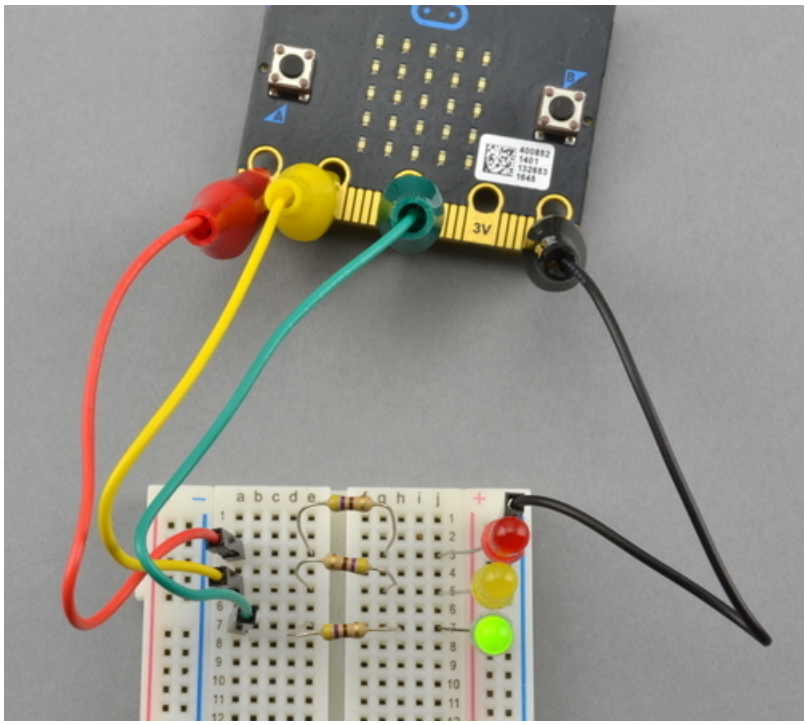
adafruit learning system

micro:bit Lesson 2. Controlling LEDs on Breadboard

Created by Simon Monk

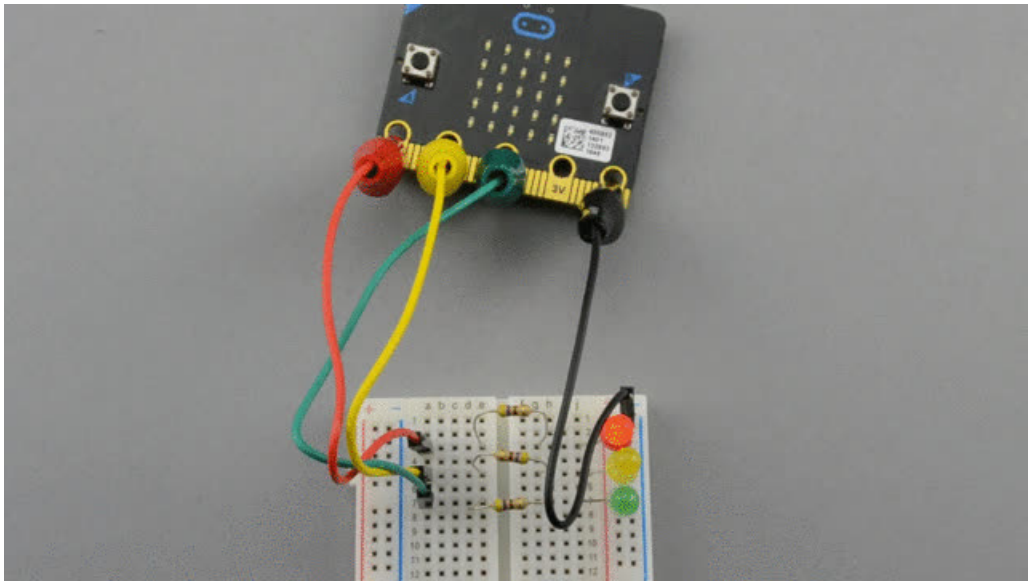


Last updated on 2018-08-22 04:05:10 PM UTC

Guide Contents

Overview

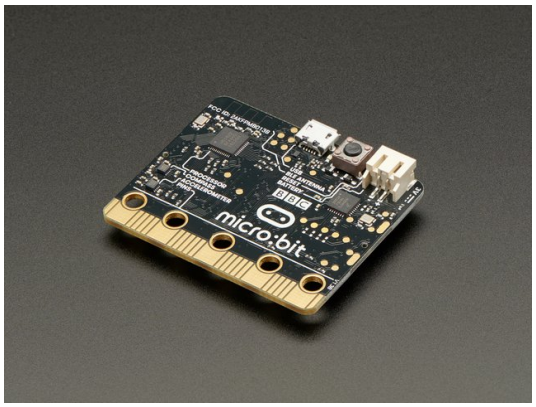
In this lesson, you will learn how to connect a micro:bit to solderless breadboard and how to turn an LED on and off, control its brightness and also use three LEDs to make a traffic signal.



You will start off just making a single LED blink, then control the brightness of the LED and finally add two more LEDs to make the traffic signal shown above.

Parts

