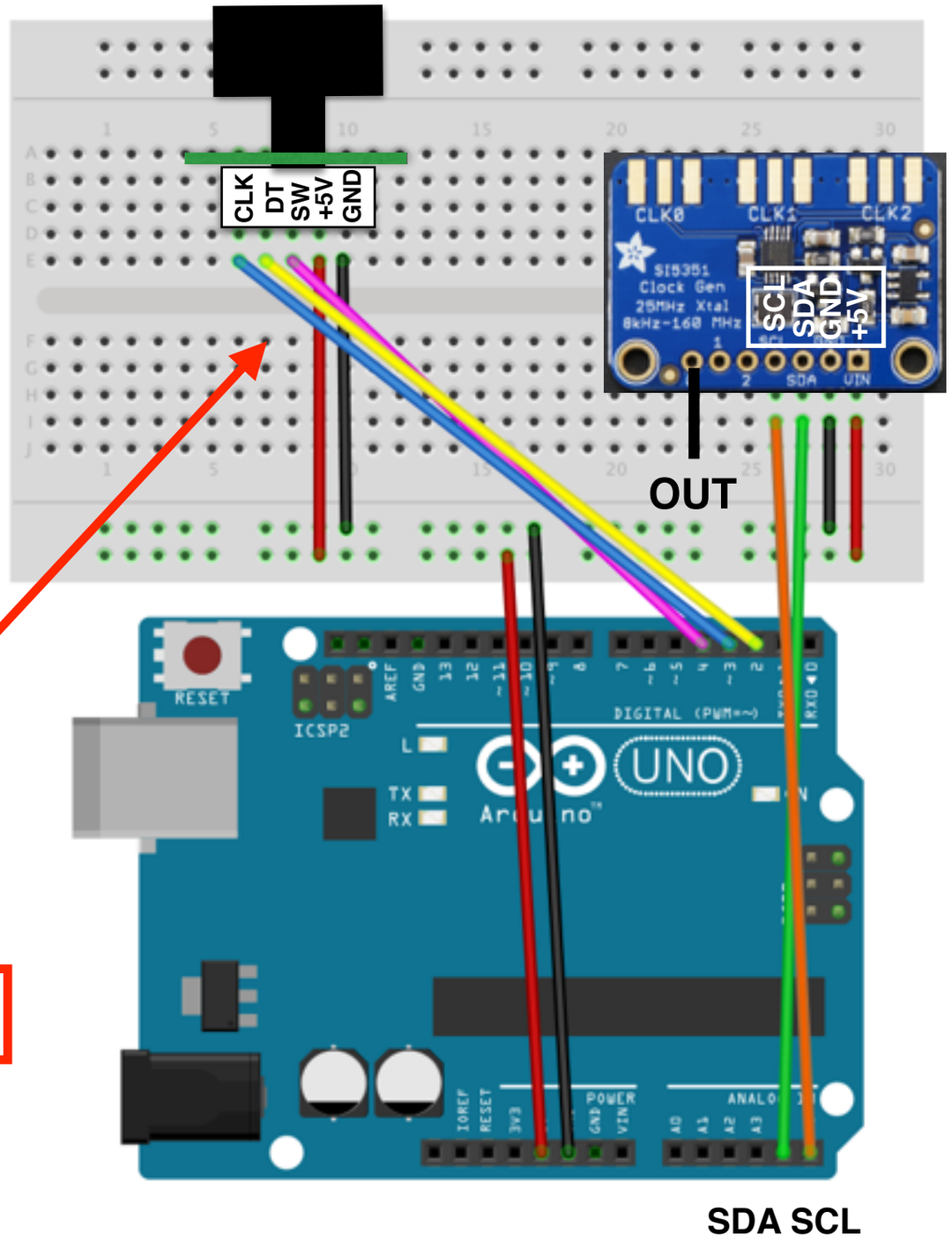


Out Rotary Encoder gives
20 pulses per revolution

Tuning the VFO

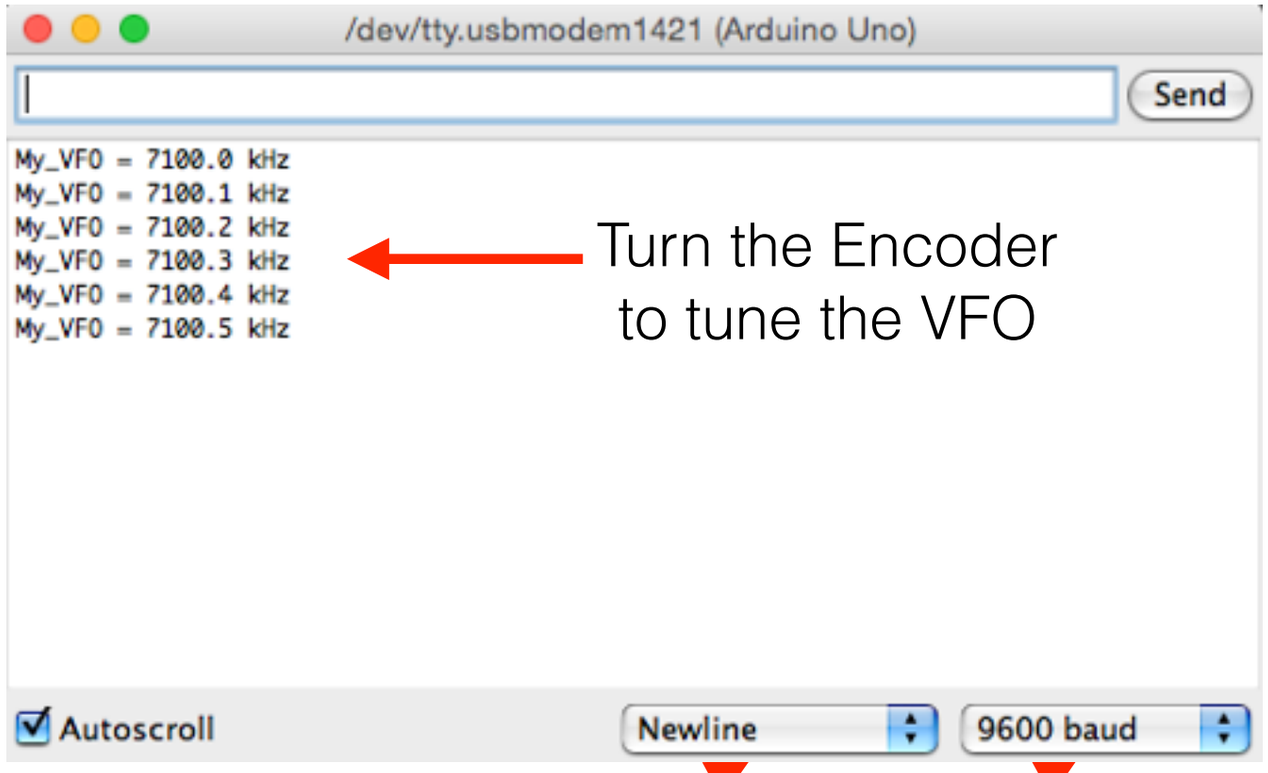
- Rebuild your VFO
- Add your Rotary Encoder
  `DT  = D2`
  `CLK = D3`
  `SW  = D4`
- Take care to get the connections correct
  `SCL = A5`
  `SDA = A4`
- Output is from CLK 0

Use a five-wire F-M cable



My_VFO_ROTARY

- File > Sketchbook > My_VFO_ROTARY
- Open the Monitor
- Your VFO will start at 7100.0 kHz
- Listen for this on your radio
- Turn the Encoder. Your VFO will tune in STEPS =100Hz
- Change band - push button



My_VFO_ROTARY

Library tunes Si5351 in 0.01Hz (cHz) steps

Include libraries for I2C, Si5351 & Rotary Encoder

Define tuning steps (cHz)

Encoder connections

Si5351 object

Rotary object

Start frequencies (cHz)

Band, init freq

```
// I2C, Si5351, LCD and rotary Encoder libraries
#include "Wire.h"
#include "si5351.h"
#include "Rotary.h"

// tuning freq STEPS (cHz), 100Hz
#define STEPS 10000

//Encoder 2 & 3 (DT & CLK), band change 4 (SW)
#define DT 2
#define CLK 3
#define SW 4

// dds object
Si5351 dds;

// rotary Encoder object
Rotary rot = Rotary(DT, CLK);

// start frequencies (cHz)
uint32_t freqStart[3] = {
  710000000, 1014000000, 1410000000
};

// band, freq (cHz)
byte band = 0;
uint32_t freq = freqStart[band];
```

