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. With the objective of using the WSJT-X program to receive digital FT-8 signals

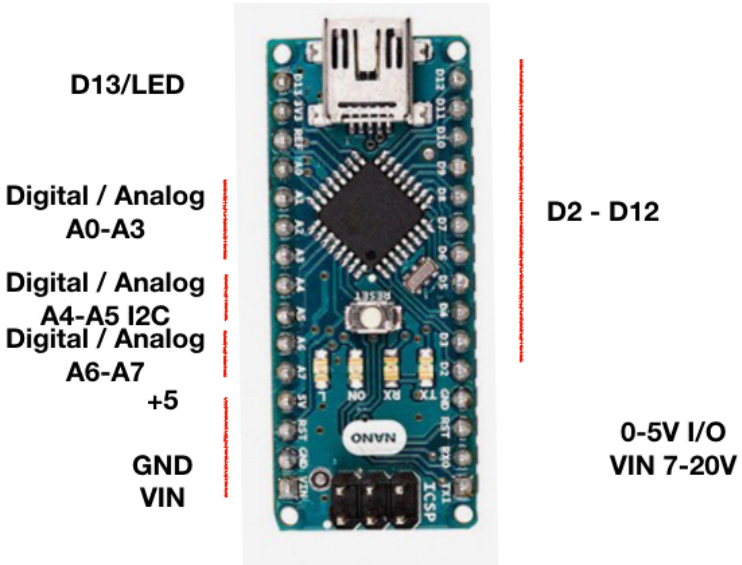
An extension of the project will cover an SSB Exciter, Power Amplifier with Low Pass Filter and an SWR/Antenna Tuning Unit

2. ARDUINO

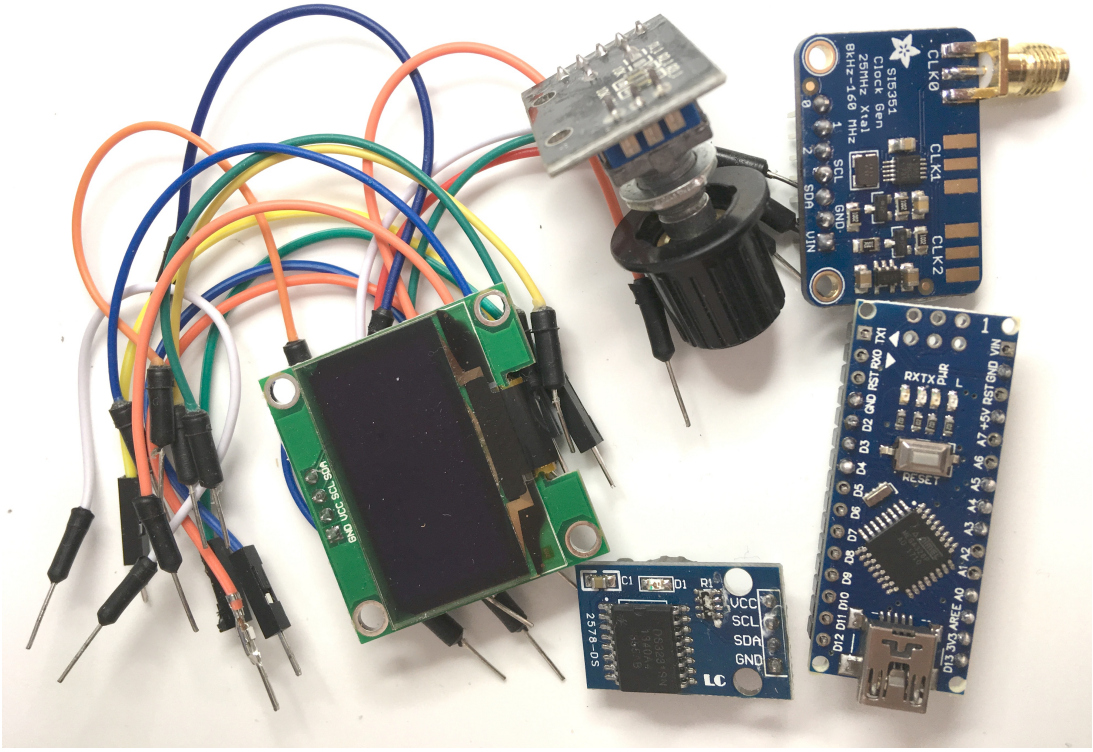
The Arduino Nano, setup and first sketches

We have covered the use of the software development IDE and learnt the basics of writing a sketch. Time to tackle the real world, Not software but hardware. A world you might be more familiar with.

