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

1. ARDUINO
Introduction, PC setup, IDE install, Load sketches and libraries from USB stick, Using IDE, Basic sketches.

This first topic of the BARSicle project covers an introduction to the Arduino Nano, the use of OLED displays, of Rotary Encoders and Digital Frequency Synthesisers. We start here with an introduction to the Arduino.

The Nano

There are many models of the Arduino, the one we will use is the Nano. This is a small 16MHz microcomputer with a USB interface for power and programming, it can also be powered by an external 7-20V supply. It has 2KB RAM and 32kB Flash memory, 30kB of which is available for user programs (called sketches in Arduino-speak). It has 22 lines which can be used for digital Input/Output: 8 of these can also be analog inputs, another 6 can also be digital PMW outputs. It is connected to your PC/Mac with a USB A to USB Mini cable for sketch uploading and 5V power supply

Note that on the market there are two types of Nano, some Chinese versions with a different USB serial interface chip, the CH340, and the original Nano which uses an FTDI chip. Make sure you have the FTDI version.

Arduino software

The Arduino is supported by a complete "Integrated Development Environment" or IDE.

The IDE combines 3 items - a code editor, a compiler and an uploader. The code editor is where you write your programs. The compiler is chosen to match the version of the Arduino

you use (or many other non-Arduino microcomputers which are supported), and the uploader also matches your chosen microcomputer.

For those just starting out, a compiler is a piece of software that takes your input, human readable, code and translates it to machine code, in binary which your Nano understands.

```
if( x > 5) step = 1000;  ⟶  0100101001000100
```

Installation

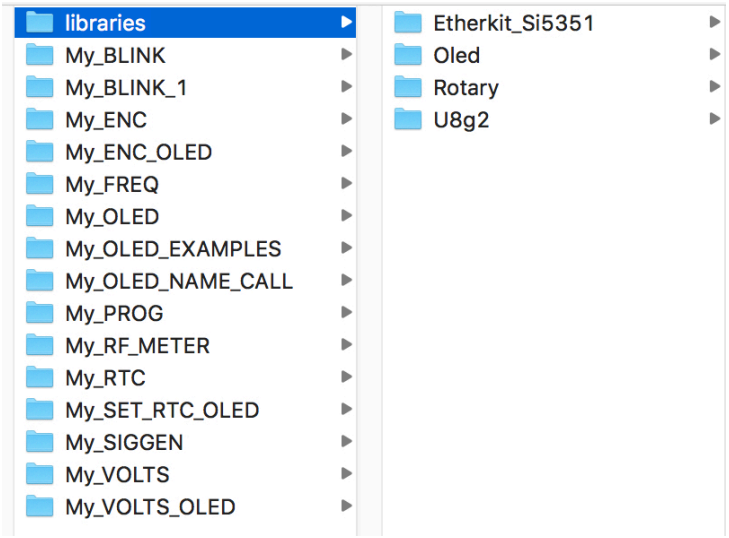
The home site of the Arduino is at arduino.cc. Here you will find a menu "Downloads" and on this page you can select a download for your operating system - Windows, MacOS or Linux, the latest version is 1.8.5 when this was written. Chose the correct one and commence the download. You can make a small contribution to Arduino before you do so if you wish. Follow the normal installation procedure for your OS.

Preferences

The Arduino installation will create a new folder in Documents > Arduino. This is where you will keep all your sketches and libraries. But you must check that the IDE knows where this is. To do so start the program and go to Preferences. Here enter, or browse for, the Arduino folder and put it in the box "Sketchbook Location". Click OK.

USB Stick with course project

The sketches and the libraries we are going to use are on the USB stick (or available as a download on the BARS web site). They should be copied to your Document > Arduino folder, which will then look like this



A word about sketches and libraries.
Sketches are kept in individual folders of the same name. Libraries are also kept in individual folers.

Libraries can be of two sorts, a simple "header" file which contains pre-written functions you can add to your code. We will see later the header file `Oled.h` that contains functions for displaying text, numbers and other things on an OLED display.

