Engduino Tutorial

Engduino Support Team - support@engduino.org University College London

1 Introduction

The Engduino is, at heart, an Arduino, and it is programmed using the Arduino IDE (The software you will using to program your board). We write the app for our Arduino is going to execute on our computer and then, using a USB cable and special communication software,**upload** it to our Arduino. An Arduino is a very simple, small computer that has a number of pins that you can connect to external circuitry. Typically, you use some of those pins as inputs to get information in to the Arduino, and some as outputs to cause some external device (motor, lights, etc) to do something. In between, you have some software that reads the inputs, and based on their value, decides how to drive the outputs.

1.1 Overview of the Engduino Board

Your Engduino board comes pre-fitted with some sensors and LEDs on it already so we can get straight into experimenting with programming the board without having to do any construction beforehand.

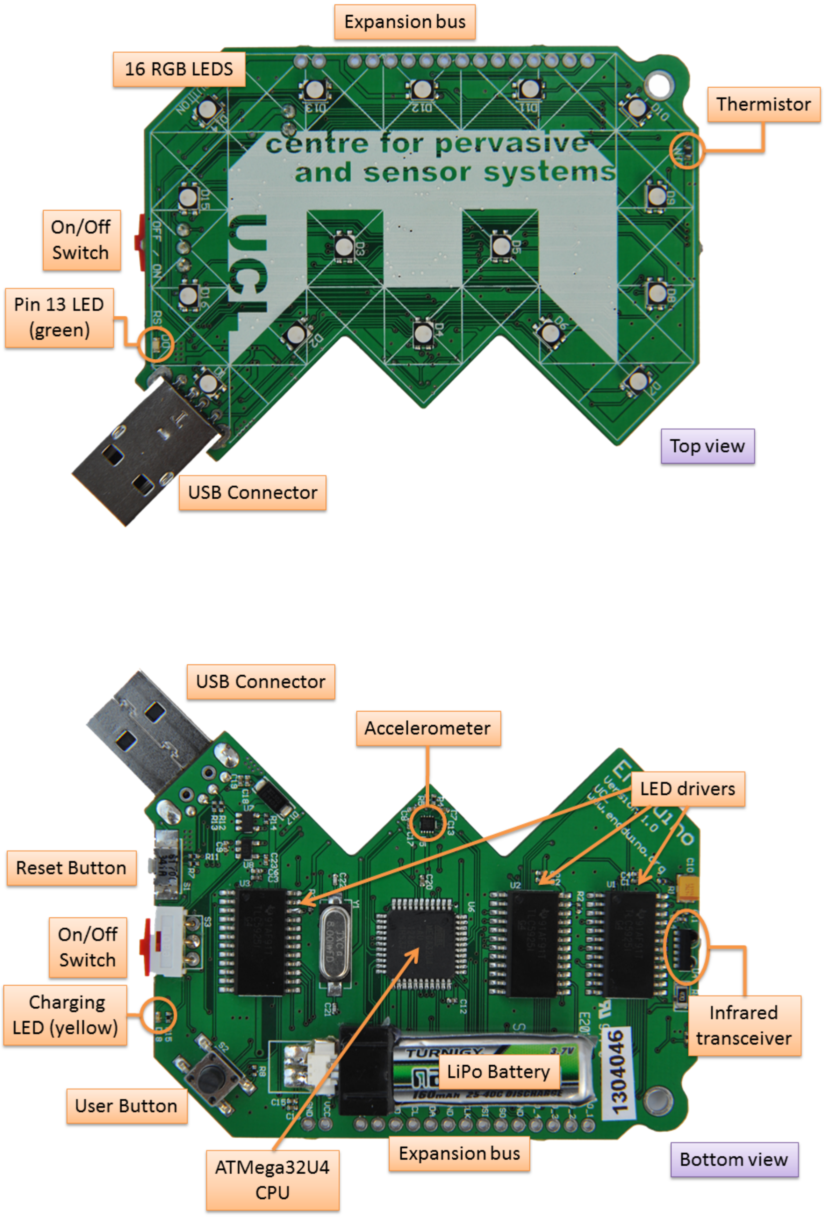


Figure 1: The top and bottom of your Arduino

On the one side of your Engduino you have 16 LED lights that have adjustable brightness and can be set to three colors: Red, Green and Blue.