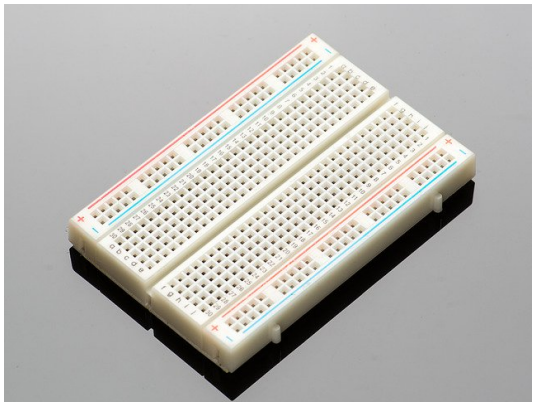
To follow all the activities in this lesson, you will need the following parts. You will also need a micro USB lead to connect your micro:bit to your computer.



BBC micro:bit

$14.95
OUT OF STOCK

OUT OF STOCK



Half-size breadboard

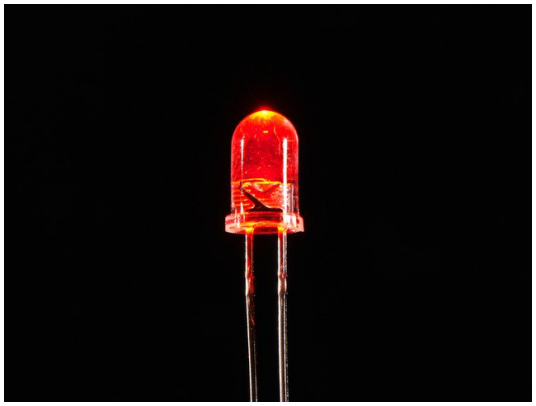
$5.00
OUT OF STOCK

OUT OF STOCK

Small Alligator Clip to Male Jumper Wire Bundle - 12 Pieces

$7.95
IN STOCK

ADD TO CART



Super Bright Red 5mm LED (25 pack)

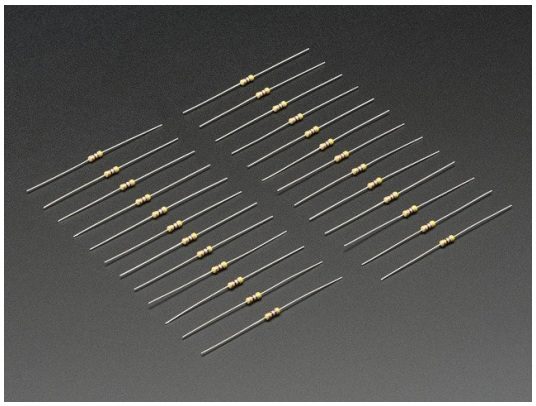
$8.00
IN STOCK
ADD TO CART

Super Bright Yellow 5mm LED (25 pack)

$4.95
IN STOCK
ADD TO CART

Super Bright Green 5mm LED (25 pack)