A different sort is a "class" library that enables you to create "objects" with certain features. For example, say we had a library called `"Make_Car.h"`, This would allow us to create a class of object which contains the description of the things a car can do - for example "steer" and "direction". We can then create two (or more) separate "car" objects like this,

```
Make_Car Car_1;
Make_Car Car_2;
...
```

The cars of this class each will have class functions for steering and direction. So we can say in our code,

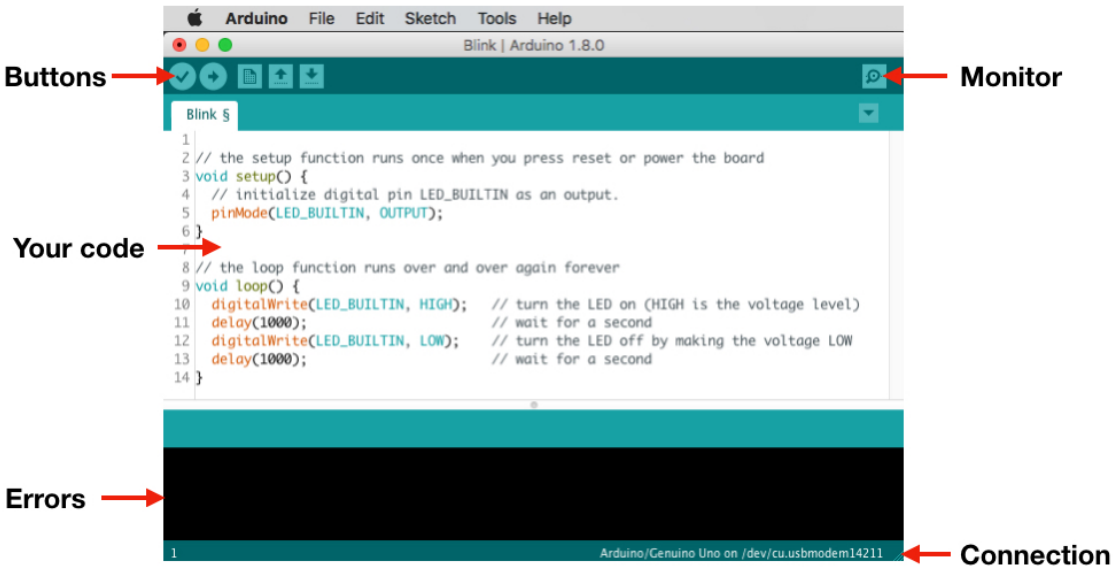
```
Car_1.direction(forward);

Car_2.steer(left);
```

etc

We shall see the use of libraries later on for reading a Rotary encoder and controlling a Digital Frequency Synthesiser.

Using the Arduino IDE
The window of the IDE looks like this (MacOS version),



The menus are at the top. Underneath are 5 buttons on the left and one on the right. This is what they do,

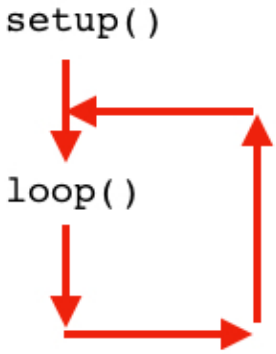
1. Compile your code and report any errors
2. Compile AND Upload to your Nano
3. Open a new sketch
4. Open an existing sketch
5. Save your sketch (they are also saved by Compile and Upload)

The right hand button opens a communication window that the Nano USB connection can use to exchange information to and from your PC.

What is a function
A function is a stand alone bit of code which can take an input and provide an output. It is stand alone. It looks like this

```
int functionName(int input, int input) {
  what the function does
}
```

Coding a sketch
All sketches follow the same pattern or flow chart. This is quite simple and is made up of two functions that must be present,



When a sketch starts to run it does so at the function "`setup()`" which runs just once, execution then passes to the function `loop()`, which runs repeatedly over and over again. It's that simple. It is best understood by opening a new sketch (File > New or the New icon). which has comments about the way the sketch works. Do it now.

IDE internal functions
The Arduino IDE provides a (hidden) library known as `Arduino.h`. This has a number of pre-defined functions to enable you to write code, add to this any functions you write yourself. A list of the Arduino functions is in the file Documents > Arduino > reference > Reference.pdf. Or you can find it on the web at arduino.cc > Resources > Reference. If you want to know how a function works go to the web and click on its name.