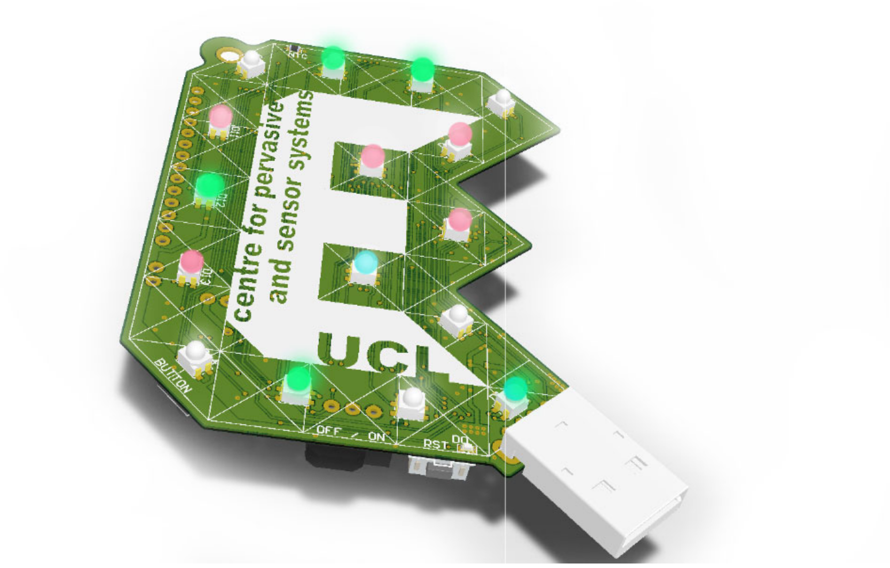


Figure 2: 16 Red, Green, Blue LEDs on one side of the board

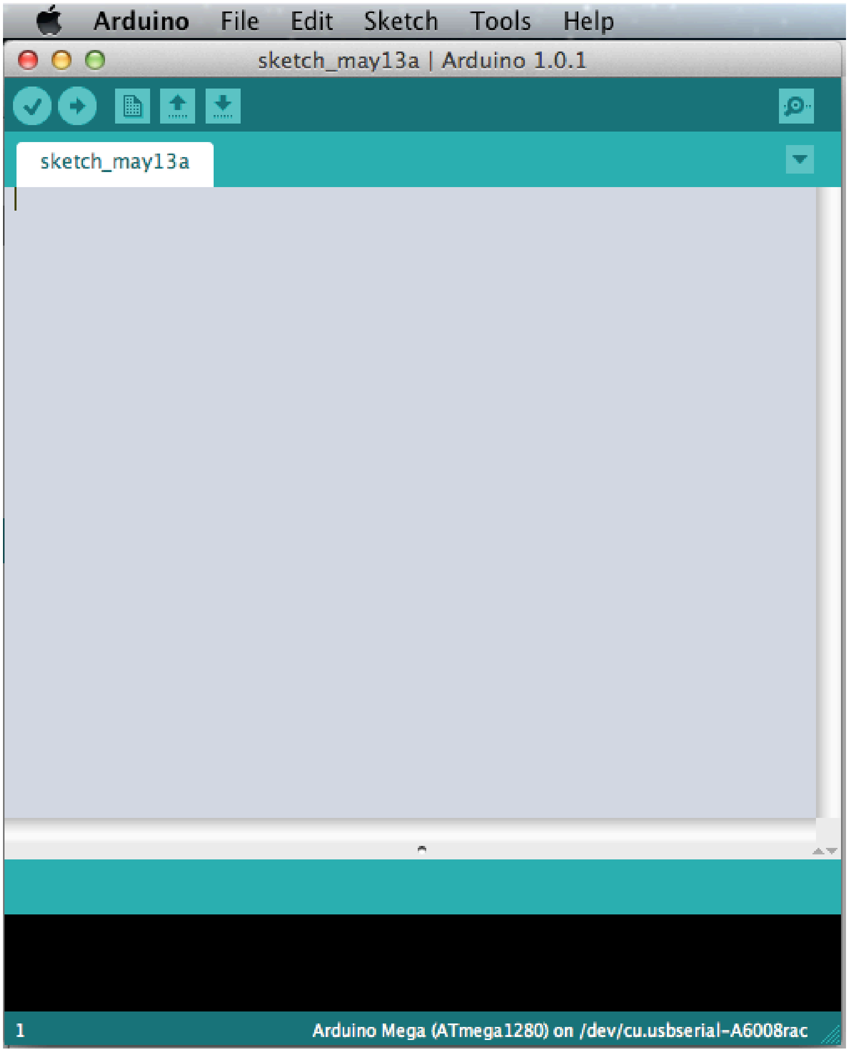
The LEDS are numbered 1 − 16 and their exact positions can be referenced at Appendix A. Also on this side of the board we have a thermistor which can be used to detect temperature. On the other side we have the battery terminals where we can install a battery. The battery charges automatically if it is plugged in while the Engduino is plugged into a USB port. There is also a button that can be used to get input from the user, a power switch to power your device down when you are not using it. The accelerometer is near the keyring hole.

2 Your first app

For your first exercise you are going to load a pre-written program up in your Arduino IDE and then upload it to your Engduino board.

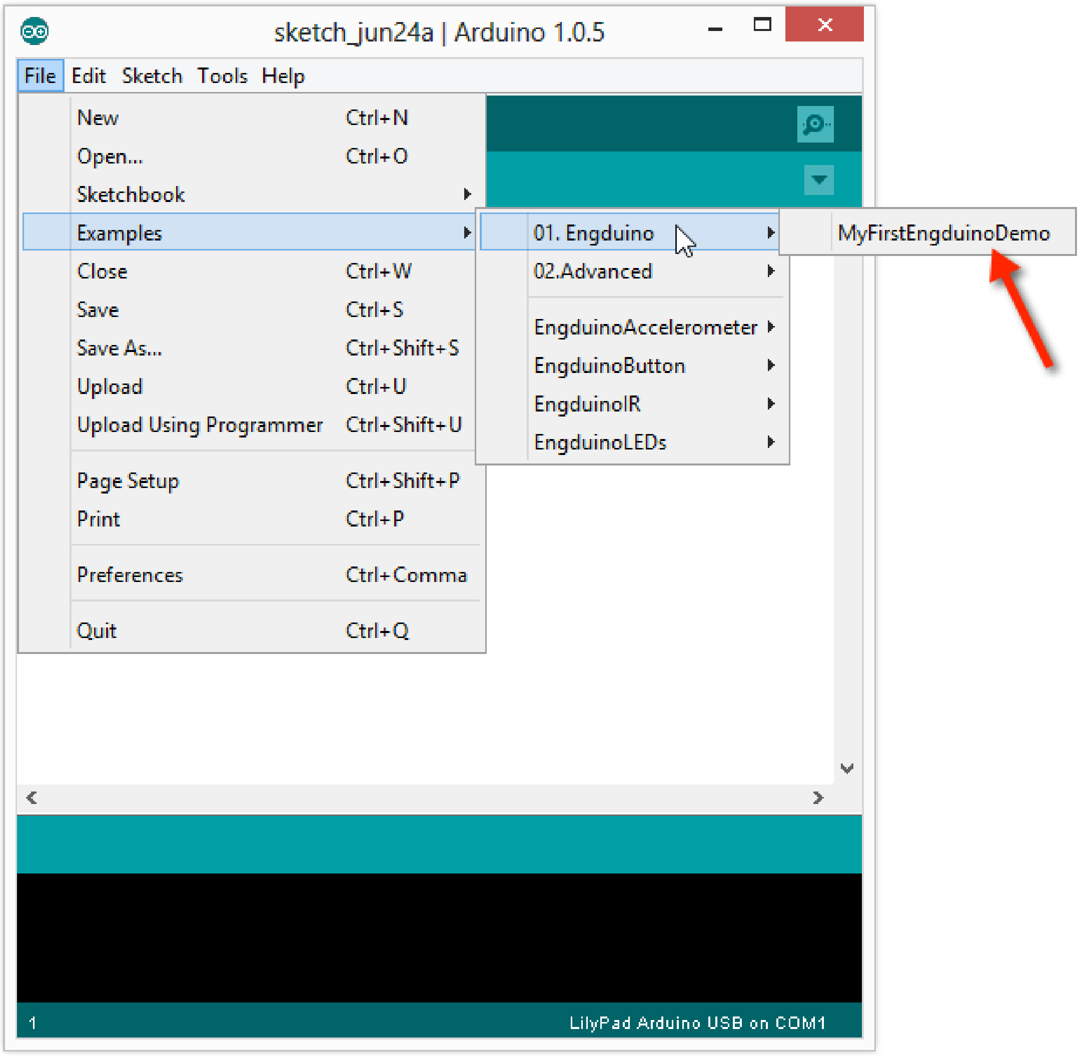
2.1

. Your Arduino IDE should be open for you already. If not, just double click on the blue Arduino icon on your desktop You should see a window like the one shown


