Traffic Lights with an Arduino

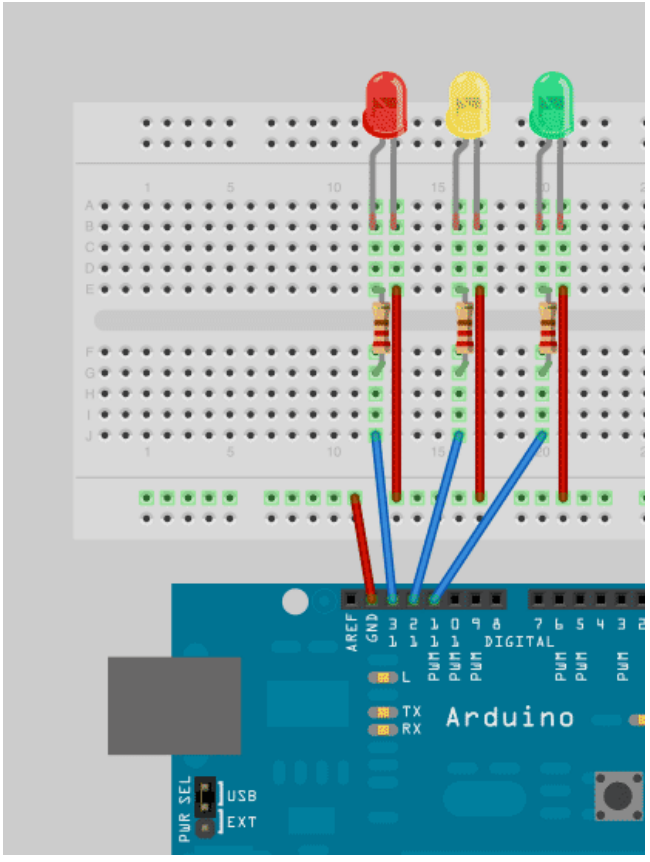
In this project we'll build a basic traffic light system using the Arduino board and LEDs. We'll then extend the functionality to include a push-button to manually change the lights

Apart from the basic Arduino, you'll need:

- A red, yellow and green LED
- A breadboard
- 3 x suitable resistors for the LEDs
- Connecting wires
- A pushbutton switch
- A high value resistor (10k)

Wiring

Here is a schematic for the wiring required :



It's very simple – just the three LEDs wired with resistors to three separate input pins, and all connected to the ground.

Programming

Setup

We'll start by defining variables so that we can address the lights by name rather than a number.

Start a new Arduino project, and begin with these lines:

```
int red = 13;
int yellow = 12;
int green = 11;
```

These define the red, yellow and green LEDs as input pins 13,12 and 11

Next, let's add the setup function, where'll we define the red, yellow and green LEDs to be output mode.

Since we've created variables to represent the pin numbers, we can now refer to the pins by names instead.

```
void setup(){
    pinMode(red,OUTPUT);
    pinMode(yellow,OUTPUT);
    pinMode(green,OUTPUT);
}
```

Logic

Now we're going to write the actual logic for controlling the traffic lights

Inside the `loop` function, we're going to create a separate function for changing the lights called `changeLights`, and call it every 15 seconds using the `delay` function

```
void loop(){
    changeLights();
    delay(15000);
}
```

We can now write the `changeLights()` function. It will switch

```
void changeLights(){
    // green off, yellow for 3 seconds
    digitalWrite(green,HIGH);
    digitalWrite(yellow,LOW);
    delay(3000);

    // turn off yellow, then turn red on for 5 seconds
    digitalWrite(yellow,LOW);
    digitalWrite(red,HIGH);
    delay(5000);

    // red and yellow on for 2 seconds (red is already on)
    digitalWrite(yellow,HIGH);
    delay(2000);

    // turn off red and yellow, then turn on green
    digitalWrite(yellow,LOW);
    digitalWrite(red,LOW);
    digitalWrite(green,HIGH);
}
```

