Coding and amateur radio

Digital VFO by M0KUK

Using microprocessors opens a new ability to easily create and modify the tool to provide bespoke solution. Every receiver, transmitter and transceiver use some version of a frequency generator. However, usage of readily available digital modules in connection with Arduino platform provides quick and easy way to create a very useful device.

About 1.5 Hours

Kit List

- Arduino board (e.g. Arduino Uno, however almost any Arduino compatible board will work including Teensy and ESP32)
- LCD screen I^{2}C
- Si5351
- Jumper/dupont wires
- Breadboard
- Computer with Arduino IDE
- Some kind of dummy load, might be 50 Ohm resistor.

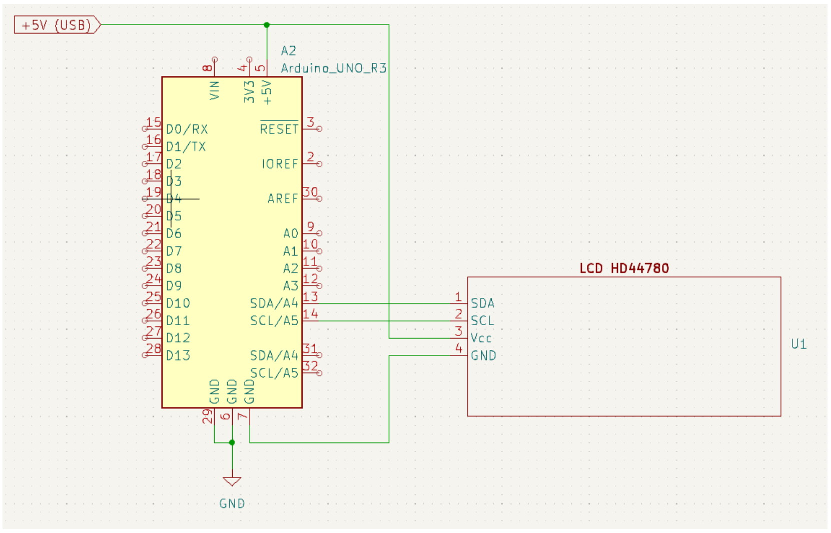
Instructions

If you don't have Arduino IDE please download and install it. Latest version is available on the Arduino webpage: www.arduino.cc/en/software or you can use Arduino Web Editor available from the same link.

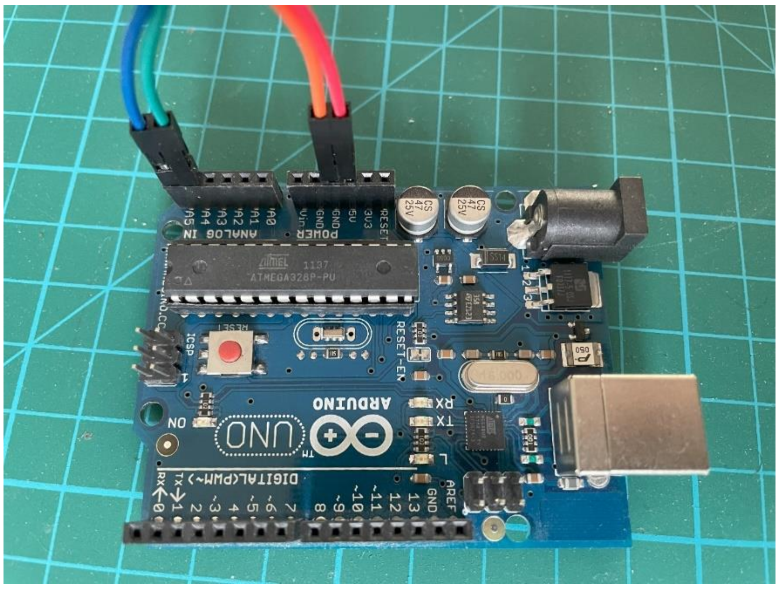
For this example, we will be using Arduino Uno board, but any other Arduino board can be used. Please remember to adjust the connection pins to the board that you are using as this will be different for different boards.