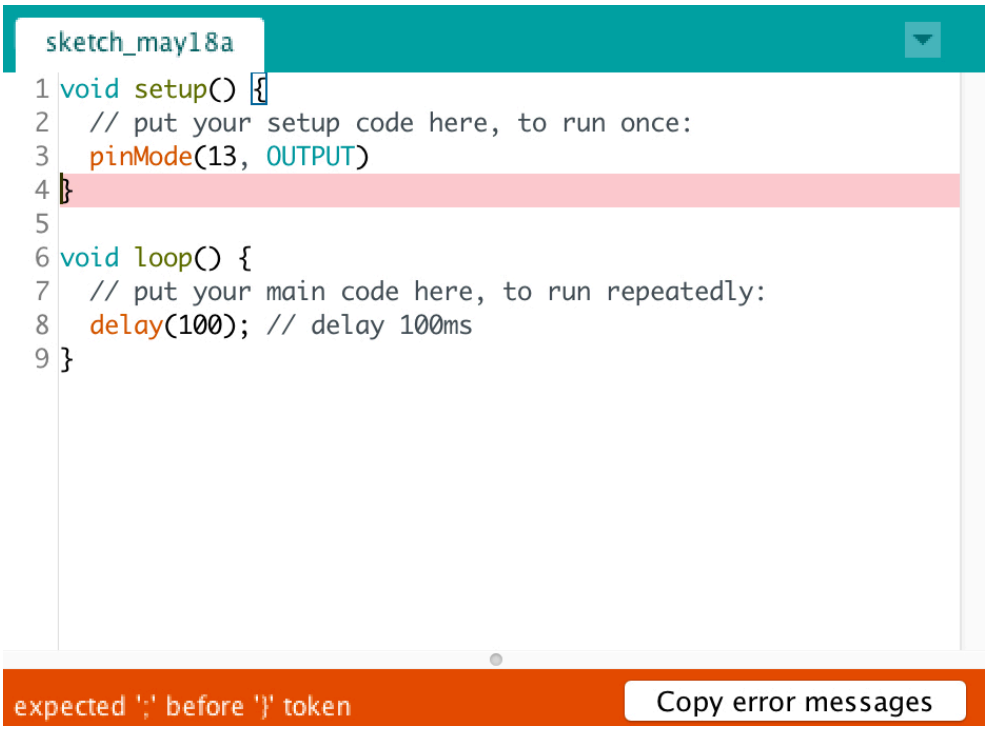
Write a little code
Open a new sketch and enter this code

```
void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT)
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  dekay(100);
}
```

There is a deliberate mistake here, a ";" is missing after the `pinMode(...)` in the code. Now we will check the code for errors, Hit the Compile icon and the error will be shown - when you hit Compile you will be asked to save the sketch, do so, it can be deleted later.

This is the error report



reporting that a "`;`" is expected before "`}`" token. Correct this (add a "`;`") and compile again to see that is now correct. All functions MUST be terminated by a "`;`", but they can spread across more than one line of code.