Arduino
File
Edit
Sketch
Tools
Help
sketch_may13a | Arduino 1.0.1
sketch_may13a
1
Arduino Mega (ATmega1280) on /dev/cu.usbserial-A6008rac

2.2

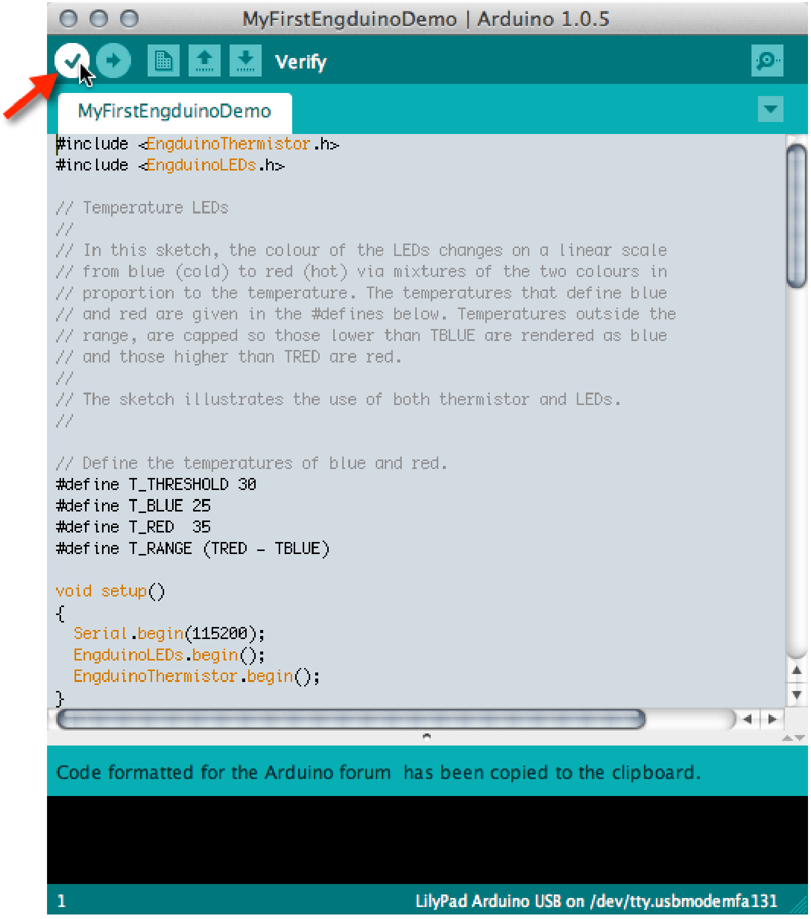
Now we're going to laod the program into the IDE. Go to the File menu, near the top of the IDE window, and select **File** → **Examples** → **01. Engduino** → **MyFirstEngduinoDemo**



A new window will pop up with the pre-written app in it. We don't need to modify this code for now so we're going to go ahead and upload it to the Engduino.

2.3 Uploading

On the top toolbar, click the arrow button to "verify" the app code is correct before uploading.

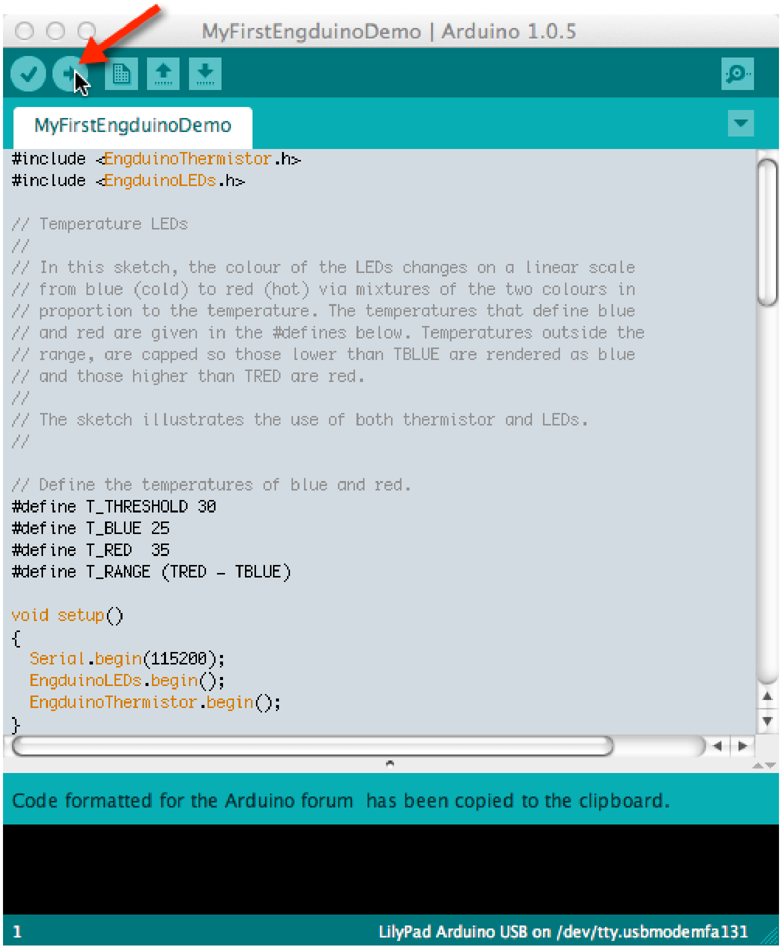


```
#include <EngduinoThermistor.h>
#include <EngduinoLEDs.h>

// Temperature LEDs
//
// In this sketch, the colour of the LEDs changes on a linear scale
// from blue (cold) to red (hot) via mixtures of the two colours in
// proportion to the temperature. The temperatures that define blue
// and red are given in the #defines below. Temperatures outside the
// range, are capped so those lower than TBLUE are rendered as blue
// and those higher than TRED are red.
//
// The sketch illustrates the use of both thermistor and LEDs.
//

// Define the temperatures of blue and red.
#define T_THRESHOLD 30
#define T_BLUE 25
#define T_RED  35
#define T_RANGE (TRED - TBLUE)

void setup()
{
  Serial.begin(115200);
  EngduinoLEDs.begin();
  EngduinoThermistor.begin();
}
```

Code formatted for the Arduino forum has been copied to the clipboard.

