Comp 1421 Foundations of Computing

Workshop1 Introduction to the Arduino

CBPrice September 2021

Color Coding for this Worksheet	
	Information
	Instructed Learning
	Autonomous Learning
	Assignment Information (when applicable)

Arduino Code	
Digital Input and Output	
Set a digital pin (number X) to be an input	**pinMode(X,INPUT);**
Set a digital pin (number X) to be an output	**pinMode(X,OUTPUT);**
Write a low value to digital pin X	**digitalWrite(X,LOW);**
Write a high value to digital pin X	**digitalWrite(X,HIGH);**
Read the input from a pin X (when it is configured INPUT)	**int val = digitalRead(X);**
Delay Function calls	
Delay for M milliseconds	**delay(M);**
Delay for M microseconds	**delayMicroseconds(M);**
Using the Serial Monitor	
Setup the Serial Monitor	**Serial.begin(9600);**
Write the value of a variable to the Monitor	**Serial.print(val);**
Write the value of a variable to the Monitor with LF	**Serial.println(val);**

Activities

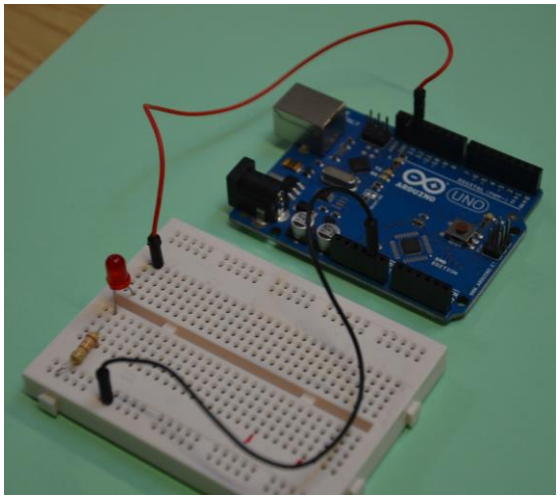
1 **The Arduino Software**

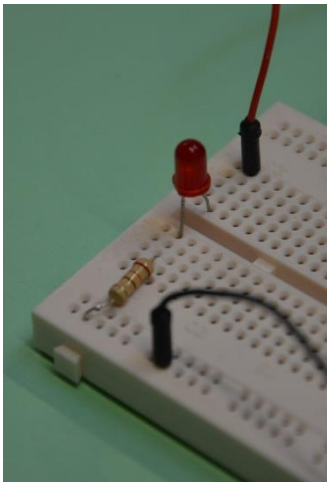
Details of the function calls you will use are provided in the above table.

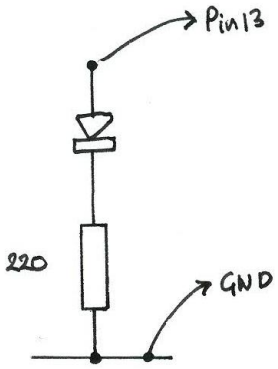
2 **Driving an Output (1) A single LED**

(a) Download the prepared Sketchbook from the module web-pages and unzip into a location of your choice. Select File > Preferences and Browse for your Sketchbook folder. **DO NOT connect your Arduino to the PC.**

(b) Now we need to construct our first electronic circuit, a single LED with its current-limiting resistor. The circuit and photographs of the setup are shown below. Remember the instructions you were given, it's important that your circuit construction looks exactly like this! **NOTE: The longer leg on the LED must be connected to the red wire.**







Ask Dr.C to check your circuit before you connect the Arduino to the PC. This is important. An incorrect circuit may damage the Arduino, or even cause a fire to start!

(c) After connecting your Arduino, find Tools > Port and select the Arduino

(d) Now let's write some code to drive the light on and off. Open up the sketch EOCS_Blinky1 which contains the code template. Note the declaration and assignment of the variable **pinLed1** which identifies the pin you shall be driving.

(e) Add code to **setup()** to set **pinLed1** to an OUTPUT pin.

(f) Add code to **loop()** to write a value HIGH to the **pinLed1**

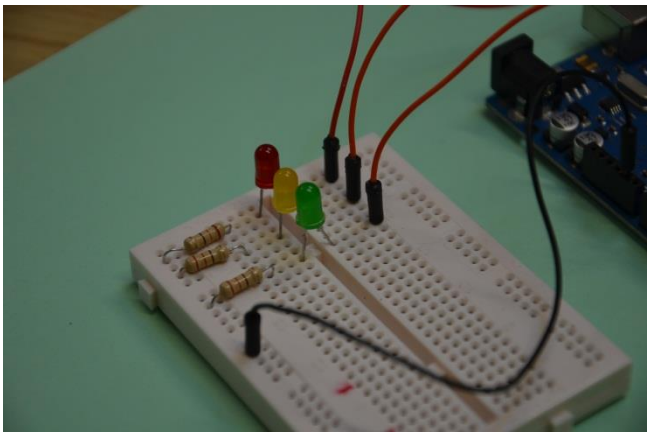
(g) Now add code to (i) make the pin stay on for 0.5 sec (500ms) and then (ii) write a value LOW to **pinLed1** then (iii) make the pin stay off for 0.5 sec.