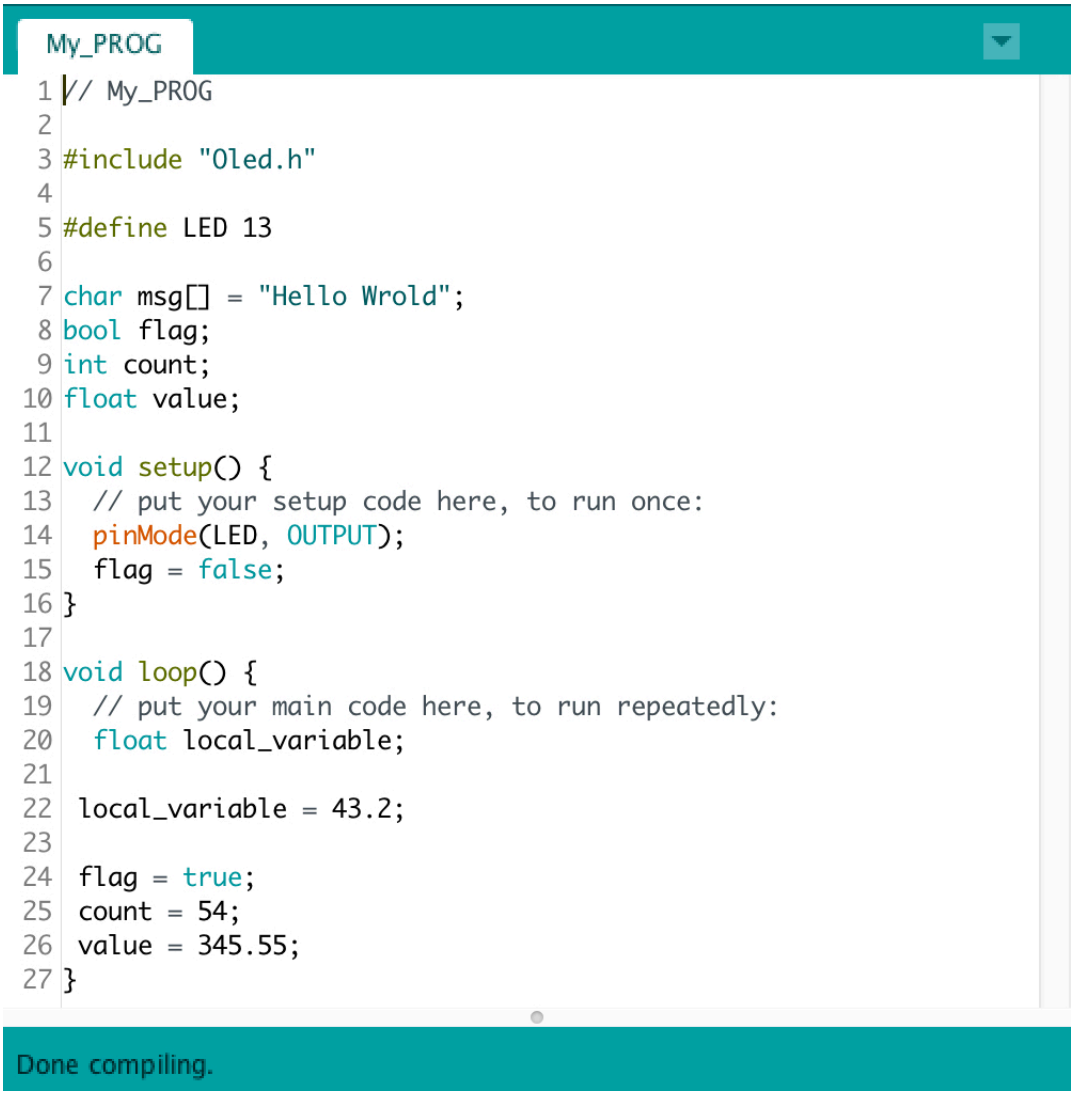
The next thing to learn is how to put comments in your code, so that if you come back to it later it is clear how it works. This is simple. A comment can start anywhere, on the same line as code, or a new line, it must start with "`//`" and ends when there is a new line. There is an example after the code "`delay(100);`" above. Comments are ignored by the compiler.

Global instructions and variables
When you want to include an external header or library in your sketch, or define a constant you can do this outside any function, normally at the top of your code. Same goes for global variables.

In any function of your own that you write, or in `setup()` or `loop()` the variables you use are valid only *within* that function and can't be accessed outside. You can pass inputs to and get an output from a function, but within the function any inputs are copied and will not be changed by the function.

But if you define a variable outside any function it becomes global and can be accessed anywhere in the sketch.

For example,



Here the compiler will include the header `"Oled.h"` in your code and substitute `"13"` for `"LED"` wherever it finds it. The global variables `(m[ ], flag,` etc) can be accessed anywhere in any function, for example in `"loop()"`, which also has its own `"local_variable"` not accessible outside.

Open the sketch `File > Sketchbook > My_BLINK.` Now create an error, say you remove a `";"` Compile the sketch and it will detect the error.

The end
If you have not understood this, then there is no solution, go back and study it again. Read the function operations from Reference on-line, and look at code in the IDE File > Examples. Try to write some of your own code and compile it. E.g. blink the LED twice at 100ms, followed by once at 1sec ...see `My_BLINK_1.`

In the next section you will learn about using the Nano and trying out simple programs.