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,

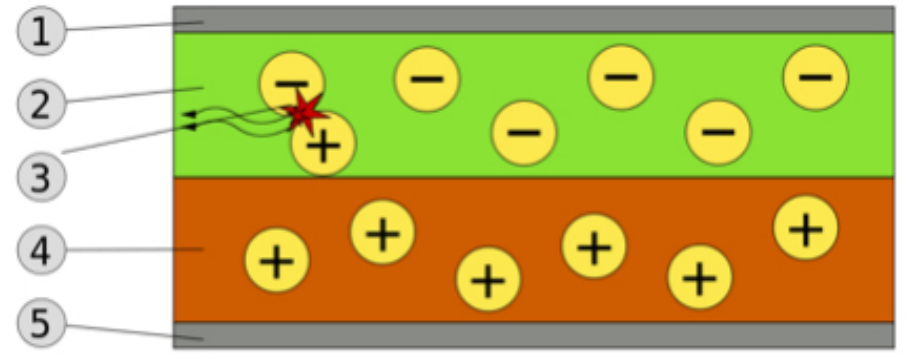
* the use of the Arduino Nano
* the build of a digital Signal Generator / VFO, an RF Volt/power meter, a Direct Conversion Receiver.

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

3. ARDUINO

OLED displays

An OLED (Organic Light Emitting Diode) is a sandwich of pixels with an organic negative electroluminescent emissive layer and a positive conductive layer, each pixel has a cathode and anode connection.



Schematic of a bilayer OLED: 1. Cathode (–), 2. Emissive Layer, 3. Emission of radiation, 4. Conductive Layer, 5. Anode (+)

The top cathode is transparent.

We are using a small 1.3" OLED, with a built-in controller chip, the SH1106. It is 5V operation and controlled over a two wire I2C bus. It has 128 x 64 pixels.

I2C bus
A quick word about the I2C bus. This is a two wire bi-directional serial data bus, one wire is the clock (SCL) an the second the data (SDA). More than one I2C device can be connected to the same bus. Each has a it's own unique address. On the Arduino there is a library called `"Wire.h"` which drives pins A4 (SDA) and A5 (SCL) with the correct bus signals.

So there are four connections from the Nano to your OLED display (which will be shared later with the Si5351 and the RTC, both of which have i2C interfaces)

GND -> GND
5V -> VCC
A5 -> SCL
A4 -> SDA

The Oled.h header file
We are going to use a public library u8g2. This is extensive, but also rather complex, as it has many display functions, many fonts and can support multiple display types. For this reason we have our own header file `"Oled.h"` just for our OLED with SH1106 controller, and containing a series of functions & fonts useful for our applications. Below is a summary of the `Oled.h` function descriptions and the meaning of the variables used.

Oled.h functions

The first line creates an object `"oled"` with the correct class for the OLED we are using.

```
U8G2_SH1106_128X64_NONAME_1_HW_I2C oled(U8G2_R0);

// functions & usage
void dispBar(u8g2_uint_t x, u8g2_uint_t y, byte h, byte l)
void dispScn(u8g2_uint_t sx, u8g2_uint_t sy, uint64_t *a)
void setPix(u8g2_uint_t x, u8g2_uint_t y, uint64_t *a)
void dispFreq(u8g2_uint_t x, u8g2_uint_t y, double f, double cf, byte d)
void dispStep(u8g2_uint_t x, u8g2_uint_t y, unsigned int s)
void dispMsgS(u8g2_uint_t x, u8g2_uint_t y, char *m)
void dispMsg(u8g2_uint_t x, u8g2_uint_t y, char *m)
void dispMsgL(u8g2_uint_t x, u8g2_uint_t y, char *m)
void dispMsgUL(u8g2_uint_t x, u8g2_uint_t y, char *m)
void dispNum(u8g2_uint_t x, u8g2_uint_t y, double n, byte d)
void dispNumL(u8g2_uint_t x, u8g2_uint_t y, double n, byte d)
void dispNumUL(u8g2_uint_t x, u8g2_uint_t y, double n, byte d)
void dispDate(u8g2_uint_t x, u8g2_uint_t y, byte dw, byte da, byte mo, byte yr)
void dispTime(u8g2_uint_t x, u8g2_uint_t y, byte h, byte m, byte s)
void dispTimeL(u8g2_uint_t x, u8g2_uint_t y, byte h, byte m, byte s)


x, y  display position (top left, down)
h     height, time hour
l     length
f     Hz freq
cf    cHz freq
d     freq phase, others decimal place
*m    message
n     number
dw    day of the week
da    day
```

```
mo     month
yr     year
m      minute
s      second, or step, or screen array pointer
```

As you can see each function takes input parameters, but some are of a strange type. As integers can have different sizes on different computers (16bit, 32bit, 64bit) the `u8g2` defines its own variable type it calls `u8g2_uint_t`. Essentially this is an unsigned integer suitable for use on any machine including the Nano, which is a 16 bit machine.

Now, as an example, if we want to display on the OLED `"HELLO WORLD"` in a large font we can say

```
dispMsgL(x, y, "HELLO WORLD");
```

X and Y are the coordinates of the top left of the text being displayed, measured from the top left of the display across (X) and down (Y). So X = 10 and Y = 30 starts the top/left of the `"H"` text at 10 pixels in from the left, and 30 pixels down.

