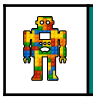
Section A
Unit #4
Boolean

Contents

Section A Unit #4 Boolean

Introduction to Unit #4 Boolean

Boolean is a form of logic which can be described as gates, the most common are AND gates, and OR gates. There are more of them and they form the basics of computing. They are also used widely in higher level programming. Two symbols are used `&&` for AND, and `||` for OR (called pipes). Trying to explain makes it seem quite confusing and yet it is literally the most simple logic. The best way is to use it and you will see how simple and powerful it is. The `if()` statement can be similar to the `while()` loop, if something is true or a condition is met then do something. To put it into words if (`x < 100 && y > 50`) means if x is less than 100 and y is greater than 50.



Sketch A4.1 starting to fade (part 1)

Going back to built in LED we can fade the LED using analogWrite() function. The pins all support PWR (I won't go into the details here just yet but it won't work with 13 and 12 but it will on pin 11) but it means we can assign a value to the pin, so if we give pin 11 a value of 255, that means fully on (100%), if we assign 0 (0%) then it is totally off. Any numbers in between those two values will give a brightness between the two. The following sketch will demonstrate this. First full brightness = 255

```
void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  analogWrite(11, 255);
}
```

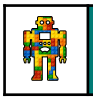
Code Explanation:
`analogWrite(11, 255);` This writes a value (255 in this instance) to pin 11. Although it is a digital pin it is treating it as an analog output.

starting to fade (part 2)

Now give it the minimum value = 0. This should be completely off.

```
void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  analogWrite(11, 0);
}
```

Code Explanation:
`analogWrite(11, 0);` This writes a value (0 in this instance) to pin 11. Although it is a digital pin it is treating it as an analog input.



starting to fade (part 3)

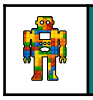
This is somewhere in between the two extremes, 25. It is a bit hard to see the difference so we will do an incremental fade in the following sketches.

```
void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  analogWrite(11, 25);
}
```

starting to fade (part 4)

The delay is small and it is the amount of time (in milliseconds) that it shines for that intensity of brightness.

```
void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  analogWrite(11, 25);
  delay(10);
}
```



starting to fade (part 5)

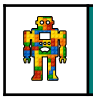
Instead of a fixed value we want to increment it by one every 10 milliseconds.

```
void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  for (int i = 0; i < 256; i++)
  {
    analogWrite(11, i);
    delay(10);
  }
}
```

Code Explanation:
`analogWrite(11, i);` This increments to brightness from 0 through to 255 incrementing the value of i by 1 each iteration of the loop

Notes

You should see it start off dim and gradually getting brighter until maximum brightness and then restart all over again.



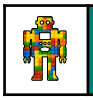
starting to fade (part 6)

Now to fade back down again. Notice the `i--` at the end of the decreasing fade. What you should have now is the LED getting rapidly brighter and then quickly fade and then brighter repeatedly.

```
void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  for (int i = 0; i < 256; i++)
  {
    analogWrite(11, i);
    delay(5);
  }
  for (int i = 255; i > 0; i--)
  {
    analogWrite(11, i);
    delay(5);
  }
}
```

Notes

There is quite a bit of unnecessary repetition. Can you think of a way to combine the two? Don't worry if you can't, just have a thought.



Sketch A4.2 boolean operator (part 1)

Start newish sketch with the basics below. Another way to do this is to use a boolean operator with an `if()` statement. So start with a basic sketch putting the LED on max brightness (255).

```
void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  analogWrite(11, 255);
}
```

boolean operator (part 2)

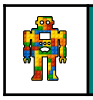
Now adding a variable for brightness. Nothing too dramatic here.

```
int brightness = 255;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  analogWrite(11, brightness);
}
```

Boolean Operators

This is all to do with logic, 'and', 'or', 'not'

Code Explanation:
! logical 'not', i.e. a condition where something is not true can be written != (not equal to)
&& logical 'and', i.e. a condition where two things have to be true
|| Logical 'or', i.e. a condition where either one of two conditions is true



boolean operator (part 3)

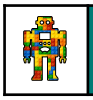
We want to fade it up and down. So the initial brightness we will change to 0 so that when it reaches 255 the brightness decreases and when it is 0 it increases. We are using a boolean operator that are called pipes `||` which means one OR the other has to be true. The `fadeValue` is an integer variable that is how much it will fade by in steps of 5 in this case.

```
int brightness = 0;
int fadeValue = 5;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  analogWrite(11, brightness);
  brightness = brightness + fadeValue;
  if (brightness == 0 || brightness == 255)
  {
    fadeValue = -fadeValue;
  }
  delay(10);
}
```

