**Mini Project**

Embedded Systems Programming, **4CS016**



***Faculty of Science and Engineering***

**Abstract**

The purpose of this mini project is to design and implement an embedded system using hardware and software to make an interactive traffic lights system for cars and pedestrians. This will be produced by utilising the Arduino Uno development platform, and the programming will be generated by using the Arduino IDE environment. The results will be displayed through the four LED's constructed on the breadboard. From completion of this mini project we learned a new programming language C, how to effectively control an electronic interactive traffic lights circuit of LED lights by connecting them to specific ports on the Arduino, and calling them within the Arduino programming code. In addition we learned how to make use of actuators within the embedded system, and how they proved to be a vital part in ensuring the traffic lights worked harmoniously.

**Introduction**

The Arduino Uno is one of the most popular open-source micro controller boards, and for good reason. In 2005 the Arduino platform was created to provide a cheaper way for the professional world, students and as a hobby to construct programs that perform in the human interface world using actuators, sensors, and motors. It's designed to make applications, interactive controls or environments easily adaptable, making it perfect for creating an embedded system for specifically one purpose only.

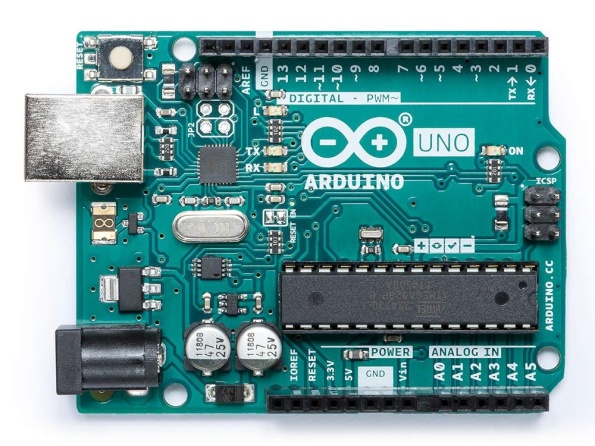
Being open-source, it's design is made freely and openly available for everyone so that you are able to make and sell and distribute the design along with the hardware based on that design. Being able to pick up a basic one for as little as £3-10, it's perfect for DIY projects and first-timers creating a mini project. In addition, you can use it across-platforms, meaning you can program the Arduino on Windows, Macintosh OSX, and Linux operating systems; most micro controller boards are tied to Windows only. Furthermore, another great benefit of using the Arduino Uno is that it encompasses a core set of library functions that control secondary connected hardware that you are using, as standard.

There's lots of sensors and actuators that can be used with the Arduino. Being aware of the Arduino memory and it's CPU constraints, it can be used effectively to control many things in small DIY/professional projects; control in the respect of temperature, light, humidity, and respond when a limitation is reached such as turning on a LED light, pump or heater etc.

The Arduino Uno allows users to program using the C or C++ programming language. Programming an Arduino in C makes use of its memory far more efficiently. The C programming language is perfect for getting the most out of your code/project whilst still keeping it fairly user-friendly to understand. Perfect for our purposes.