1 LilyPad Arduino USB on /dev/tty.usbmodemfa131

Some text will scroll in the black console window near the bottom and the last message should say that the compilation was successful. now click the Up Arrow button which will upload your app to the Engduino board.



```
#include <EngduinoThermistor.h>
#include <EngduinoLEDs.h>

// Temperature LEDs
//
// In this sketch, the colour of the LEDs changes on a linear scale
// from blue (cold) to red (hot) via mixtures of the two colours in
// proportion to the temperature. The temperatures that define blue
// and red are given in the #defines below. Temperatures outside the
// range, are capped so those lower than TBLUE are rendered as blue
// and those higher than TRED are red.
//
// The sketch illustrates the use of both thermistor and LEDs.
//

// Define the temperatures of blue and red.
#define T_THRESHOLD 30
#define T_BLUE 25
#define T_RED  35
#define T_RANGE (TRED - TBLUE)

void setup()
{
  Serial.begin(115200);
  EngduinoLEDs.begin();
  EngduinoThermistor.begin();
}
```

Code formatted for the Arduino forum has been copied to the clipboard.

1 LilyPad Arduino USB on /dev/tty.usbmodemfa131

Some more text will appear in the console window and if you observe your Engduino board you should see a small LED near the USB port flashing as your app is being uploaded to the board. Once it's complete, you should see a message in the black console window saying the upload is complete. Congratulations, you've just programmed yor first Engduino. Obviously you can tell, our app doesn't do anything yet. Let's make it interesting.

3 Customising Your first App

Let's take a quick look at some interesting parts of the basic app code and see how we can customise, personalise or enhance it.

```

void setup()
    {
     EngduinoLEDs.begin();
    }
```

This is a function. A function is similar to the ones you may have experienced in mathematics. It can take some value/s and returns another value after it executes. **void** is what the function returns (nothing in this case), then comes the function name (**setup** in this case) and then brackeets that contain the values we pass to the function (None in this case so the brackets are empty.) The curly brackets delimite our code, so in a file with many functions everything within the set of curly brackets just after our function name belongs to that function. Our funciton **setup()** has just one line

```
      EngduinoLEDs.begin();
```

This line tells the object **EngduinoLEDs** to run the funciton **begin()** which gets them started so we can play with them. The semi-colon at the end of the line is common to many programming languages and signals to the computer that is the end of one instruction. Don't forget these when you start writing your own code. The next function

```

void loop()
    {

    }
```

is called **loop()** and as you might have guessed returns nothing (void) and takes no values (hence the empty brackets). The loop function is where we give instructions to the Engduino to perform. As you can see there is no code between the curly brackets and this is why our app does nothing so far. Let's fix that!

To begin with let's try something simple. We'll tell the Engduino to switch on all the LEDs. Inside the Arduino IDE editor window, delete the lines 14 – 16

```
14      /**
15       * Your app instructions go here.
16       */
```

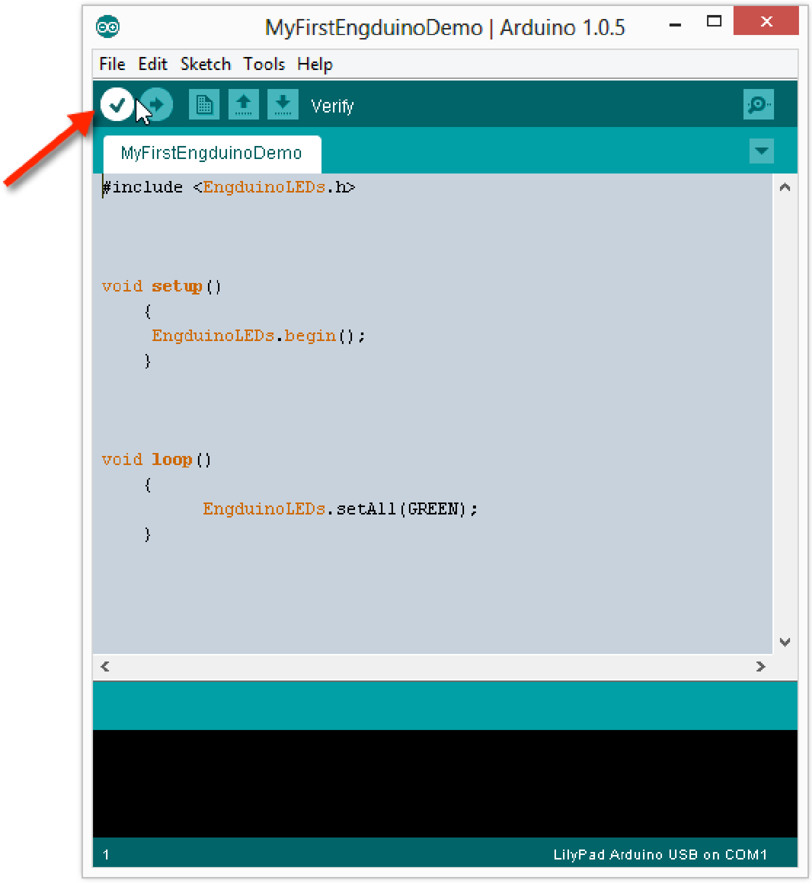
and insert the instruciton

```
    EngduinoLEDs.setAll(GREEN);
```