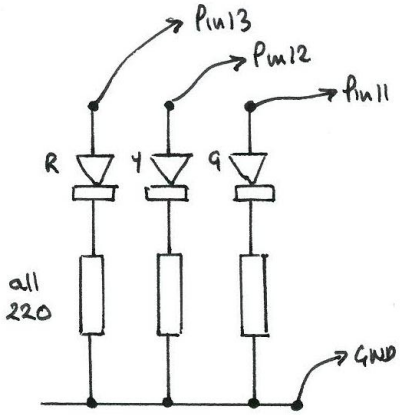
(h) Compile your code (and debug if necessary). Now download your code and the LED should flash on and off with a period of 1 second. **Save your sketch.**

(i) Now reduce both delays to 10ms (0.01) of a second. What do you see? Can you explain? Try an even smaller delay.

3 Driving Multiple Outputs (Traffic Lights)

(a) Add two extra LEDs and two extra current-limiting resistors (220 Ohms) and build the circuit shown below





Now let's write some code to drive the LEDs as a traffic-light sequence (red, red-and-amber, green, amber, red).

(b) Open up the template EOCS_TrafficLight1 where you can see some variables have been declared and one is assigned (**RED_LED)**.

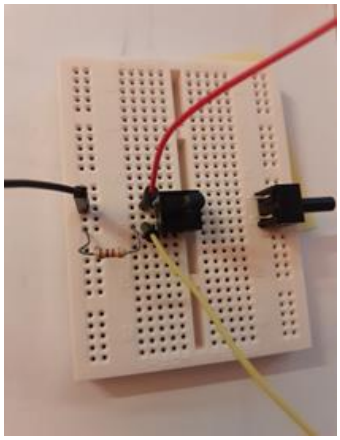
(b) Assign values to **AMBER_LED** and **GREEN_LED** so they are assigned respectively to pins 12 and 11. Then set these two pins to OUTPUT

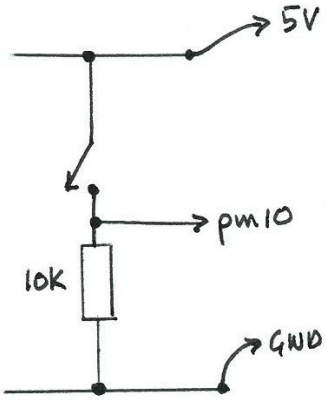
(c) In the **loop()** code block write appropriate values to **RED_LED, AMBER_LED, GREEN_LED** to output the "red" state of the traffic light. Then add a line of code to hold this state for 2 seconds.

(d) Now repeat (c) for the "red-and-amber", "green", "amber" and red states. Think! You may choose how long the traffic light remains in each state to make a believable traffic light.

4. Getting input from a switch

Let's explore how to read the input from a switch. The circuit below behaves as follows:
* When the switch is open, it sends a LOW value to the input pin
* When the switch is closed (pressed) it sends a HIGH value to the input pin.





(a) Build the circuit as shown above on the same breadboard as your traffic-light solution. **Keep your traffic light solution intact.** The extra button on the right shows you the pin orientation.

Now to the coding.

(c) Open up the sketch EOCS_Input1 where you will see the variable **pinSw1** declared which you will assign to the input pin. Let's agree to use pin 10 for this.

(d) In **setup()** assign the value of **pinSw1** and set the pinMode for this pin to INPUT.

(e) Now assign the value of **pinLed1** to pin 13 and set this pin to be an OUTPUT.

(f) Declare an int variable of your choice (at the correct place in the code) to which you will assign the value of the pin when it is read. I shall choose **inVal.**

(g) In the **loop()** function make the correct function call to read the input **pinSw1** value and assign it to your variable. Consult the software overview sheet if you are uncertain.

(h) Now write some code to test the value of the variable (HIGH or LOW) and to illuminate the LED if the switch is pressed, else turn it off. You will need a selection statement which looks like this

```
if(val = = HIGH) {
        … some code goes here maybe
} else {
        … some code goes here maybe
}
```

(i) Compile, upload your code and run it.

(j) **Save your sketch.**

5. Now we are going to integrate your traffic light solution and your input solution. The idea is that the traffic lights are green and **stay green until the switch is pressed**. Then they cycle through the appropriate states to "red" and they stay "red" for a certain time after which they cycle to the state "green".
There is no additional circuitry to build, but there's quite a lot of code. You will also need to copy in your code from the switch example. Don't forget declarations and assignments.