$8.00
IN STOCK
ADD TO CART

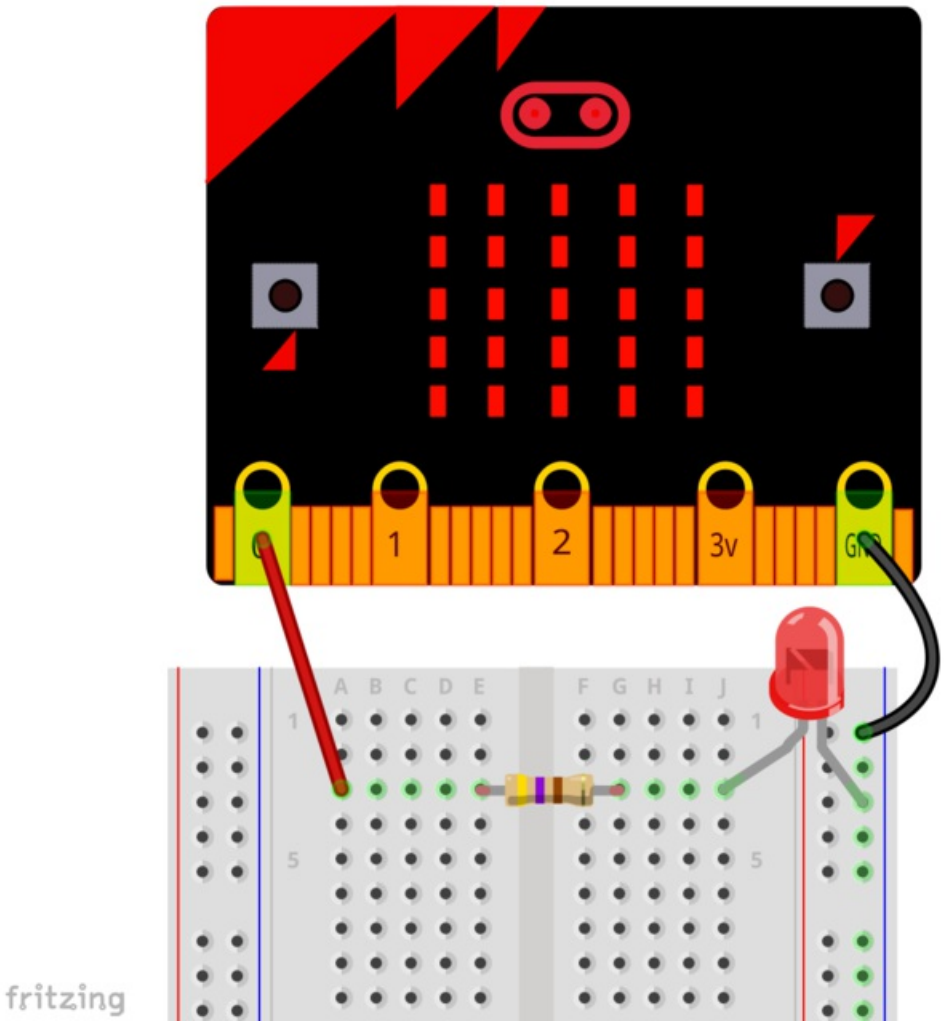


Through-Hole Resistors - 470 ohm 5% 1/4W - Pack of 25

$0.75
IN STOCK
ADD TO CART

Breadboard Layout

Connect the resistor and LED as shown in the diagram below. Make sure that the longer lead of the LED (the positive lead) is to the left. It doesn't matter which way around the resistor goes.



When connecting the micro:bit to the breadboard, its best to put the alligator clips vertically into the holes on the micro:bit's pads, otherwise the clips can slip off or make accidental connections with neighbouring connectors on the micro:bit.