Step 1. Display

1. Connect your LCD to Arduino board, you might want to use breadboard for that.



LCD Pins	Arduino UNO
Vcc (5V)	**5V**
GND	**GND**
SDA	**A4**
SCL	**A5**



2. Open Arduino IDE and start new project

3. Add LiquidCrystal_I2C library to your sketch, use menu *Tools* -> *Manage Libraries* and search for the library name. Then click *Install*.

4. Once it's completed, start with adding required library to the sketch and defining the LCD screen in the very first lines:

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2); //Define your LCD (16 column and 2
rows) using the library, I2C address 0x27
```

5. Then initialise the screen and turn on the backlight. This should be the `setup()` part of the sketch

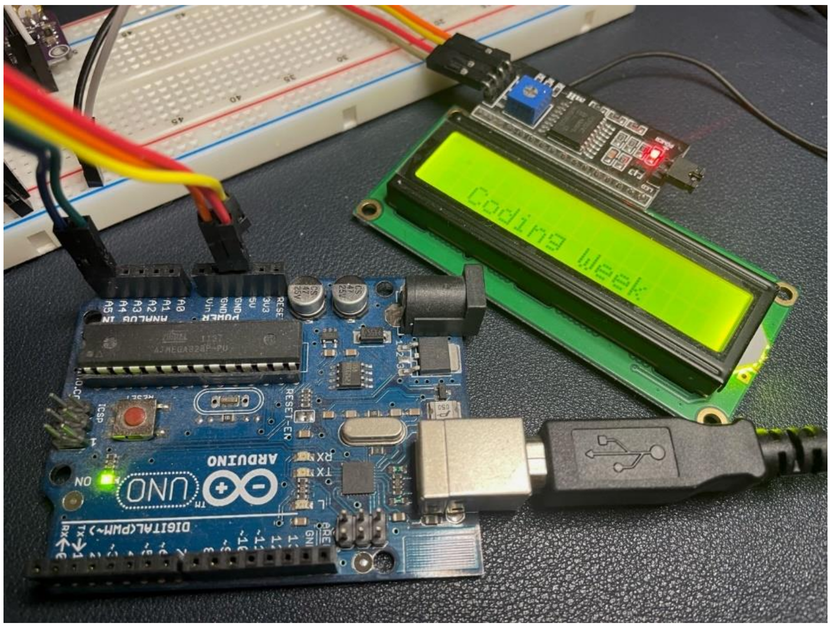
```
  void setup()
{
  lcd.init(); //Initialize screen
  lcd.backlight();
}
```

6. In main `loop()` after `setup()` we will clear the screen and display (print) the message

```
void loop()
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Hello World!");
  lcd.setCursor(2, 1);
  lcd.print("Coding Week");
  delay(10000);  //wait for 10 seconds before restarting the loop
}
```

7. Connect your Arduino Uno board to your computer and click *Upload*. After a short while you should be able to see the result on the LCD screen.

Note: Full code can be found here: https://github.com/M0KUK/DigitalVFO/blob/master/step_one.ino



Next Steps

- Try changing text and its position
- Try displaying your call sign and grid locator
- Try changing the text visible on the screen every three seconds

Step 2. Serial input

In this step we will add the ability to display data provided from connected computer using serial terminal.

1. Add command to initialise serial connection with a preferred baud rate. For the purpose of this exercise, we'll use low 9600 speed. New setup() should look like this:

```
void setup()
{
  lcd.init(); //Initialize screen
  lcd.backlight();

  Serial.begin(9600);  //Initialize serial connection
}
```

2. In main loop we will add ability to read from serial input