So now your program should look like this

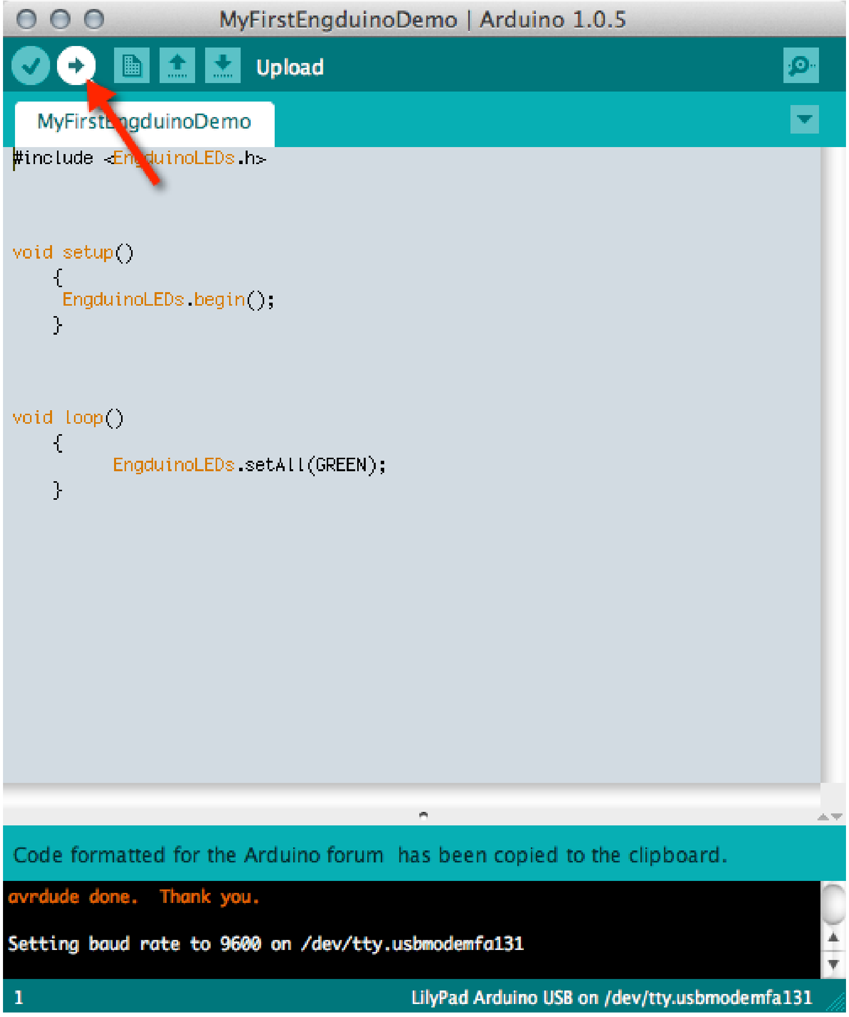
```
 #include <EngduinoLEDs.h>



void setup()
    {
     EngduinoLEDs.begin();
    }



void loop()
    {
      EngduinoLEDs.setAll(GREEN);
    }
```

If it does, go ahead and verify the app by clicking on the tick button on the IDE


MyFirstEngduinoDemo | Arduino 1.0.5
File Edit Sketch Tools Help
Verify
MyFirstEngduinoDemo
1
LilyPad Arduino USB on COM1

```
#include <EngduinoLEDs.h>



void setup()
    {
     EngduinoLEDs.begin();
    }



void loop()
    {
          EngduinoLEDs.setAll(GREEN);
    }
```

once the compilation messages finish showing in the black console window, the app is ready for uploading. Go ahead and click on the upload Arrow button to upload your app.



```
#include <EngduinoLEDs.h>


void setup()
    {
     EngduinoLEDs.begin();
    }


void loop()
    {
          EngduinoLEDs.setAll(GREEN);
    }
```

Code formatted for the Arduino forum has been copied to the clipboard.

```
avrdude done.  Thank you.

Setting baud rate to 9600 on /dev/tty.usbmodemfa131
```

1   LilyPad Arduino USB on /dev/tty.usbmodemfa131

Some messages will appear at in the console, and when it's done you should see the lights on your Engduino board light up and turn green! The line

```
    EngduinoLEDs.setAll(GREEN);
```

is where all the magic happens. This line tells the object **EngduinoLEDs** (all the LEDs) to execute the function setAll(). We pass this funciton a value, namely what color to make the LEDs (Green in this case). It doesn't return a value but simply does what you tell it, and switches the LEDs on.

3.1 Personalise

The function **setAll()** can take a number of different types of argument including three pre-defined colours; Red, Green, and Blue. Go agead and delete the word GREEN inside the brackets and replace it with the color of your choice all in Capital letters. For example if you prefer red, your app will look like this.

```
#include <EngduinoLEDs.h>



```

```
void setup()
    {
     EngduinoLEDs.begin();
    }



void loop()
    {
      EngduinoLEDs.setAll(RED);
    }
```

When you've done that. Verify your code with the tick button like before and then wait for the verification to complete and then click **Upload** to upload your code. The LEDs will no be lit according to the colour you choose. Well done!

3.2 Delay

The Engduino has a function built in called **delay()**. This function takes a number which tells the board to wait those number of milliseconds (remember a millisecond is one thousandth of a second) before executing the next function. We are going to use this useful little funciton to make our Engduino blink.

3.3 Blink

Just below the line

```
    EngduinoLEDs.setAll(RED);
```

add the lines

```
    delay(2000);
    EngduinoLEDs.setAll(OFF);
    delay(1000);
```

Don't forget your semi-colons! Your code should now look something like this. (With the color you chose instead of RED if you chose something else.)

```
    #include <EngduinoLEDs.h>



void setup()
    {
     EngduinoLEDs.begin();
    }



void loop()
```

```
    {
      EngduinoLEDs.setAll(RED);
      delay(2000);
      EngduinoLEDs.setAll(OFF);
      delay(1000);
    }
```

Go ahead and verify your code with the tick buton and then one that's done upload it. You should see your Engduino now rythmically blinking the LEDs in the color that you chose. Go ahead and experiment with the timings now. Change the numbers inside the brackets of the delay functions in both the lines

```
    delay(2000);

    delay(1000);
```