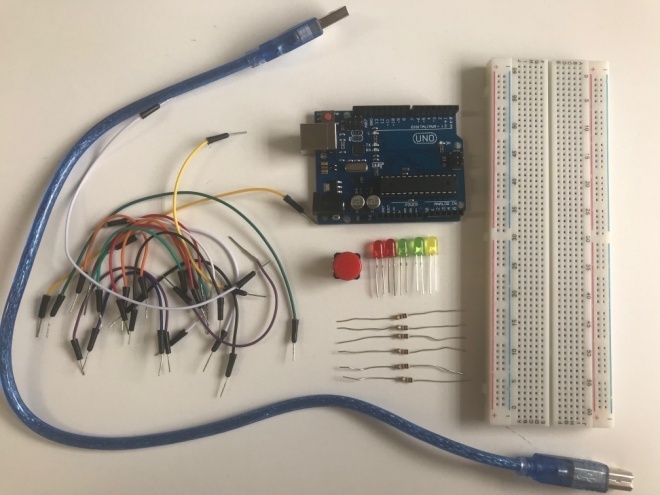
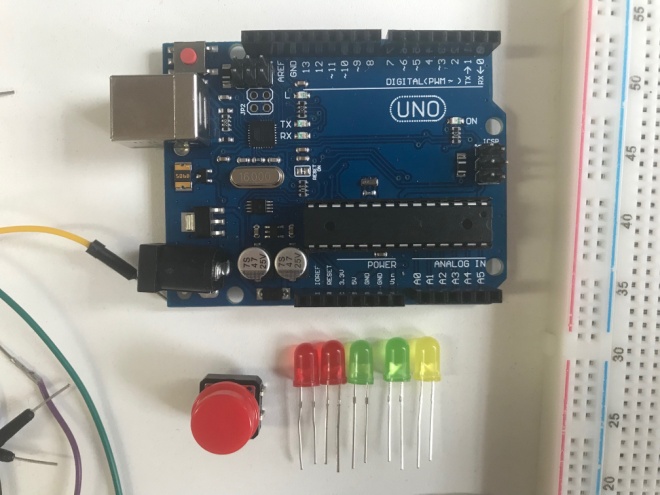
The Mini Project Assessment specifies that we are to create an embedded system by utilising with an Arduino Uno micro controller board. This mini project will take advantage of LED's, resistors, a breadboard, a push button and the Arduino Uno to create a fully-functioning traffic lights set for cars and pedestrians. The traffic lights give the LED feedback to pedestrians and cars notifying them when it's safe to go, and when to stop.

**Materials**



images source: [Amazon](https://www.amazon.com/Arduino-Uno/dp/B0044X2E5S)

* Arduino UNO Microcontroller Board
* C programming language
* Arduino IDE and Fritzing Software
* Breadboard
* 2 x Red LED's, 1 x Yellow LED, 2 x Green LED's, 6 x 150 Ohm Resistors, 1 x Tactile switch
* Male jumper wires

**Specifications:**

Arduino UNO microcontroller board

Based on the ATmega328P

SRAM 2KB (ATmega328P)

