Interfacing with Arduino UNO - part 2: output signals

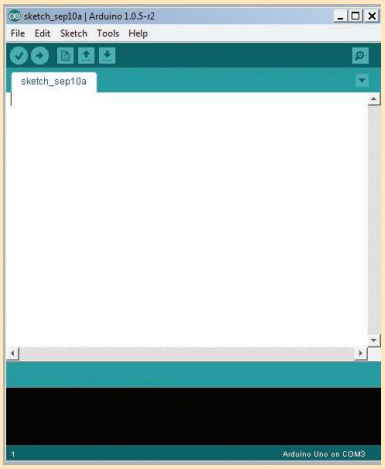

Figure 1 - IDE Screen.

In a previous issue (SSERC Bulletin (2014), 249, 6-7) we promised an article on interfacing; here it is...

Double click the Arduino icon, select FILE and then NEW. The blank IDE screen is displayed as shown in Figure 1.

A program is called a SKETCH.

Making comments within a program

At any position in a sketch, comments can be made. Comments are ignored by the microprocessor - they exist purely for human assistance.

The // comment

is limited to one line, for example:

```
//   This program flashes LED's
```

Notice, you don't require a second set of '//' at the end of the text. The new line ends the text!

You can program as many '//' comments as you like, for example:

```
//   this program flashes LEDs
//   Red for stop
//   Green for go
```

Note: Comments are removed during the 'compiling process' and hence never reach the Arduino board.

The /* Comment

This allows text to go beyond one line, e.g. long sentences or paragraphs. The /* indicates the start of the text, and in this case, the end of the text needs to be indicated, so */ is the end of text marker. So:

```
/*   start of text
*/   end of text
```

Typically, this command could be used to contain the sketch title and a brief description of what the sketch does.

The semi colon (;)

Because the programming language used in Arduino is based on the C Language, each line of program requires an end marker, in 'C' this marker is the semi colon.

Typical code:

```
delay(1000);
delayWrite(13, LOW);
```

REMEMBER: use the semi colon.

Program outline structure

C Programming involves 'Functions' - blocks of code. The structure of an Arduino sketch at the start always involves two functions and is shown in Figure 2. This is a sketch outline taken from the IDE display. The two functions are called 'setup' and 'loop' The system will run through 'setup' once only, while within the loop, it continually loops around the contents of the loop function as shown by the arrows in Figure 2. Setup is the ideal place to declare constants, values etc. (more later). After the setup function is used the program enters the loop function, where it is essentially trapped within a loop, over and over again. Note, the arrows on Figure 2 are there to demonstrate the program direction, they do not exist on the IDE.

This 'loop' function is stopped basically (at this stage) by disconnecting power from the Arduino.

Curly brackets

Notice the curly brackets (Figure 2), each bracket indicates the start or the end of each function.

REMEMBER: curly brackets in pairs.

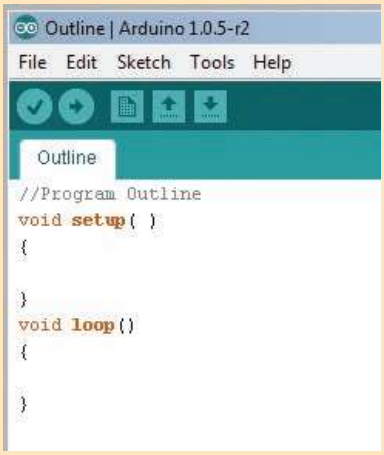

Figure 2 - Program outline.

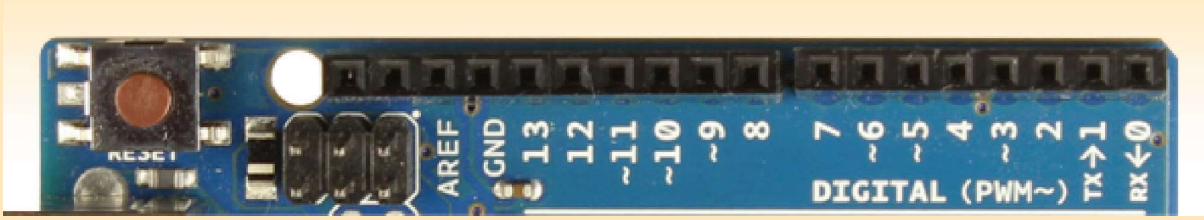

Figure 3 *- Digital connections on the Arduino UNO Board.*

Time delay

The Arduino system works with a millisecond clock. The command 'delay' gives a time delay, but the value of the delay must also be included with each 'delay' command,

Delay(value);
value must be a number.