A quick look at the code

Start serial for monitor

Init the Si5351 module

Enable CLK0

Define Encoder pins

Output freq & display

```
void setup() {
  Serial.begin(9600);

  // init dds si5351 module, "0" = default 25MHz XTAL
  dds.init(SI5351_CRYSTAL_LOAD_8PF, 0);

  // set 8mA output drive
  dds.drive_strength(SI5351_CLK0, SI5351_DRIVE_8MA);

  // enable VFO output CLK0, disable CLK1 & 2
  dds.output_enable(SI5351_CLK0, 1);
  dds.output_enable(SI5351_CLK1, 0);
  dds.output_enable(SI5351_CLK2, 0);

  // encoder, button, RX, TX, band and XMIT pins
  pinMode(DT, INPUT_PULLUP);
  pinMode(CLK, INPUT_PULLUP);
  pinMode(SW, INPUT_PULLUP);

  freqOut(freq); // output freq
  dispFreq(freq); // display freq
}
```

A quick look at the code

Main loop

```
void loop() {
  // tune?
  if (tune()) {
    freqOut(freq);
    dispFreq(freq);
  }

  // band?
  if (button()) {
    freq = freqStart[band];
    freqOut(freq);
    dispFreq(freq);
  }
}
```

Tune, returns ‘true’ if turned

```
// tune?
bool tune() {
  unsigned char dir; // tuning direction CW/CCW

  // turned?
  dir = rot.process(); // read encoder
  if (dir != DIR_NONE) { // turned?
    if (dir == DIR_CW) freq += STEPS; // increment freq +/- STEPS
    if (dir == DIR_CCW) freq -= STEPS;
    return true;
  }
  return false;
}
```