Flash Memory 32KB (ATmega328P); 0.5KB of this is used by bootloader

Clock Speed 16MHz

Input Voltage Recommended: 7-12 V Limit: 6-20 V

Digital I/O Pins 14

Pulse Width Modulation I/OPins 6

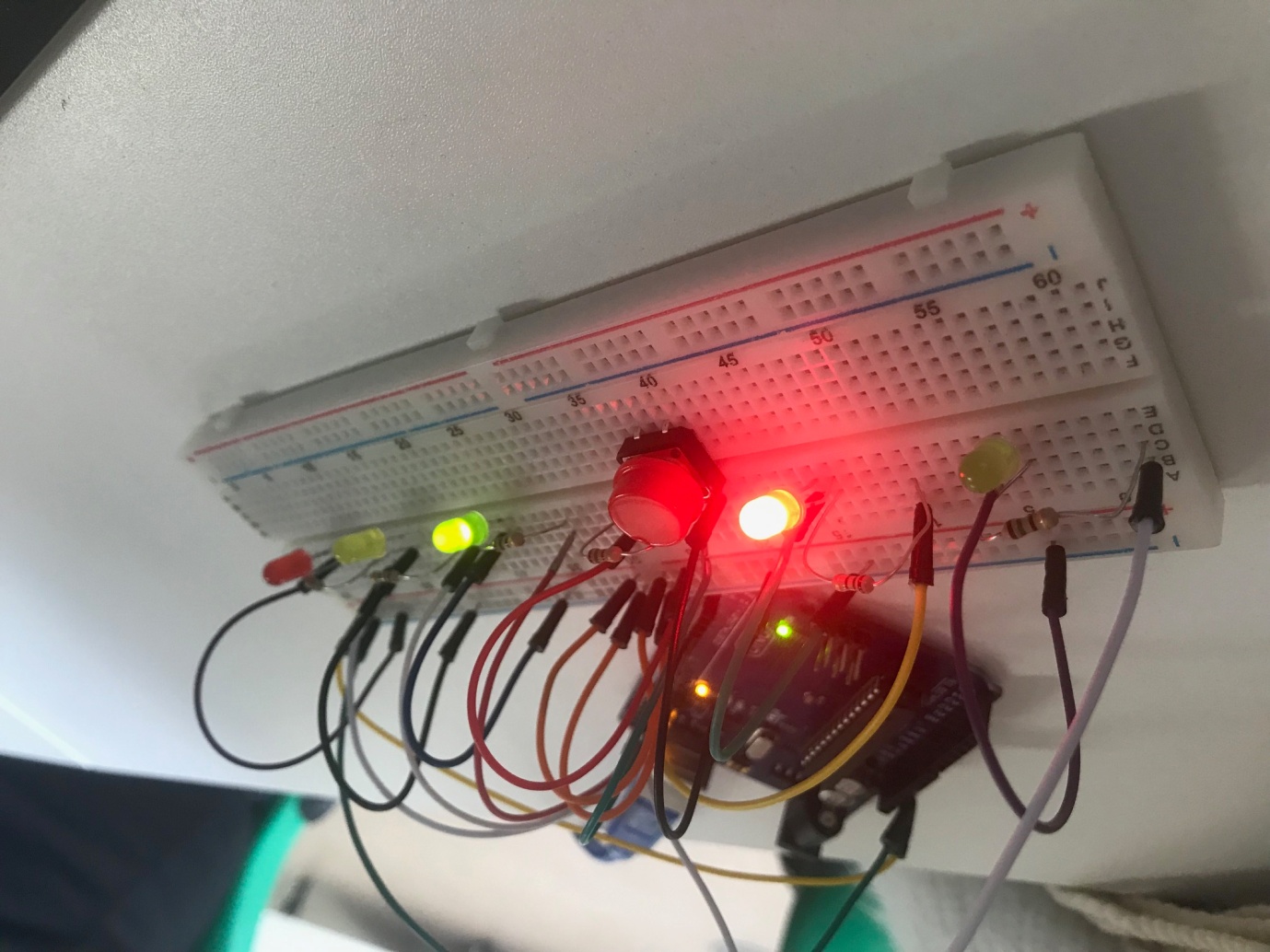
Analog Input Pins 6

C Programming Language

* Structured programming compiled language (code is compacted into executable instruction)
* Made up of modules and blocks, collections of which make up a complete program
* Highly portable language
* Rich library resource and a number of built-in functions
* Dynamic memory allocation

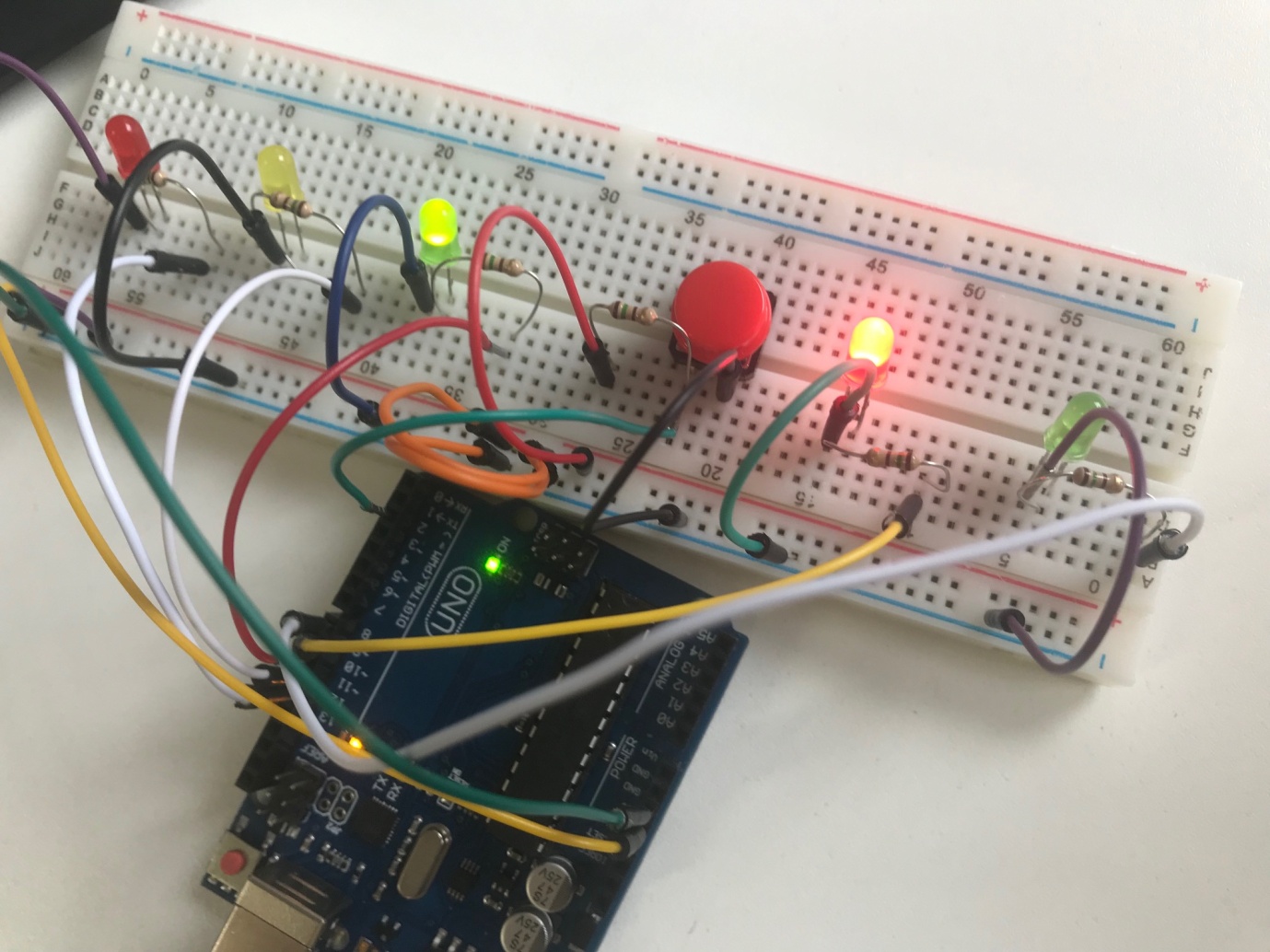
Interesting to note however, that the kind of C language which is used to program an Arduino UNO is not the normal C language, but a slight variation towards Object Orientated C. Kind of a mixture of concepts of Java with syntaxes of C language.