```
void loop()
{
  if (Serial.available()){
   inputText = Serial.readString();
  }
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(inputText);
  lcd.setCursor(2, 1);
  lcd.print("Coding Week");
  delay(500);  //wait for half a second before restarting the loop
}
```

3. Please note that we also need to add definition of the variable:
`String inputText = "";`
As we want it to be usable globally, i.e. in any part of our program, we will add this line just before definition of our screen at the very beginning of the file.

4. Now try uploading the sketch and afterwards open *Serial Monitor* from Arduino IDE. When you type some text and press Enter you should see what was typed on your LCD screen.
Note: The full script for this step can be downloaded from here:
https://github.com/M0KUK/DigitalVFO/blob/master/step_two.ino

Next Steps

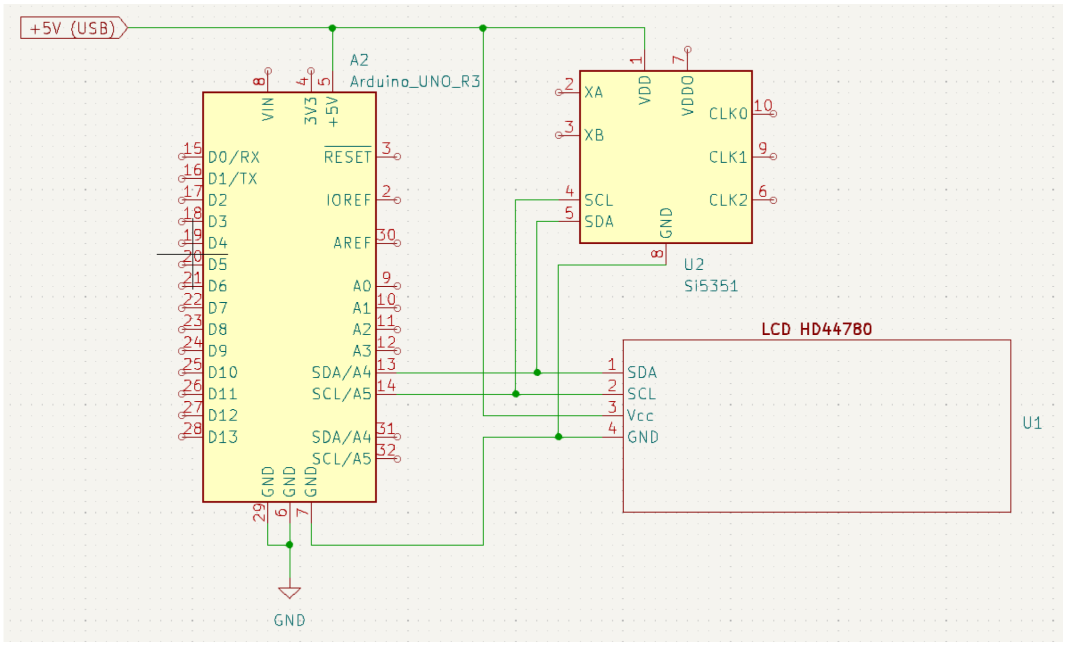
- Try different text positions
- Try to make sure that the text is always displayed in a way that ends in the same place of the screen
- Try controlling the text displayed via commands or keys pressed
- Try adding scrolling screen effect if the input text is too long.

Step 3. Adding frequency generator

Once we created interface to our VFO and we not only can control it, but we can display messages, it’s time to add the heart of our digital VFO: Si5351 module

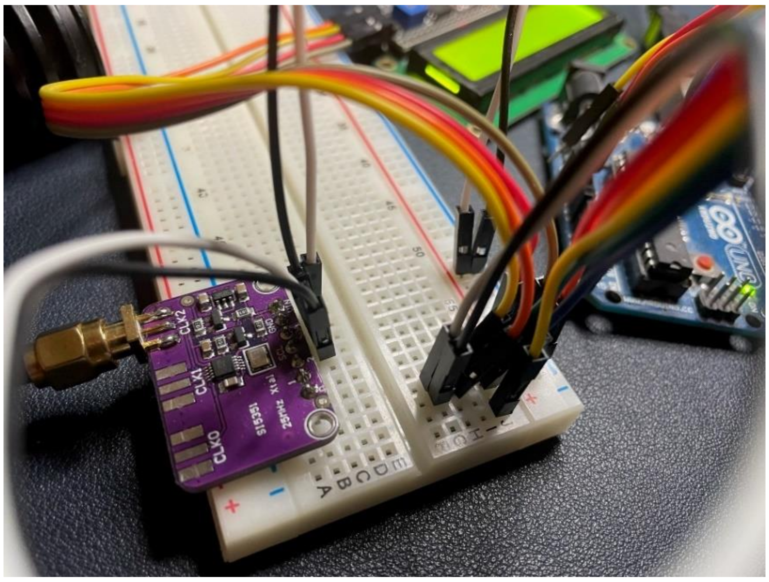
1. Connect Si5351 module to breadboard so that:

SI5351 Pins	Arduino UNO
Vin	**5V**
GND	**GND**
SDA	**A4**
SCL	**A5**



One of useful features of I^{2}C is the fact that multiple devices can be connected on the same ‘rail’ as most devices will have different address (recall ‘0x27’ of the LCD when we initialised it in code.

Please make sure that your version of Si5351 works with 5V (most do) as some modules can only be provided 3.3V. Fortunately some Arduino boards provide 3.3V pin.



2. Now we need to install another library to our project. This time we will use Etherkit Si5351 library by Jason Milldrum.

3. Once added we need to include reference to the libraries in the beginning of our sketch and define our module. Thus, first lines of code will look like this:

```
#include <LiquidCrystal_I2C.h>
#include "si5351.h"
#include "Wire.h"

String inputText = "";
uint64_t frequency = 701000000ULL; // 7.030 MHz, in hundredths of hertz

LiquidCrystal_I2C lcd(0x27, 16, 2); //Define your LCD (16 column and 2
rows) using the library, I2C address 0x27
Si5351 si5351;
```

We also added new variable to store current frequency that we want to generate.

4. To initialise and add following lines to `setup()` section:

```
  // The crystal load value needs to match in order to have an accurate
calibration
  si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0);

  //Start on target frequency
  si5351.set_pll(SI5351_PLL_FIXED, SI5351_PLLA);
  si5351.set_freq(frequency, SI5351_CLK2);
```

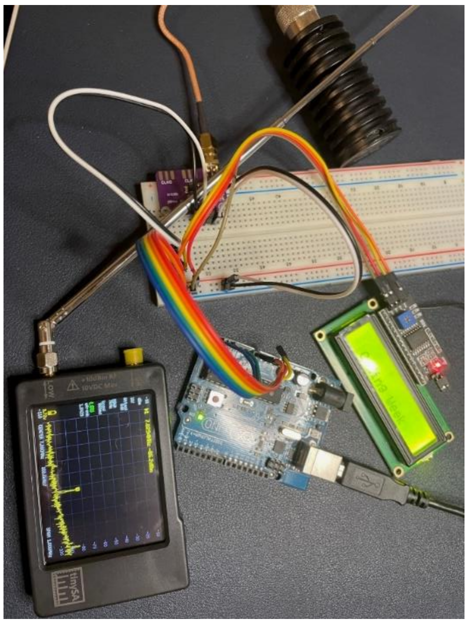
5. Now it we can test our code by uploading it to the board.

As with all radio waves we will need some test device to verify if our VFO works properly, however anything can be used starting with frequency counter, oscilloscope, spectrum analyser or even a radio receiver.

6. As the final step we can modify our loop() to convert serial input to new frequency.
Note that we made an assumption here that the input will be a number and it will be a frequency in kHz, for example 7030. This code is simplified and does not handle any input validation.
Final version of `loop()` can look like this:

```
void loop()
{
  //We expect frequency in kHz, e.g. 7030 to be given on input
  if (Serial.available()){
   inputText = Serial.readString();
  }
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(inputText);
  lcd.setCursor(2, 1);
  lcd.print("Coding Week");
  //Calculate new frequency based on the input
  frequency = inputText.toInt() * 100000ULL;
  //Change the frequency on Si5351 module
  si5351.set_freq(frequency, SI5351_CLK2);
  delay(500);  //wait for half a second before restarting the loop
}
```

Note: Full script can be found here:
https://github.com/M0KUK/DigitalVFO/blob/master/step_three_last.ino



Next Steps

- Try adding improved formatting on the LCD
- Try adding validation to input
- Try adding instructions to users
- Try adding rotary encoder or buttons to provide computer-free control
- Try turning the VFO on and off to send Morse
- Calibrate your VFO**.**

More Information

- Full code and all the steps can be found here: https://github.com/M0KUK/DigitalVFO
- Additional Arduino resources and examples are available here: https://www.arduino.cc/reference/en/
- Etherkit library documentation can be found on github: https://github.com/etherkit/Si5351Arduino
- Interesting projects based on the same concept are presented by Charlie Morris, ZL2CTM on his blog and YouTube channel https://www.youtube.com/@CharlieMorrisZL2CTM