For a 1 second delay
delay(1000);

For half a second delay
delay(500);

REMEMBER the semi-colon.

Outputting digital signals

Notice in Figure 3 that there are 14 output connectors (pins) on the Arduino UNO board.

(numbered 0 - 13). Before they can be used, they need to be 'set up'.

pinMode(pin, mode)

pinMode is the code for setting up these digital pins. 'pin' refers to the number of the pin(connector), pin is simply a number between 0 and 13. As these pins can be set to either input or output, the mode command indicates which.

So to set pin 13 for output:
pinMode(13,OUTPUT);

To set pin 5 for input:
pinMode(5,INPUT);

The ideal place for these commands would be within the 'set up( )' function.

REMEMBER: you must set the pins up before any attempt to use them!

REMEMBER the semi-colon.

Using the Pins

Assuming that the pins are set up using the pinMode(pin, mode) command(s) output signals can now be generated from within the loop() function.

The code to trigger an output signal is:
digitalWrite(pin, value)

In order to output a signal, the pin number must be known and whether that pin is to be switched HIGH or LOW(value), where HIGH is 5 V, LOW is 0 V.

Typical:
digitalWrite(13, HIGH);
//this would turn pin 13 On (5 V)

digitalWrite(13, LOW);
//this would turn pin 13 Off (0 V)

TASK 1

Using a blank IDE screen (Figure 1), type an outline sketch structure as shown in Figure 2. Use FILE and SAVE this sketch outline as 'Outline' When the sketch is saved, the name 'Outline' is displayed just above the sketch code. This sketch 'Outline' saves retyping 'setup() and loop() each time. This gives a template sketch for further use.

TASK 2

Using digital pin 10, flash a LED connected to this pin, on for 0.5 seconds, off for 0.5 seconds. This flashing is to be repeated continuously. At this stage write the sketch and save the sketch as 'task 2'.

Notice in Figure 4, within setup() is the setting of pin 10 to a digital output signal. Remember, pins must be 'setup' before using the 'pinMode' command.

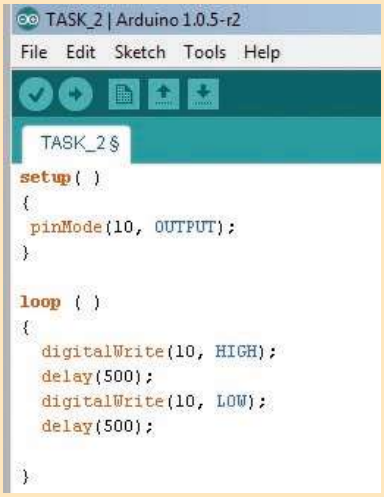

Figure 4 *- TASK 2 Sketch Code.*

Within the loop function (Figure 4), pin 10 is turned on via 'digitalWrite(10, HIGH);

Then a 0.5 second delay...
delay(500);

Then switched off...
digitalWrite(10, LOW);

Then, off for 0.5 second using...
delay(500);

Note: this second delay within the loop is often missed out when developing the code, without this delay, due to the loop the 'LOW' signal is immediately followed by the start of the next loop and digitalWrite(5, HIGH) swamps the 'LOW' - the LED will never appear to be LOW!

Give the 'LOW' a chance...
include its own delay(500);

Hard wiring an LED to the Arduino UNO Board

Figure 5 shows the connection of an LED. This circuit uses Pin10 switching on or off via a sketch (Figure 4). The circuit is completed via the Arduino GND (ground) connection. When Pin 10 is switched on, 5 V can be measured between GND and Pin 10. This would quickly 'blow' the LED and so a resistor needs to be inserted into the circuit in order to reduce the voltage across the LED. The red wire can of course be connected to any of the digital output pins.

What size of resistor? is a common question. The resistor value depends upon which type of LED is used. Typically a white LED requires 3.2 V voltage drop across the LED, 25 mA. LED current. With Arduino there is a 5 V drop between 'GND' and any 'on' pin.