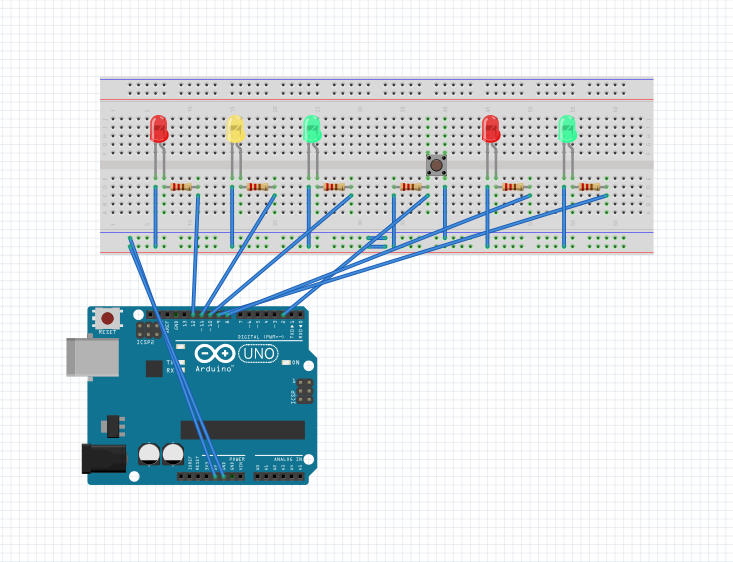
**Method**



Actuators are the elements of the traffic lights embedded system which are accountable for moving and controlling a system, by means of a control signal and source of energy. The sensors and actuators which were used in this embedded system mini project:

An input push button as an analogue actuator input, and LED's as output actuators. A push button in this mini project causes the device to perform a specific action. I.e. start the process through the programming code (if state == HIGH) of changing the LED lights for the cars from green to red, and the LED lights for the pedestrians from red to green, and then back to green again for the cars and red for the pedestrians. The state global variable is what enables the coding to perform this action when the button is pressed. High state is equal to the push button being pressed.





Above is the Fritzing design I created alongside constructing the programming code for the interactive traffic lights embedded system.

After the programming for the interactive traffic lights was complete, it was time to connect the LED's in the order red, yellow, green (car LED's), red, green (pedestrian LED's) on the breadboard, with a switch in between the car and pedestrian's. Connect the 150 ohm resistors in series. Connect the male jumper wires in their appropriate places. Connect the other end of the wires from the resistors to the Arduino Uno in the pins (12, 11, 10, 9, 8), the push button into the pin 2, as well as ensuring the breadboard was connected to to the ground of the Arduino.