A quick look at the code

Band switch

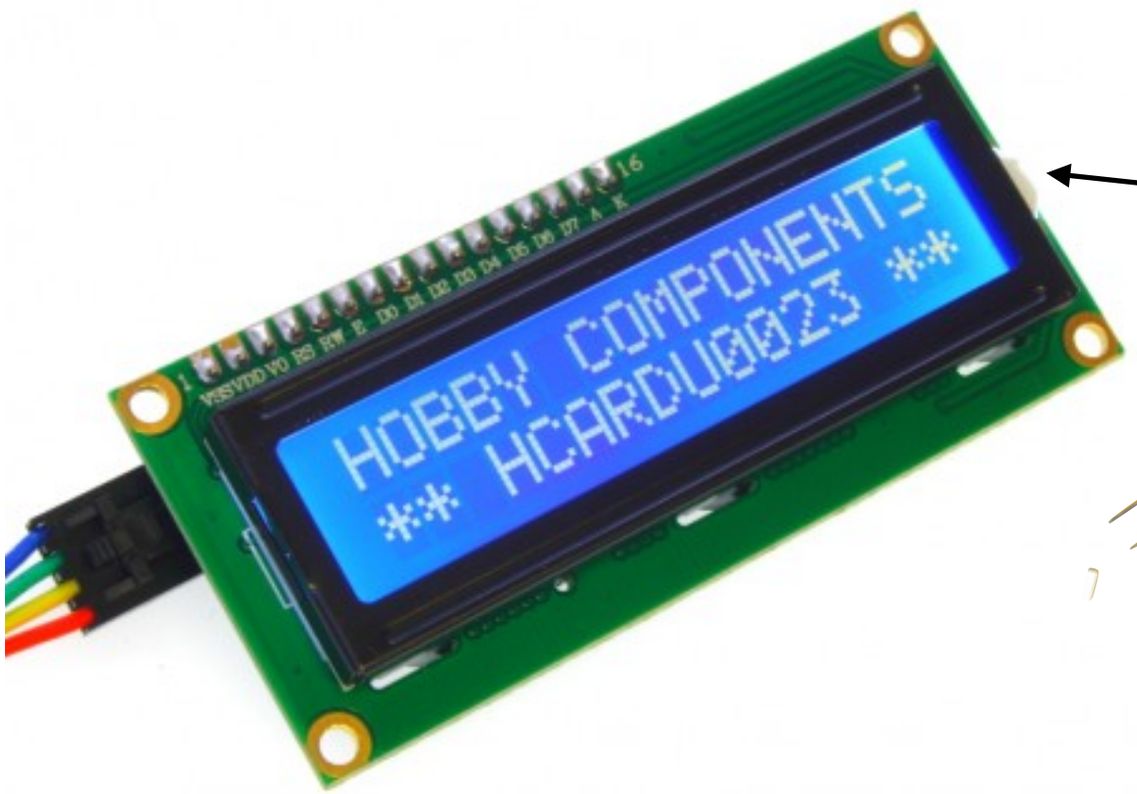
Output freq

Display freq (kHz)

```
// band?
bool button() {
  if (digitalRead(SW) == LOW) { // button pressed?
    while (!digitalRead(SW)); // wait for release
    if (band == 2) band = 0; // loop
    else band++;
    return true;
  }
  return false;
}

// frequency (in cHz) for VFO, on CLK0
void freqOut(uint32_t f) {
  dds.set_freq(f, 0ULL, SI5351_CLK0); // freq in cHz
}

// display freq in cHz
void dispFreq(uint32_t f) {
  Serial.print("My VFO = ");
  Serial.print((float)f / 100000, 1); // convert to float for print function
  Serial.println(" kHz");
}
```