To use the `Oled.h` header you must “#include" it at the top of your sketch.

My_OLED

```
1 // OLED
2
3 #include "Oled.h"
4
5 void setup() {
6   oled.begin();
7   dispUpdate();
8 }
9
```

Then you must initialise the OLED by the function `oled.begin();` after that you can use any of the header functions.

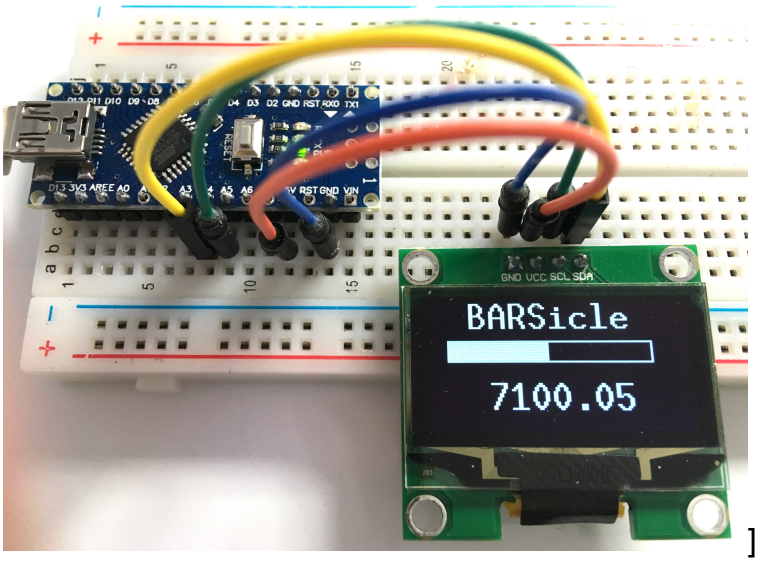
The display is refreshed using the `dispUpdate();` function in your sketch, which accesses the variables to be displayed. These are passed to it by making them global variables. or as literals.

Here's an update example, with the variables literally put in, they could be passed as global variables

```
void dispUpdate() {
  oled.firstPage();
  do {
    dispMsgL(20, 0, "BARSicle");
    dispBar(10, 20, 10, 50);
    dispNumL(30, 40, 7100.05, 2);
  } while ( oled.nextPage() );
}
```

Run an example
With the OLED display wired up, Open and Upload the sketch

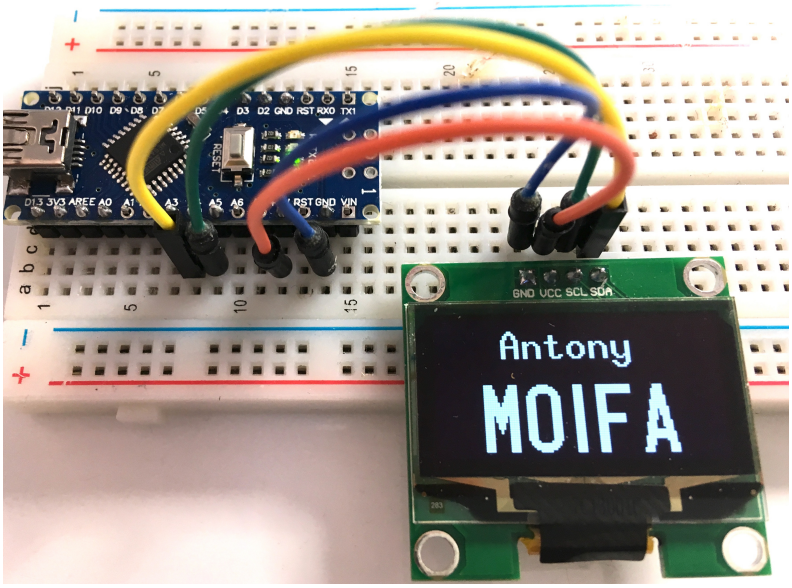
`File > Sketchbook > My_OLED_EXAMPLES`



]
Have a good look at the code for this display to see how the functions are used.

Now try the sketch

`File > Sketchbook > My_OLED_NAME_CALL` this will display



Modify the sketch code to display your own name and call sign.

A voltmeter
Since the Nano has analog inputs we can measure external voltages (not < 0, not > +5V or damage will occur). A simple way to display the voltage is on our OLED.

Open

File > *Sketchbook* > *My_VOLTS_OLED* to see how this is done.

My_VOLTS_OLED §

```
// VOLTS_OLED
// Measure +5V line and use below (mine was 4.62)

#include "Oled.h"                                   // include header
float volts;

void setup() {
  oled.begin();                                     // begin OLED
}

void loop() {
  volts = analogRead(A0) * 4.62 / 1023;             // read ADC -> volts, "+5V" ref
  dispUpdate();                                     // display
  delay(100);
}

void dispUpdate() {                                 // display loop
  oled.firstPage();
  do {
   dispNumUL(40, 10, volts, 2);                     // display volts, 2 dec places
   dispMsgL(40, 45, "VOLTS");                       // display string
  } while ( oled.nextPage() );
}

```

There is a floating point global variable "`volts`" used to pass the value from `loop()` to the `dispUpdate()` function. The OLED is initialised in `setup()`. The analog input is converted from a digital 0-1023 to an analog floating point number "`volts`". The ADC uses the actual +5V line as a reference (for me this was actually 4.62V).