Websites exist which will calculate resistor values in LED circuits, for example, http://ledcalc.com/ For the values above, the resistor value is 72 ohms. The website also give the nearest actual resistor value, in this case 82 ohms. The supply voltage for Arduino is 5 V, the voltage drop across the LED and the LED current need to be provided from data sheets for the LED that are being used.

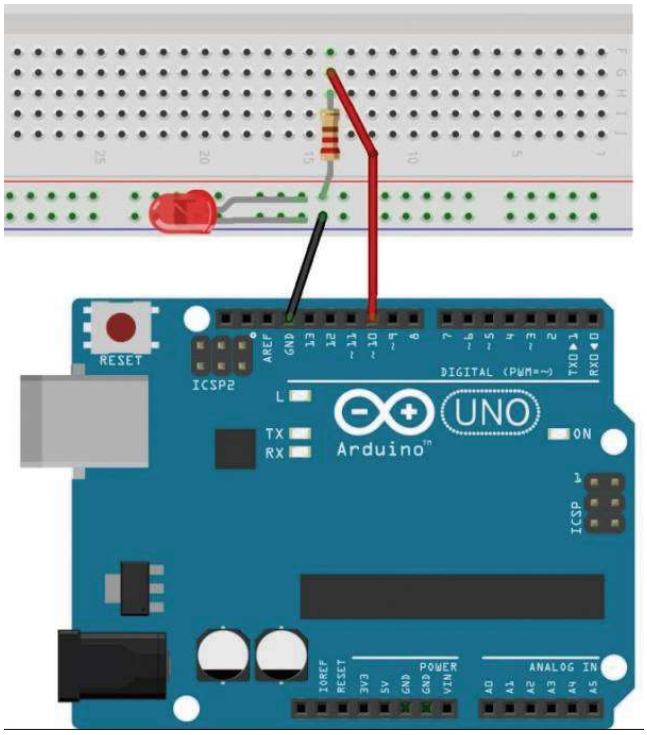

Figure 5 *- Connecting an LED to Arduino.*

Calculation of resistor value for an LED circuit

Arduino supply voltage = 5 V, typically LED voltage drop 3.2 V, LED current = 25 mA = 0.025 A

$$\text{Resistor Value} = \frac{\text{Supply Voltage - LED voltage drop}}{\text{LED current (I)}} = \frac{(5 - 3.2)}{0.025} = 72 \text{ ohms}$$

TASK 3

The Northern Lighthouse Board has a website www.nlb.org.uk. Investigate the Butt of Lewis lighthouse and determine the 'character' of this lighthouse.

'Character' is the pattern of light with corresponding timings. Develop, using Arduino a continuous flashing circuit which mimics the Butt of Lewis lighthouse. Use as a digital output, pin 1.

(Answer: Butt of Lewis Lighthouse: flashing white light every 5 seconds).

TASK 4

Write code for a set of traffic lights, build using three LEDs a working model.

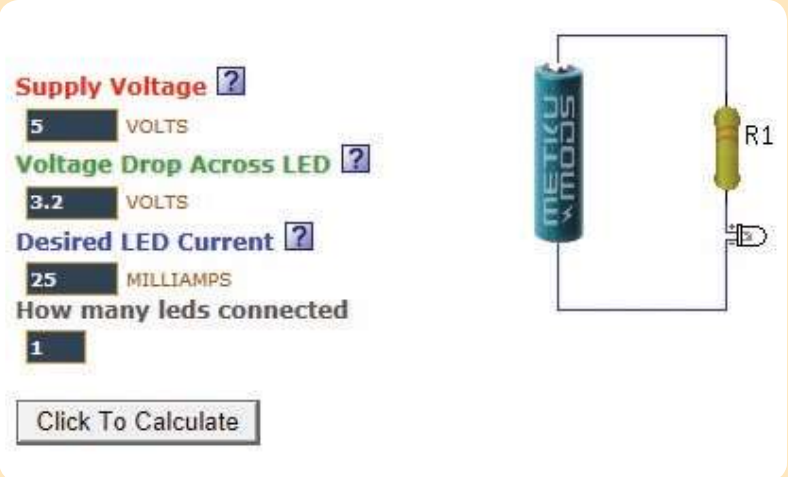

Figure 6 *- 'ledcalc.com' Resistor Calculator.*