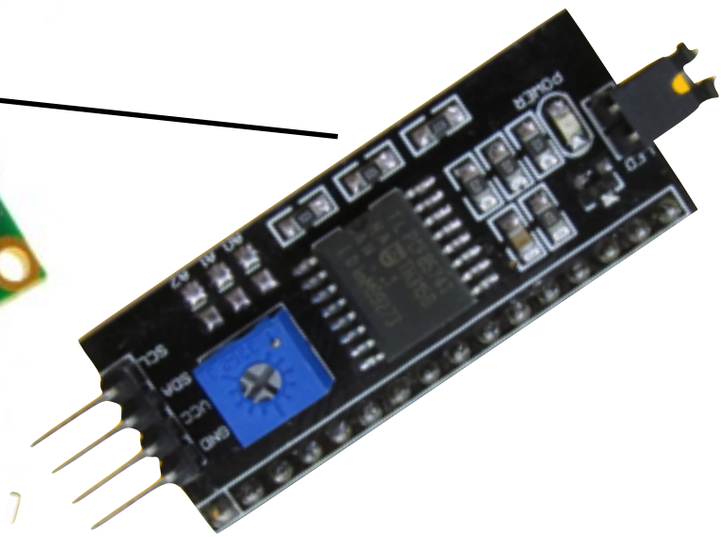
LCD Display

I2C converter is already mounted in Kit 2



LCD I2C

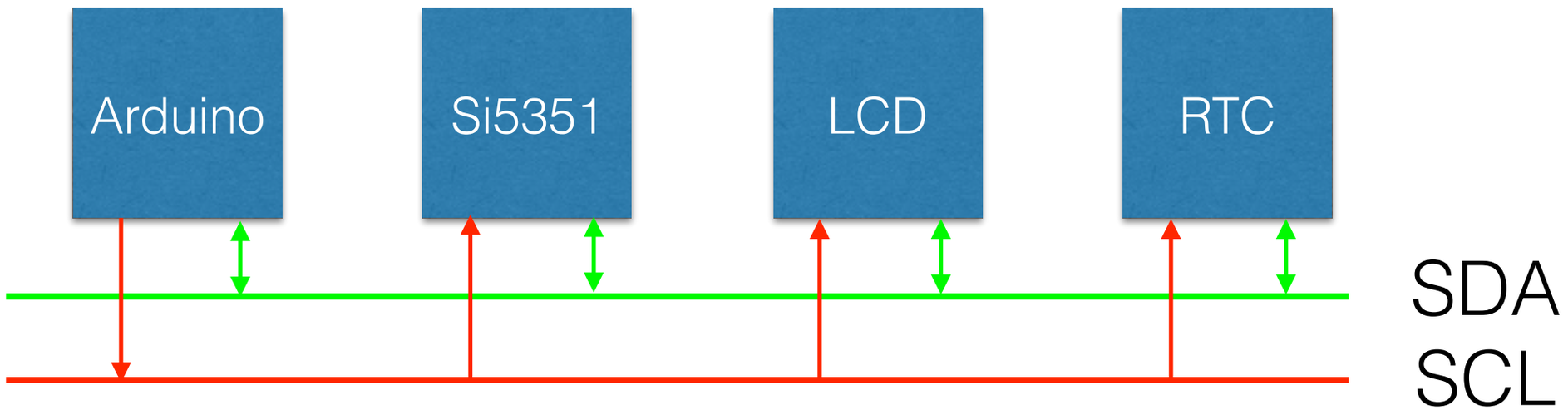
Mounted underneath



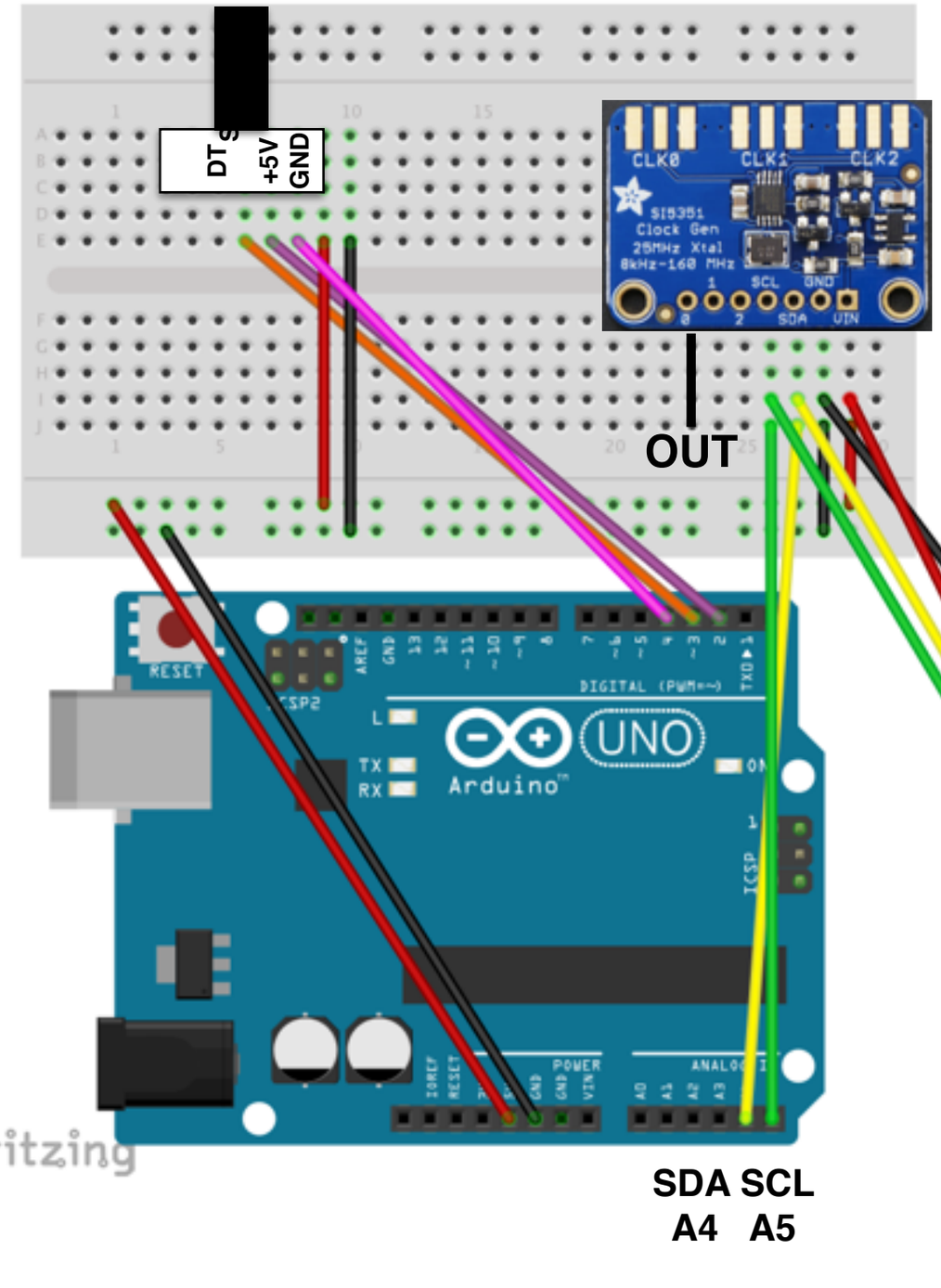
LCD I2C converter

The I2C bus

- The I2C bus was invented over 40 years ago, it was first intended for use in consumer Radio & TV to allow circuits to talk to each other
- It is a 2 wire bus, Clock (SCL) and bi-directional Data (SDA).
- Each circuit connected to the bus has a unique address. So that it can be individually addressed for reading or writing