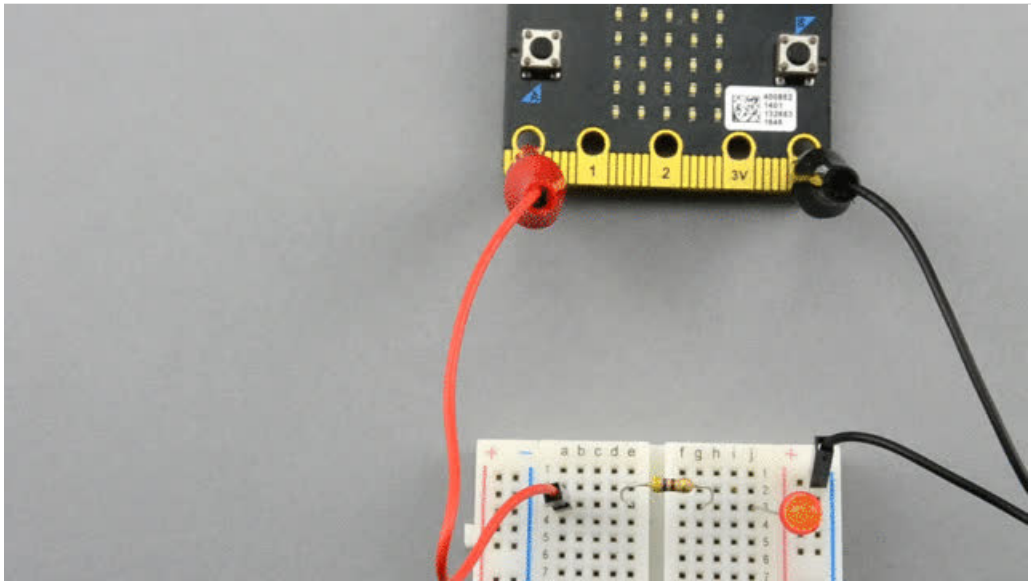
The way that solderless breadboard works is that behind the holes into which you poke component leads you will find a metal clip.

The red lead that connects pin0 on your micro:bit to row 3 of the breadboard actually connects the lead to every hole position on that row (at least for the left-hand bank of rows of five). The right-hand side of the resistor then connects to the clip underneath row 3 on the right-hand bank and hence to the positive lead of the LED.

The long columns down the sides of the breadboard work differently from the main rows of connections in the central area of the breadboard. Each of these long columns are bade up of a single long clip that is often used to provide power. In this case the right hand column (with a blue line next to it) is connected to the micro:bit's GND connection and also to the negative side of the LED.

The resistor is necessary to limit the current flowing through the LED. A digital output from a micro:bit should not be allowed to draw more than 3mA of current. The 470Ω resistor limits the LED current to under the 3mA limit.

Blinking an LED



Lets start by making our LED blink on and off. The way to make an LED blink is to follow this sequence of code instructions over and over again:

- Turn the LED on
- Delay for a period of time
- Turn the LED off
- Delay for a period of time

The code for this is different depending on which programming platform you have chosen to use.

JavaScript Block Code

The JavaScript Blocks Code editor is embedded directly on this page below. From the editor, you can click on Download button (bottom right) and then copy the downloaded file onto your micro:bit. Alternatively, you can Click here (https://adafru.it/CiI) to open the editor in a separate browser tab.

To install the program, copy the hex file onto your micro:bit. See this guide (https://adafru.it/CiJ) for getting started with Javascript Blocks.

The *Forever* block will repeatedly run all the code contained in the block.

MicroPython

To run the MicroPython version of the code, open up the online Python editor here (https://adafru.it/CiK) and paste the following code into the editor window.

```
from microbit import *

while True:
    pin0.write_digital(1)  # turn pin0 (and the LED) on
    sleep(500)             # delay for half a second (500 milliseconds)
    pin0.write_digital(0)  # turn pin0 (and the LED) off
    sleep(500)             # delay for half a second
```

The program first imports the *microbit* library that contains the *write_digital* function needed to control the pins on and off.

The *while* loop will repeat the commands it contains until you unplug your micro:bit. These commands first turn the pin on, delay for half a second, turn it off again and then delay again.

Arduino

Make sure that you have your Arduino environment set up for micro:bit by following this guide (https://adafru.it/CiL).