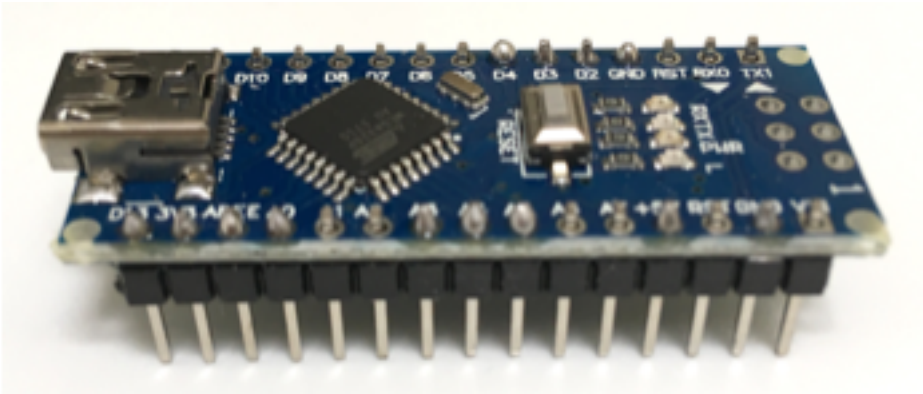
We are going to power up and run the Arduino Nano. The Nano is a microcomputer, it runs at 16MHz (slow by PC standards!) and has small memories of 2kB RAM and 32kB Flash (SSD) memory - 2 kB of which is used for talking to your PC over the USB bus for loading programs, so 30kB available for you. It has lots of input/output pins, 22 altogether. 8 pins can be used for digital I/O or analog input A0-A7, of which A4 & A5 can be used for an I2C serial bus,. It has 12 digital I/O pins D2-D13., of which 6 can be programmed to output PWM signals (when smoothed these are analog outputs), Pin D13 has an onboard red LED connected. The final 2 digital pins are TX & RX for communication with the USB interface through a dedicated interface FTDI.



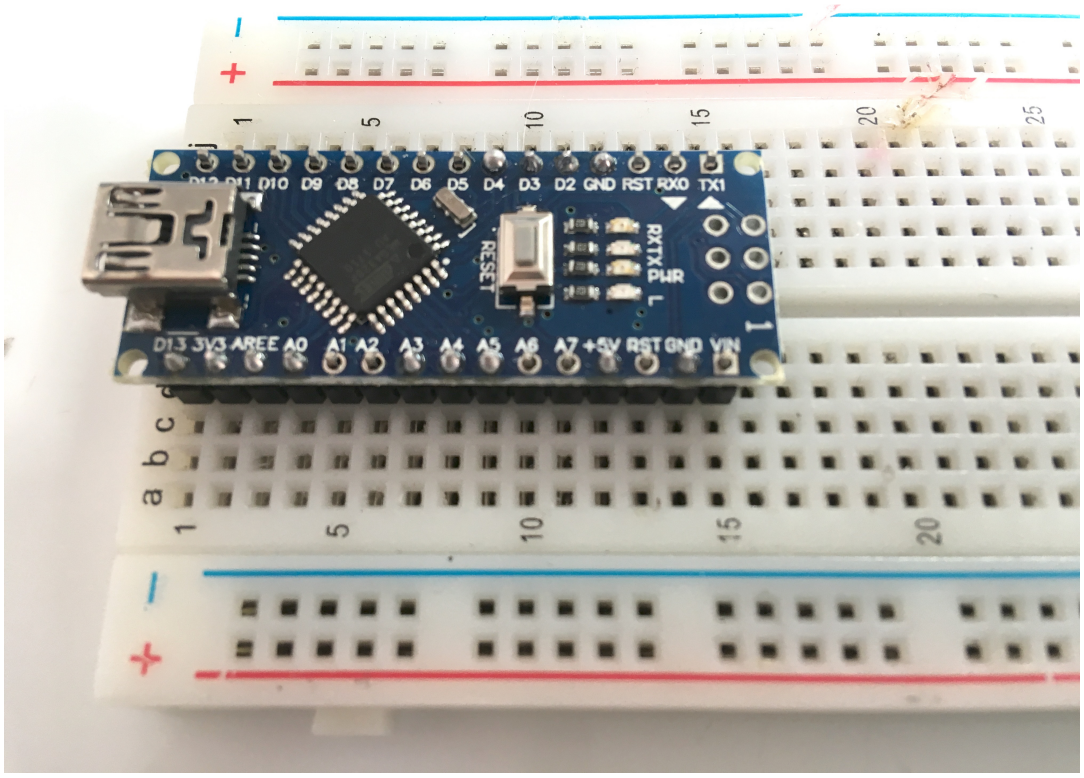
Readying the Nano
These are your Kit parts



The Nano is delivered in a anti-static packet along with 2 x 15pin headers. These should be inserted in the connections of the Nano and carefully soldered, taking care not to bridge adjacent pins. Make sure you get them vertical - you could plug them into the breadboard and solder them there.



Plug the Nano into your breadboard at the left hand end. The pins may be a bit hard to push in, but persevere, give them a good shove.



Connect your Nano to your PC using the USB cable. The green power LED will light.

Configure the communications with your PC.
Start the Arduino IDE program. Go to the menu T*ools* > *Board* and select *"Arduino Nano",* then go to *Tools* > *Port* and select the USB port your Nano is plugged into. [on a PC this will be a COMn port, on a Mac it will look like /dev/cu.name).

Test the Nano
At last we can run a sketch! Open the *File* > *Sketchbook* > *My_BLINK* sketch. Now check the code is without errors by hitting the *Compile* button (1st on the top left), no errors should be reported, if there are they must be corrected now. At last hit the *Upload* button. The program will compile then upload to your Nano, the TX/RX lights will flash. *My_BLINK* is a sketch that blinks the on-board red LED at 1 sec intervals.