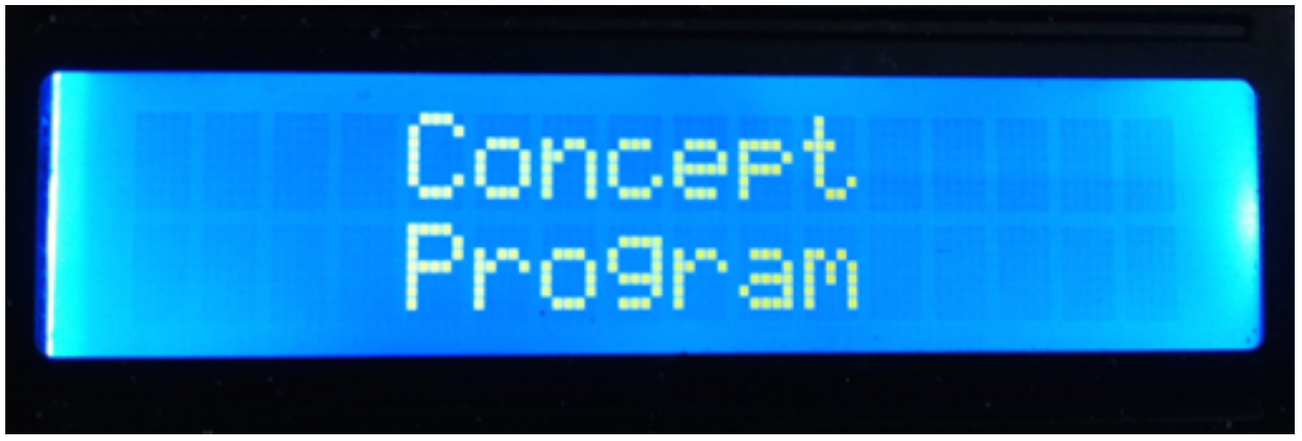


Test your LCD



- Add your LCD
- Take care to get the connections correct
- File > Sketchbook > My_LCD_test

My_LCD_test



Include libraries for I2C, LCD

LCD object

```
// 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);
```

Note: some LCDs have address 0x3F
If yours doesn't work, change this line

More code

Setup

LCD init

Output first line

Wait 2 sec

Output second line

Loop does nothing,
but it must be here

```
// setup runs once on upload
void setup() {
  // 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() {

}
```

Try the Encoder

- File > Sketchbook > My_VFO_ROTARY_LCD
- Your VFO will start at 7100.00 kHz
- Listen for this on your radio

- Turn the Encoder. Your VFO will tune in 100Hz steps.

  Push button to change bands (40, 30 or 20m)

- Check this on a radio

Code

- You can see the importance of the code
- Hardware designs are easy, what they *do* is defined by the software
- So we will increasingly focus on the code
- To enjoy this part of the hobby coding is essential

My_VFO_ROTARY_LCD

Include libraries for I2C, Si5351
Rotary Encoder & LCD

Define tuning steps (cHz)

Si5351 object

Rotary object

LCD object

Start frequencies (cHz)


```
// I2C, Si5351, LCD and rotary Encoder libraries
#include "Wire.h"
#include "si5351.h"
#include "Rotary.h"
#include "LiquidCrystal_I2C.h"

// tuning freq STEPS (cHz), 100Hz
#define STEPS 10000

// Encoder pins 2 & 3 (DT & CLK), band pin 4 (SW)
#define DT 2
#define CLK 3
#define SW 4

// dds object
Si5351 dds;

// rotary Encoder object
Rotary rot = Rotary(DT, CLK);

// lcd object
LiquidCrystal_I2C lcd(0x27, 16, 2);

// start frequencies (cHz), band names
uint32_t freqStart[3] = {
  710000000, 1014000000, 1410000000};

// band, freq (cHz)
byte band = 0;
uint32_t freq = freqStart[band];
```