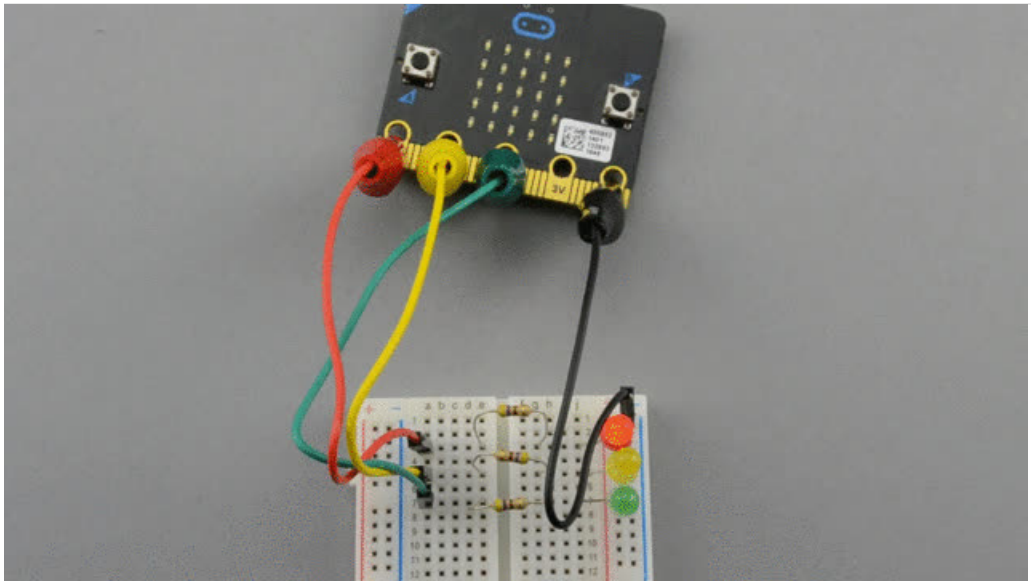
Now start a new Sketch by clicking on the *File* menu and *New*. Then paste the following code into the editor window.

```
// define a constant for the LED pin
const int ledPin = 0;

// setup is run just once when the micro:bit starts up
void setup() {
  pinMode(ledPin, OUTPUT);    // set the ledPin (pin0) to be an output
}

void loop() {
  digitalWrite(ledPin, HIGH); // turn the ledPin (and LED) on
  delay(500);                 // delay for half a second (500 milliseconds)
  digitalWrite(ledPin, LOW);  // turn the ledPin (and LED) off
  delay(500);                 // delay for half a second
}
```

Save the file and then upload it onto your micro:bit.

The Arduino version of the code is slightly different from MicroPython and the JavaScript Blocks code, because when using an Arduino you have to specify that the pin is to act as an output, whereas for the other languages this happens automatically the first time you use the pin as an output.