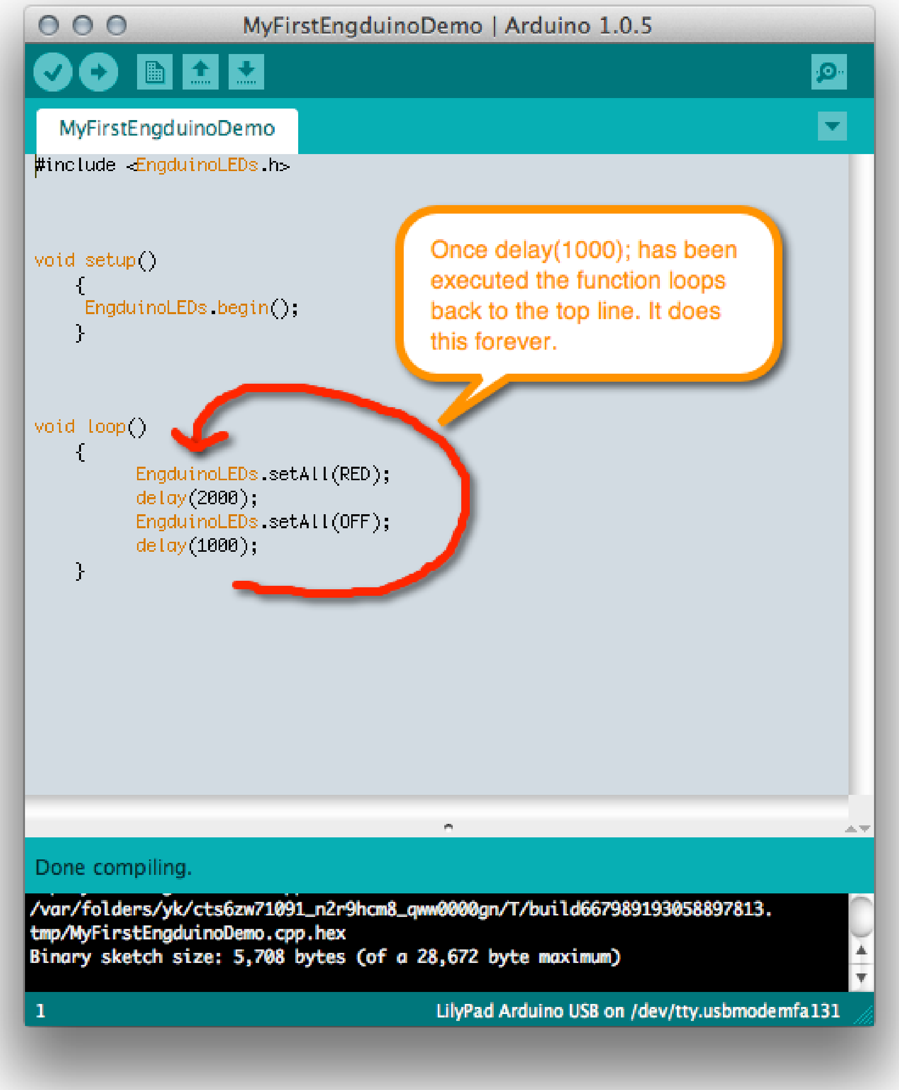
to some number between $800 - 5000$ or 0.8 seconds to 5seconds. So your personalised app code may look something like this

```
    delay(3050);
    EngduinoLEDs.setAll(OFF);
    delay(900);
```

By playing around with combinations of numbers you can modify the sequence the blinking takes. See how different blink sequencing and coloring can have a meaning. For example, a fast blinking red board could signify danger. A constant green could mean ”All clear”.

As you may have noticed, the app function **loop()** executes line by line from top to bottom. The Engduino sets the LEDs to a color. Then it waits for a specified amount of time. Then it turns them all off and then waits again.

If you're very smart you may have guessed that the **loop()** function does just what it says. When it gets to the last line inside the curly brackets it loops back around again and starts from the first line inside the curly brackets



```
#include <EngduinoLEDs.h>


void setup()
    {
     EngduinoLEDs.begin();
    }


void loop()
    {
         EngduinoLEDs.setAll(RED);
         delay(2000);
         EngduinoLEDs.setAll(OFF);
         delay(1000);
    }
```

Once delay(1000); has been executed the function loops back to the top line. It does this forever.

Done compiling.

```
/var/folders/yk/cts6zw71091_n2r9hcm8_qww0000gn/T/build667989193058897813.
tmp/MyFirstEngduinoDemo.cpp.hex
Binary sketch size: 5,708 bytes (of a 28,672 byte maximum)
```

If we were to add the lines

```
        EngduinoLEDs.setAll(GREEN);
        delay(2000);
        EngduinoLEDs.setAll(OFF);
        delay(1000);
```

to the bottom so our code would look like this

```
        EngduinoLEDs.setAll(RED);
        delay(2000);
        EngduinoLEDs.setAll(OFF);
        delay(1000);
         EngduinoLEDs.setAll(YELLOW);
        delay(2000);
```

```
      EngduinoLEDs.setAll(OFF);
      delay(1000);
```

What do you think would happen? Well done if you guessed it would blink one color then blink another!