Note: some LCDs have address 0x3F
If yours doesn't work, change this line

A quick look at the code

Init LCD

Init the Si5351 module

Enable CLK0

Define Encoder pins

Output freq & display

```
void setup() {

  // init LCD & backlight on
  lcd.init();
  lcd.backlight();

  // init dds si5351 module, "0" = default 25MHz XTAL
  dds.init(SI5351_CRYSTAL_LOAD_8PF, 0);

  // set 8mA output drive
  dds.drive_strength(SI5351_CLK0, SI5351_DRIVE_8MA);

  // enable VFO output CLK0, disable CLK1 & 2
  dds.output_enable(SI5351_CLK0, 1);
  dds.output_enable(SI5351_CLK1, 0);
  dds.output_enable(SI5351_CLK2, 0);

  // encoder, button, RX, TX, band and XMIT pins
  pinMode(DT, INPUT_PULLUP);
  pinMode(CLK, INPUT_PULLUP);
  pinMode(SW, INPUT_PULLUP);

  freqOut(freq); // output freq
  dispFreq(freq); // display freq
}
```

A quick look at the code

Main loop

Tune

```
void loop() {
  // tune?
  if (tune()) {
    freqOut(freq);
    dispFreq(freq);
  }

  // band?
  if (button()) {
    freq = freqStart[band];
    freqOut(freq);
    dispFreq(freq);
  }
}

// tune?
bool tune() {
  unsigned char dir; // tuning direction CW/CCW

  // turned?
  dir = rot.process(); // read encoder
  if (dir != DIR_NONE) { // turned?
    if (dir == DIR_CW) freq += STEPS; // increment freq +/- STEPS
    if (dir == DIR_CCW) freq -= STEPS;
    return true;
  }
  return false;
}
```

A quick look at the code

Band change

Freq out

Display LCD

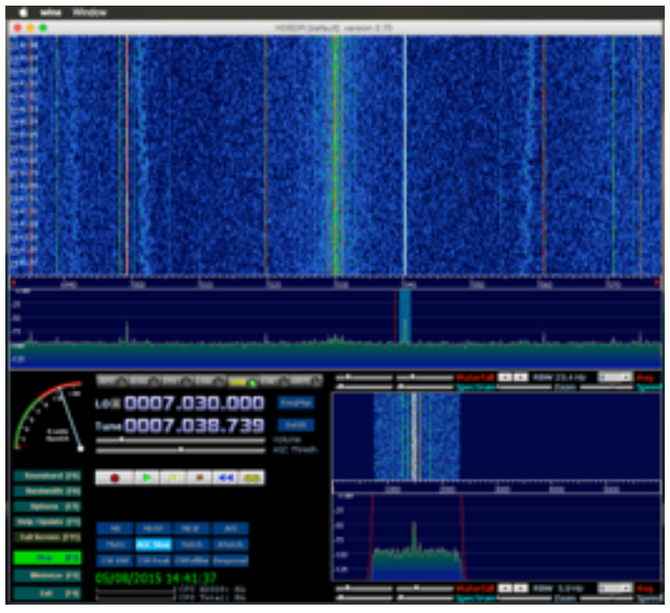
```
// band?
bool button() {
  if (digitalRead(SW) == LOW) { // button pressed?
    while (!digitalRead(SW)); // wait for release
    if (band == 2) band = 0; // loop
    else band++;
    return true;
  }
  return false;
}

// frequency (in cHz) for VFO, on CLK0
void freqOut(uint32_t f) {
    dds.set_freq(f, 0ULL, SI5351_CLK0); // converted to cHz
}

// display freq in cHz
void dispFreq(uint32_t f) {
  lcd.setCursor(0, 0);
  lcd.print("VFO                  ");
  lcd.setCursor(4, 0);
  lcd.print((float)f / 100000, 1); // convert to float for print function
  lcd.setCursor(13, 0);
  lcd.print("kHz");
}
```

Play time!

Note: some LCDs have address 0x3F
If yours doesn't work, change the address

- Try these sketches
  - My_SDR_40M - runs at f x 4 to produce IQ signals, displays band plan
  - My_VFO_40M - outputs f, displays band plan
- Add your RTC module to the breadboard, set its time accurately then
  - My_WSPR_40M WSPR generator, use with WSPR_Symbol_Generator

HDSDR



WSPR

Being practical

- Next you will be building the VFO shield, this carries an SMD the SN74AC74
- Check you have a suitable soldering iron. Wire cutters. A small set of tweezers, a magnifier would be useful
- Hobby components has a very good soldering iron (40W temp controlled). Amazon has Solder Flux Pen + 0.3mm Solder and magnifiers.

BARS

Practical build time
You need all your tools

Next

You build your 1st shield
The VFO_RTC_IQ