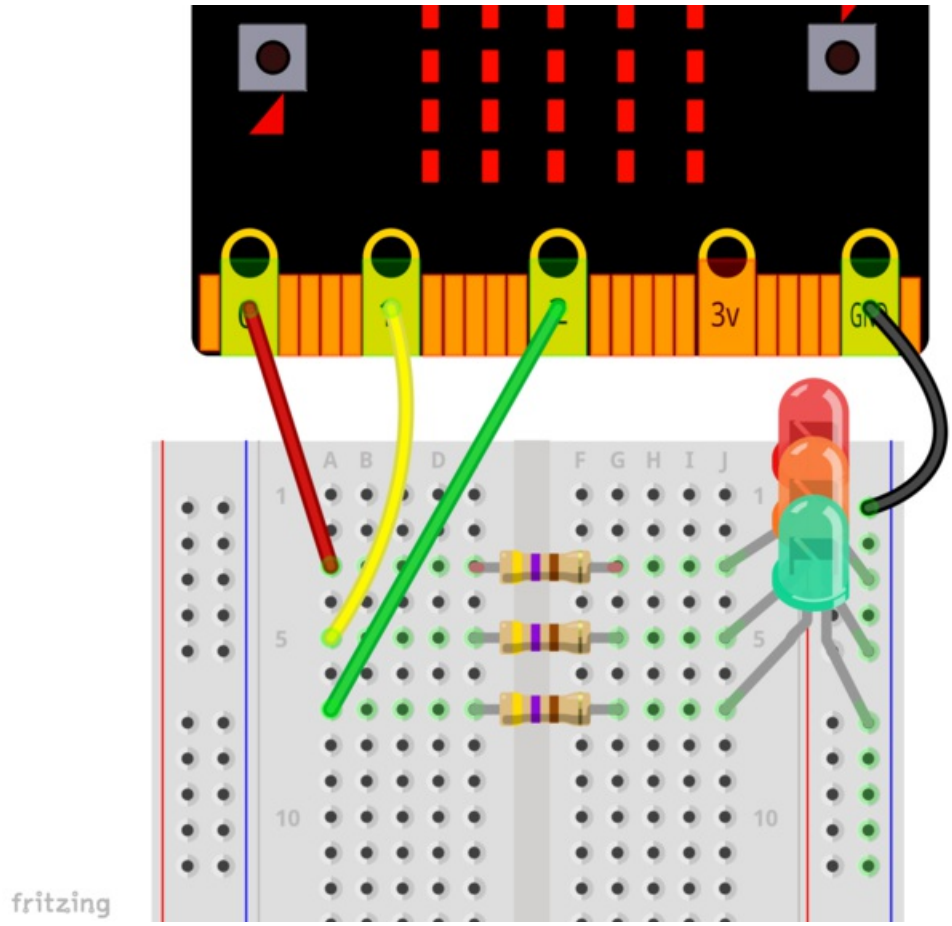
Traffic Signal



This example uses three LEDs to make a model traffic light signal. You can use all red LEDs if you like, but its more realistic if you use red, yellow and green.

The LEDs cycle around in the sequence red, yellow, green, yellow and then back to red again.

To add the extra LEDs and resistors, wire them up as shown in the breadboard layout below.



Take care to ensure that the LEDs are the right way around, with the longer positive leads (anodes) to the left.

JavaScript Blocks Code

The JavaScript Blocks Code editor is embedded directly on this page below. From the editor, you can click on Download button (bottom right) and then copy the downloaded file onto your micro:bit. Alternatively, you can Click here (https://adafru.it/CiM) to open the editor in a separate browser tab.

MicroPython Code

The MicroPython code is listed below.

```
from microbit import *

red_pin = pin0    # giving the LED pins names by using variables
amber_pin = pin1  # makes it easier to see how the program works
green_pin = pin2

while True:
    # red - turn amber LED off and red LED on
    amber_pin.write_digital(0)
    red_pin.write_digital(1)
    sleep(4000) # delay 4 seconds
    # amber - turn red LED off and amber LED on
    red_pin.write_digital(0)
    amber_pin.write_digital(1)
    sleep(1000)
    # green - turn amber LED off and green LED on
    amber_pin.write_digital(0)
    green_pin.write_digital(1)
    sleep(4000)
    # amber - turn green LED off and amber LED on
    green_pin.write_digital(0)
    amber_pin.write_digital(1)
    sleep(1000)
```