4 Traffic Lights

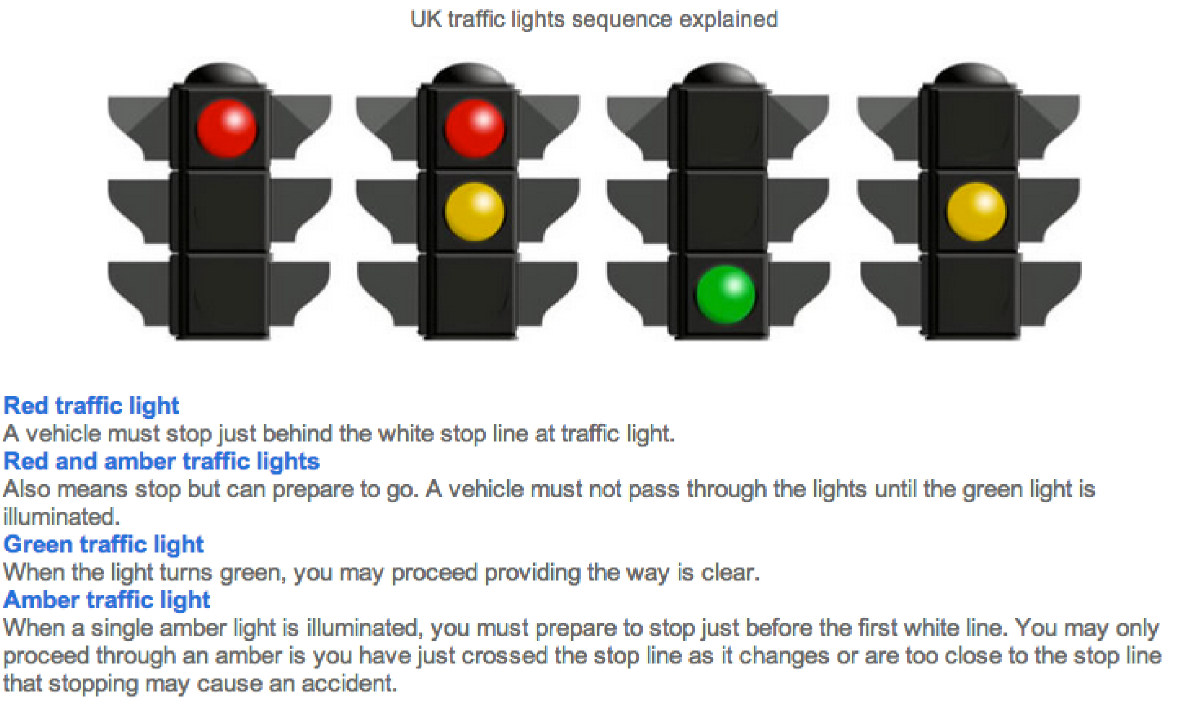
Bravo if you’ve got here. You now know enough to make your own traffic light!
If you think about it a traffic light does almost the same thing as our app except it changes three colors and the timings are slightly different. We can make a couple more modifications and ours will do the same. Let’s add the lines

```
      EngduinoLEDs.setAll(GREEN);
      delay(2000);
      EngduinoLEDs.setAll(OFF);
      delay(1000);
      EngduinoLEDs.setAll(YELLOW);
      delay(2000);
      EngduinoLEDs.setAll(OFF);
      delay(1000);
```

after the last line but still inside the curly brackets. Then change the sequence so that the Lights go from RED to YELLOW to GREEN and then finally back to YELLOW. Your code should look like this

```
    #include <EngduinoLEDs.h>



void setup()
    {
     EngduinoLEDs.begin();
    }



void loop()
    {
      EngduinoLEDs.setAll(RED);
      delay(2000);
      EngduinoLEDs.setAll(OFF);
      delay(1000);
       EngduinoLEDs.setAll(YELLOW);
      delay(2000);
      EngduinoLEDs.setAll(OFF);
      delay(1000);
       EngduinoLEDs.setAll(GREEN);
      delay(2000);
      EngduinoLEDs.setAll(OFF);
      delay(1000);
26        EngduinoLEDs.setAll(YELLOW);
27       delay(2000);
28       EngduinoLEDs.setAll(OFF);
29       delay(1000);
30
31
32     }
```

Once you're sure your code looks like this. You can go ahead and verify it with the tick button and then upload it with the up arrow on the Arduino IDE.
You'll see your Engduino wiil begin to look a traffic light but the timing seems all wrong. Experiment with different timings to see if you can get it to look like a traffic light.
sectionadvanced If you still have time, lets cahnge the lighting so it more accurately reflects UK light signals. If you remember, UK traffic signals have the following pattern



UK traffic lights sequence explained

Red traffic light
A vehicle must stop just behind the white stop line at traffic light.
Red and amber traffic lights
Also means stop but can prepare to go. A vehicle must not pass through the lights until the green light is illuminated.
Green traffic light
When the light turns green, you may proceed providing the way is clear.
Amber traffic light
When a single amber light is illuminated, you must prepare to stop just before the first white line. You may only proceed through an amber is you have just crossed the stop line as it changes or are too close to the stop line that stopping may cause an accident.

So when our traffic light changes from Red to the nest phase we want our Engduino to show half Red lights and half Yellow. This is quite easy. We just have to write a little more code because we will have to set individual LEDs to colors rather than all of them with a single line. The function **EngduinoLEDs.setLED()** takes two values (separated by a comma). The first is the number of the LED you want to access and the other is the color you want that LED to be. So for example, if I wanted to switch LED number 1 to yellow the code would look like this.

```
31     EngduinoLEDs.setLED(1,YELLOW);
```

Remember the Engduino has 16 LEDs so perhaps if we set 1 – 8 to Red and 9 – 16 to Yellow, we might make it look how we want it to. At line 48 delete the line

```
    EngduinoLEDs.setAll(YELLOW);
```

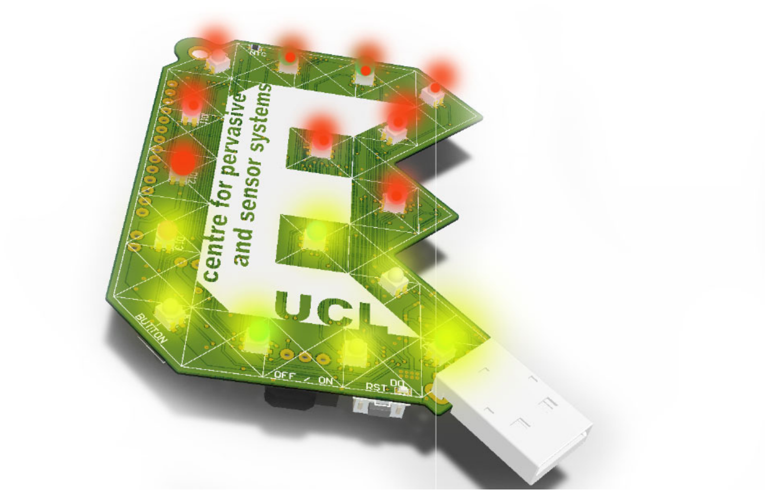
and replace it with the following lines. (Don't forget the copy and paste functionality in the Arduino **Edit** menu item to help save you some time)