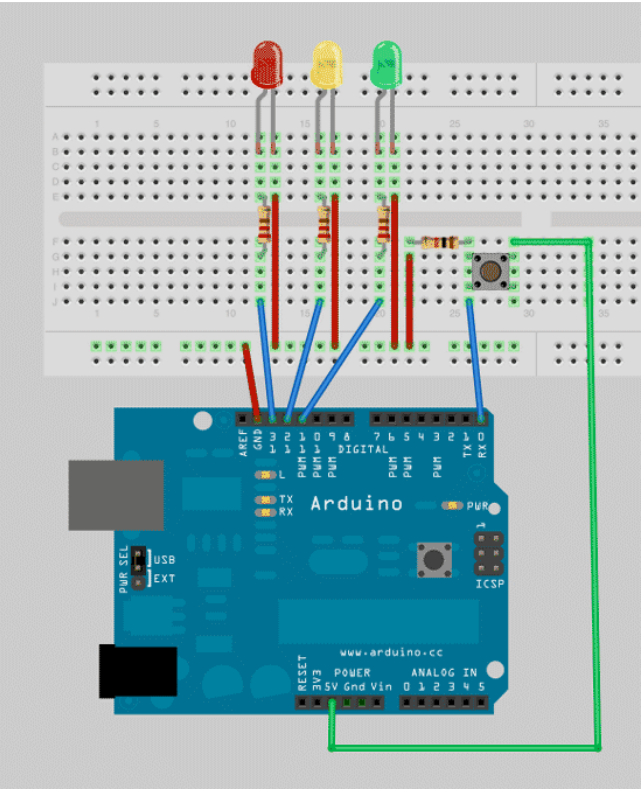
Note that in the code we don't have to turn on a light if it is already on, it will stay in the same state unless we change it

Now, upload and run. You should have a working traffic light that changes every 15 seconds.

Extending The Program

We now have a working traffic light system, but it's still a little basic. We can extend it to include a pushbutton that will change the lights as required

The new schematic is as follows :



> You'll notice that the switch has a high-impedance 10k resistor attached to it, and may be wondering why. This is called a pull down resistor. It's a difficult concept to grasp at first, but bear with me.
>
> A switch either lets the current flow, or doesn't. This seems simple enough, but in a logic circuit, the current should be always flowing in either a high or low state (remember – 0 or 1, high or low). You might assume that a pushbutton switch that isn't actually being pushed would be defined as being in a low state, but in fact it's said to be 'floating', because no current is being drawn at all.
>
> In this floating state, it's possible that a false reading will occur as it fluctuates with electrical interference. In other words, a floating switch is giving neither a reliable high, nor low state reading. A pull down resistor keeps a small amount of current flowing when the switch is closed, thereby ensuring an accurate low state reading. In other logic circuits, you may find a pull-up resistor instead – this works on the same principle, but in reverse, making sure that particular logic gate defaults to high.

We'll start by adding some new variables to the start of the app:

```
int button = 2; // switch is on pin 2
int buttonValue = 0; // switch defaults to 0 or LOW
```

In the `setup` function, we'll add a new line to declare the switch as an input.

We'll also add a single line to start the traffic lights in the green stage. Without this initial setting, they would be turned off, until the first time a `changeLights()` was initiated using a function.

```
pinMode(switch,INPUT);
digitalWrite(green,HIGH);
```

Now the key part - In the `loop` part of the code, instead of changing the lights every 15 seconds, we're going to read the state of the pushbutton switch instead, and only change the lights when it's activated

Change the entire `loop` function to this instead

```
void loop(){
    // read the value of the switch
    switchValue = digitalRead(button);
    // if the switch is HIGH, ie. pushed down - change the lights!
    if (buttonValue == HIGH){
        changeLights();
        delay(15000); // wait for 15 seconds
    }
}
```

By waiting inside the "if" statement for 15 seconds, we ensure the traffic lights can't change for at least that duration. Once 15 seconds is up, the loop restarts.

Each restart of the loop, we will read the state of the button again, but if it isn't pressed then the "if" statement never activates, the lights never change, and it simply restarts again.