Code Explanation:
`if (brightness == 0 || brightness == 255)` If brightness is equal to 0 **or** 255 then change fadeValue from positive to negative or vice versa

Notes

It toggles the value of fadeValue between 5 and -5 depending if it starting from zero or 255, just follow the logic



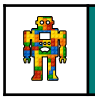
Sketch A4.3 without delay (part 1)

Sometimes it is critical that we don't use the function delay() simply because the whole programme (the Arduino) stops and you might want other things to happen while there is a delay. There is a way of doing the blink sketch without using the delay function. Starting with a very basic sketch and returning to pin 13 the red LED

```
void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
}
```

Notes

The red LED should be on bright (HIGH)



without delay (part 2)

Instead of using HIGH and LOW for on and off we will give these a variable name so that we can change the state of each. We will call this variable ledState and set it to LOW initially

```
int ledState = LOW;


void setup()
{
  pinMode(11, OUTPUT);
}


void loop()
{
  digitalWrite(11, ledState);
}
```

Notes

The red LED should be off completely (LOW)

Challenge

You can check it is working by changing the ledState to HIGH and uploading it again.

without delay (part 3)

We want the LED to blink on/off for one second intervals so we have a variable called interval and we will make it a constant (ie fixed) value of 1000. Const is short for constant and means it can never be changed accidentally.

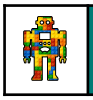
```
int ledState = LOW;
const int interval = 1000;


void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, ledState);
}
```

Code Explanation:
`const int interval = 1000;`
`const` is short for constant and cannot be changed

Data types

These are the types of data you may well come across and need to use, a brief description is given

Code Explanation:
`const` Short for constant eg pin number
`int` Short for integer eg 3
`float` Has a decimal point eg 3.76
`long` Can use much bigger numbers
`unsigned` Can only be a positive number
`char` A type of string (text)



without delay (part 4)

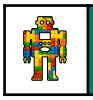
The Arduino starts counting milliseconds as soon as the sketch starts running. We can use that to check how much time has elapsed since a particular moment in time. So we need to get the number of milliseconds that have elapsed since a particular point in time.

This number is going to get very big very quickly also we don't want it to become negative for any reason. So the data type we use is called 'long' and we can make sure it is always positive by defining it as unsigned.

Our variable will be called currentMillis.

```
int ledState = LOW;
const int interval = 1000;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  unsigned long currentMillis = millis();
  digitalWrite(11, ledState);
}
```

Code Explanation:
`millis();` It returns the number of milliseconds since the programme was started, a bit like a running clock



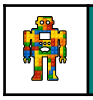
without delay (part 5)

We also need another similar variable to hold the previous number of milliseconds. We will call this `previousMillis()`. As the sketch runs we will subtract the current from the millis and compare it to the interval with an `if()` statement.

```
int ledState = LOW;
const int interval = 1000;
unsigned long previousMillis = 0;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  unsigned long currentMillis = millis();
  digitalWrite(11, ledState);
}
```

Notes

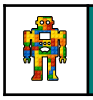
We have two variables to compare previousMillis and currentMillis, we can do maths with them to work out any delay, it is like a stop watch with a lap function.



without delay (part 6)

We need an `if()` statement to check the difference and reset the `previousMillis()`

```
int ledState = LOW;
const int interval = 1000;
unsigned long previousMillis = 0;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval)
  {
    previousMillis = currentMillis;
  }
  digitalWrite(11, ledState);
}
```



without delay (part 7)

And now to change the state depending if it is LOW to become HIGH or HIGH to become LOW. It toggles between them. That is why we need to keep a track of the state of the LED, if it is off then we want it on, if on then off.

```
int ledState = LOW;
const int interval = 1000;
unsigned long previousMillis = 0;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval)
  {
    previousMillis = currentMillis;
    if (ledState == LOW)
    {
      ledState = HIGH;
    }
    else
    {
      ledState = LOW;
    }
  }
  digitalWrite(11, ledState);
}
```

Code Explanation:
`if ... else` Another way of using an if statement where there is more than one condition. In words: if this isn't true then do something else.

Notes

Your led should blink every second

Challenge

Change the interval to a much smaller number e.g. 100, it should blink much faster