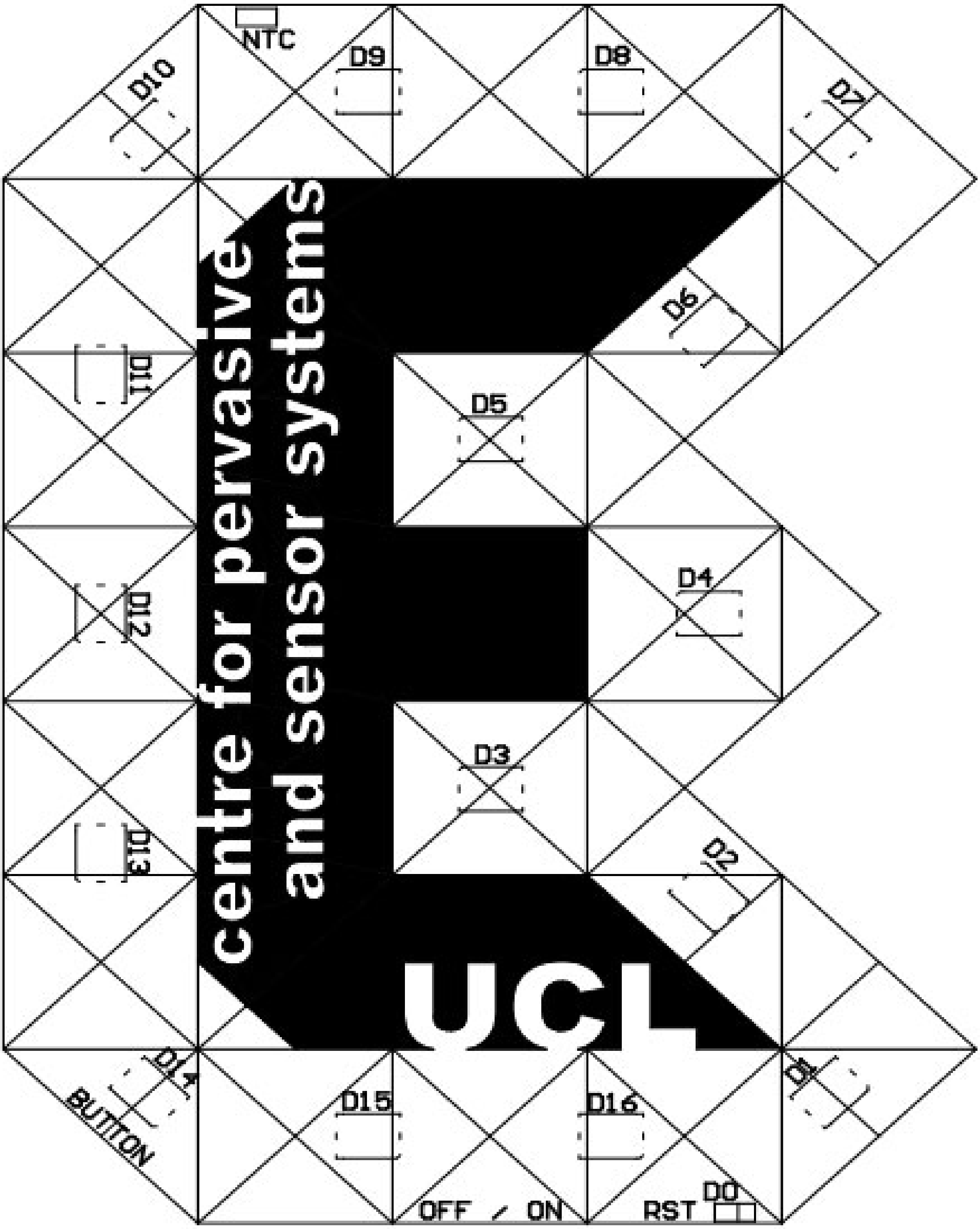
```
    EngduinoLEDs.setLED(1,RED);
    EngduinoLEDs.setLED(2,RED);
    EngduinoLEDs.setLED(3,RED);
    EngduinoLEDs.setLED(4,RED);
    EngduinoLEDs.setLED(5,RED);
    EngduinoLEDs.setLED(6,RED);
    EngduinoLEDs.setLED(7,RED);
    EngduinoLEDs.setLED(8,RED);
    EngduinoLEDs.setLED(9,YELLOW);
    EngduinoLEDs.setLED(10,YELLOW);
    EngduinoLEDs.setLED(11,YELLOW);
    EngduinoLEDs.setLED(12,YELLOW);
    EngduinoLEDs.setLED(13,YELLOW);
    EngduinoLEDs.setLED(14,YELLOW);
    EngduinoLEDs.setLED(15,YELLOW);
    EngduinoLEDs.setLED(16,YELLOW);
```

So your final code should look something liek this. (With your own tweaked timings in the **delay()** functions).

```
#include <EngduinoLEDs.h>



void setup()
    {
     EngduinoLEDs.begin();
    }



void loop()
    {
      EngduinoLEDs.setAll(RED);
      delay(2000);
      EngduinoLEDs.setAll(OFF);
      delay(1000);

        EngduinoLEDs.setLED(1,RED);
        EngduinoLEDs.setLED(2,RED);
        EngduinoLEDs.setLED(3,RED);
        EngduinoLEDs.setLED(4,RED);
        EngduinoLEDs.setLED(5,RED);
        EngduinoLEDs.setLED(6,RED);
        EngduinoLEDs.setLED(7,RED);
        EngduinoLEDs.setLED(8,RED);
        EngduinoLEDs.setLED(9,YELLOW);
        EngduinoLEDs.setLED(10,YELLOW);
        EngduinoLEDs.setLED(11,YELLOW);
        EngduinoLEDs.setLED(12,YELLOW);
        EngduinoLEDs.setLED(13,YELLOW);
        EngduinoLEDs.setLED(14,YELLOW);
        EngduinoLEDs.setLED(15,YELLOW);
        EngduinoLEDs.setLED(16,YELLOW);
      delay(2000);


      EngduinoLEDs.setAll(OFF);
      delay(1000);
       EngduinoLEDs.setAll(GREEN);
      delay(2000);
      EngduinoLEDs.setAll(OFF);
      delay(1000);
       EngduinoLEDs.setAll(YELLOW);
      delay(2000);
      EngduinoLEDs.setAll(OFF);
      delay(1000);


    }
```

Now it’s starting to look ohw we want it to. You can now change the colors of the individual LEDs to red or yellow so that you get the sape that you want. For example so that when the Traffic light is Red/Yellow it looks something like this.




NTC
D10
D9
D8
D7
D6
D11
D5
D4
D12
D3
D2
D13
centre for pervasive
and sensor systems
UCL
D14
BUTTON
D15
D16
D1
OFF / ON
RST
D0