Arduino code

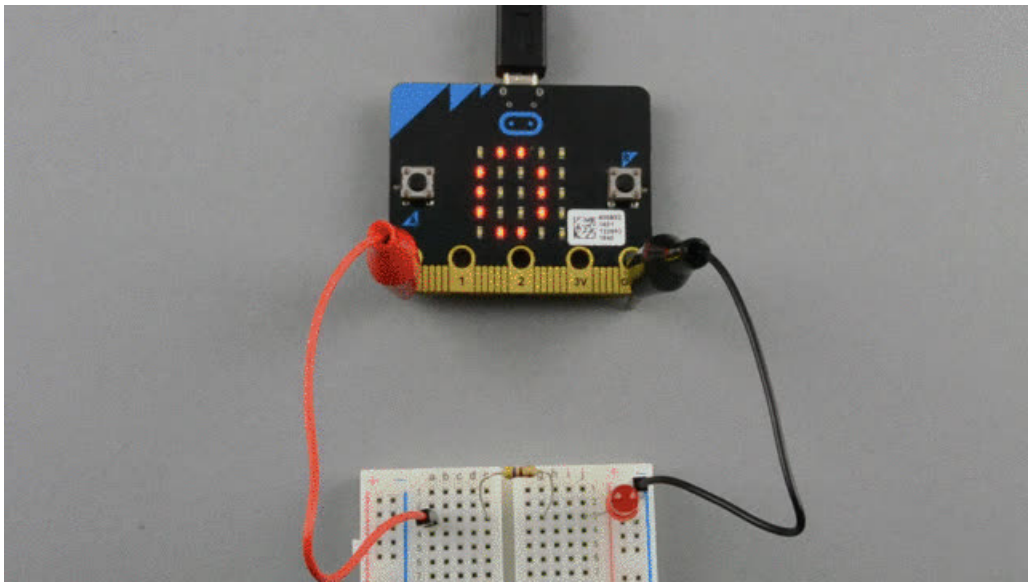
The Arduino version of this program is very similar to the other two versions.

```
// define constants for each LED pin
const int redPin = 0;
const int amberPin = 1;
const int greenPin = 2;

void setup() {
  // set all three pins to act as digital outputs
  pinMode(redPin, OUTPUT);
  pinMode(amberPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
}

void loop() {
    // red - turn amber LED off and red LED on
    digitalWrite(amberPin, LOW);
    digitalWrite(redPin, HIGH);
    delay(4000);   // delay 4 seconds
    // amber - turn red LED off and amber LED on
    digitalWrite(redPin, LOW);
    digitalWrite(amberPin, HIGH);
    delay(1000);
    // green - turn amber LED off and green LED on
    digitalWrite(amberPin, LOW);
    digitalWrite(greenPin, HIGH);
    delay(4000);
    // amber - turn green LED off and amber LED on
    digitalWrite(greenPin, LOW);
    digitalWrite(amberPin, HIGH);
    delay(1000);
}
```

LED Brightness Control

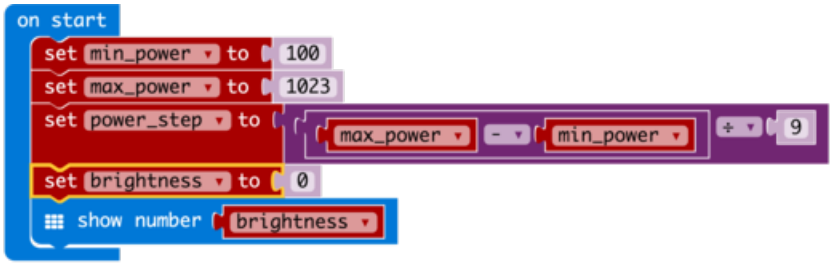
As well as turning an LED on and off, you can also use it to control the brightness of the LED.

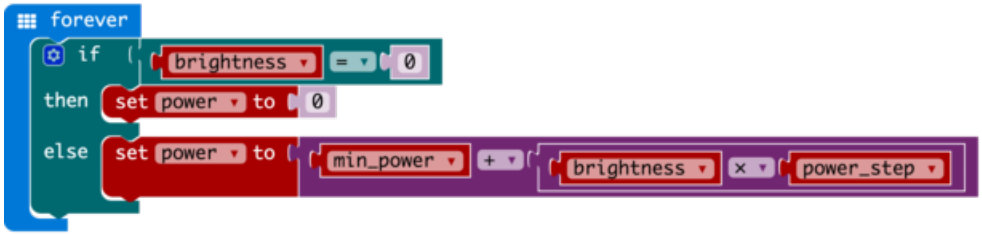
In this example, the A button will make the LED dimmer and the B button make it brighter. At the same time, the micro:bits built-in LED display will show a number between 0 and 9 indicating the brightness level.