Check out the code to see how it does it.

Your turn
Now write your own sketch to flash the LED 2 times for 100ms then one time for 1sec and repeat. Use a function of your own to flash the LED given the time as an input.

If in doubt Upload the *File* > *Sketchbook* > *My_BLINK_1*.

My_BLINK_1

```
// BLINK 1

#define LED 13

void setup() {
  pinMode(LED, OUTPUT);        // set pin 13 as an output
}

void loop() {
  flash(100);
  flash(100);

  flash(1000);
}

void flash(int t) {
  digitalWrite(LED, LOW);      // pin 13 LOW
  delay(t);                    // wait t (ms)
  digitalWrite(LED, HIGH);     // set pin 13 HIGH
  delay(t);                    // wait t (ms)
}

```

Done uploading.

Now try connecting an external resistor and LED from D4 -> 220R -> LED -> GND. Modify your program to flash this LED (hint change the #define LED pin number). The external LED will flash to the rhythm.

Now modify the program to flash "CQ" in morse (dot = 300ms, dash = 900ms, space = 1200ms). You will have to add a new *delay(1200);* to make the inter-character space... You could connect an external relay to the output pin and key your TX.

Voltmeter
The Nano has 8 analog inputs `A0- A7.` We can make a voltmeter using an analog input. The analog input on `A0` is read by the function `analogRead(A0);` this return an integer value from 0-1023. The reference for the ADC is the +5V supply line, which may not be a very accurate or stable +5V..

So the first thing to do is to measure the actual +5V line value, mine was 4.62V. This value is put in the code for the conversion

```
volts = (float)(analogRead(A0) * 4.62 / 1023);
```

Note that `analogRead();` returns an integer, this is converted to a floating point number `"volts"` to display the decimal points

Here's the code

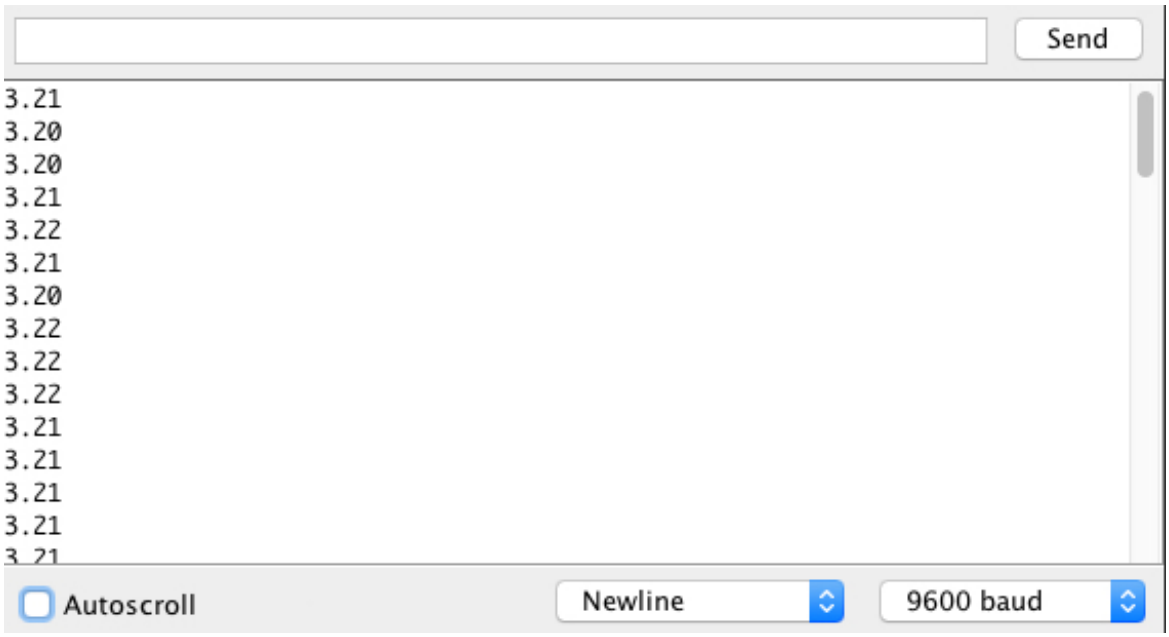
My_VOLTS §

```
// VOLTS
// measure +5V output, mine was 4.62
float volts;                                    // ADC voltage read

void setup() {
  Serial.begin(9600);                           // start serial comms
}

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

To see the voltage we need to display it. This can be done on your PC. There is a window that can be opened by the right hand button on the menu bar. This opens a serial communication window to the USB connection.



The code `Serial.begin(9600);` sets the baud rate, and `Serial.println(volts);` displays the `"volts"` variable.

Load the sketch *File* > *Scetchbook* > *My_VOLTS* . Upload it. A convenient voltage to measure is the Nano `3V3` output line so connect `A0` to `3V3`. Again you might fined the 3V3 output is not actually 3V3, in my case it is 3.21V.