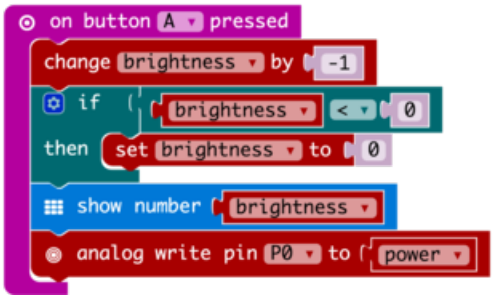


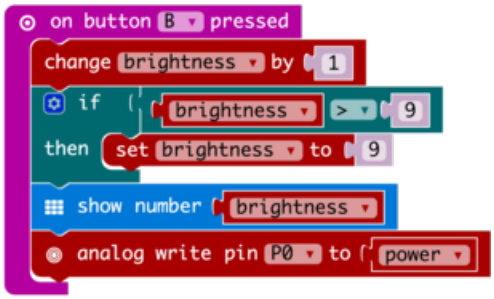
JavaScript Block Code

To open the JavaScript Block code in a separate tab, click here (https://adafru.it/CiN).









As you can see, there is actually quite a lot going on here. Lets start with the *on start* block. This block is run just once when the micro:bit starts and it defines four variables:

- min_power - the minimum output level for the LED when 0 is off and 1023 is maximum brightness
- max_power - the maximum output level for the LED. These two variable allow you to set the possible range of brightnesses.
- power_step - the brighness will be changed in 10 steps and so this value is calculated from the minumum and maximum.
- brightness - the brightness level as a number between 0 and 10.

The *forever* loop which runs repeatedly sets the value of a variable called *power* according to the *brightness* level.

To increase and decrease the brightness, two handlers are used. These respond to either a press of button A or button B and then display the brightness level and set the output level on *pin0* using the *analog write* block.

MicroPython

Here is the MicroPython code for the LED dimmer.

```
from microbit import *

min_power = 50
max_power = 1023
power_step = (max_power - min_power) / 9
brightness = 0

def set_power(brightness):
    display.show(str(brightness))
    if brightness == 0:
        pin0.write_analog(0)
    else:
        pin0.write_analog(brightness * power_step + min_power)

set_power(brightness)

while True:
    if button_a.was_pressed():
        brightness -= 1
        if brightness < 0:
            brightness = 0
        set_power(brightness)
    elif button_b.was_pressed():
        brightness += 1
        if brightness > 9:
            brightness = 9
        set_power(brightness)
    sleep(100)
```

Arduino Code

The Arduino code for using the micro:bit's display is a little different because it uses the Adafruit GFX Library (https://adafru.it/doL).

```
#include <Adafruit_Microbit.h>

const int ledPin = 0;

const int minPower = 50;
const int maxPower = 255;
const int powerStep = (maxPower - minPower) / 9;

int brightness = 0;
Adafruit_Microbit_Matrix microbit;

void setup() {
  pinMode(ledPin, OUTPUT);
  microbit.begin();
  pinMode(PIN_BUTTON_A, INPUT);
  pinMode(PIN_BUTTON_B, INPUT);
  setPower(brightness);
}

void loop() {
  if (digitalRead(PIN_BUTTON_A) == LOW) {
    brightness --;
    if (brightness < 0) {
      brightness = 0;
    }
    setPower(brightness);
    delay(200);
  }
  else if (digitalRead(PIN_BUTTON_B) == LOW) {
    brightness ++;
    if (brightness > 9) {
      brightness = 9;
    }
    setPower(brightness);
    delay(200);
  }
}

void setPower(int brightness) {
  microbit.print(brightness);
  if (brightness == 0) {
    analogWrite(ledPin, 0);
  }
  else {
    analogWrite(ledPin, brightness * powerStep + minPower);
  }
}
```

Other Things to Do

Using whatever programming language for the micro:bit that you like best, try modifying the *blink* or *traffic signal* code to change the timing of the LEDs.

For the traffic signal project, you could also try changing the code so that it behaves like a pedestrian crossing and the lights change in response to you pressing button A.

About the Author

If you have found this lesson useful, and want to learn Python, then you might like my book Programming the BBC micro:bit: Getting Started with MicroPython.