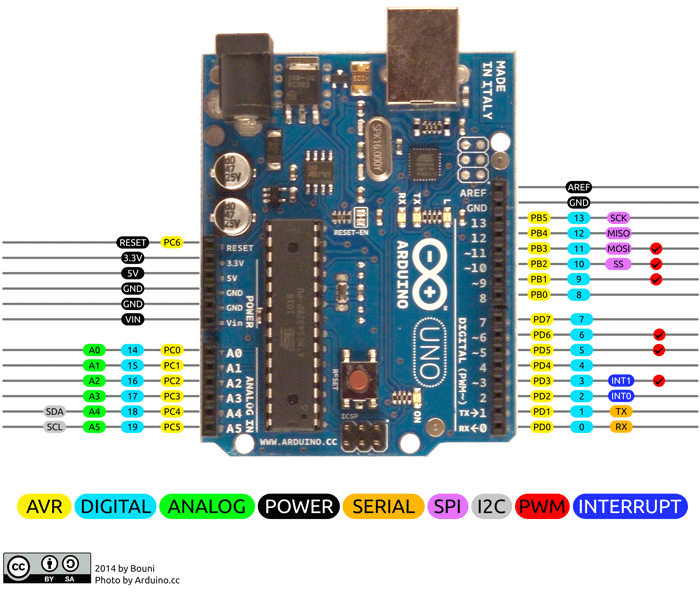


image source: [Components 101](https://components101.com/microcontrollers/arduino-uno)

The code for this Arduino Traffic lights mini project is fairly simple and relatively easy to understand. It's a traffic lights for both cars and pedestrians, the code enables the LED's to glow in a particular sequence when the push button for the pedestrians is pressed. Once the button is pressed, it initialises the sequence of LED lights to change; the yellow LED for the cars begins to flash, indicating for them to stop as the lights will change to red soon. It flashes for 5 seconds, before turning off and igniting the red car LED light to turn on. After 1 second of the red LED for the cars has been on, the red LED for pedestrians turns off and the green LED for the pedestrians turns on. After 5 seconds of the green LED for pedestrians is turned on, it then begins to flash for 5 seconds, before turning off and the red LED for the pedestrians turns on. This in turn allows the yellow car LED to turn on for 1 second, before proceeding to the green LED for the cars, and turning the yellow LED off. This process is demonstrated nicely in the video below in the results section of this report. This process is looped within the voidloop() function enabling its continuous action.

The LED lights are necessary within this circuit to notify both cars and pedestrians, when it is their turn to go and to stop. Without clearly notifying them in this clear manner, all hell would break loose and accidents (possibly fatalities) would occur. The push button actuator is necessary in order for there to be a means for which the pedestrians can control when they'd like to cross the road. By implementing this function, they are able to cross in a safe bearing shortly after pressing the button. Likewise with the coding that partners up with the push button actuator. Furthermore, the 6 resistors within the circuit are crucial for the LED's and circuit. The resistors are required in order to limit the current through the LED, preventing it from blowing the bulb. It reduces the current flow within the circuit in order to lower the voltage levels within the circuit.

**Results**

**CODE:**

// Interactive Traffic Lights

int carRed = 12; // To assign the car lights

int carYellow = 11;

int carGreen = 10;

int pedRed = 9; // To assign the pedestrian lights

int pedGreen = 8;

int button = 2; // Tactile button pin

int crossTime = 5000; // time allowed to cross

unsigned long changeTime; // time since button pressed

void setup() {

pinMode(carRed, OUTPUT);

pinMode(carYellow, OUTPUT);

pinMode(carGreen, OUTPUT);

pinMode(pedRed, OUTPUT);

pinMode(pedGreen, OUTPUT);

pinMode(button, INPUT); // button on pin 2

// turn on the green light

digitalWrite(carGreen, HIGH);

digitalWrite(pedRed, HIGH);

}

void loop() {

int state = digitalRead(button);

/\* check if button is pressed and it is

over 5 seconds since last button press \*/

if (state == HIGH && (millis() - changeTime) > 5000) {

// Call the function to change the lights

changeLights();

}

}

void changeLights() {

digitalWrite(carGreen, LOW); // green off

digitalWrite(carYellow, HIGH); // yellow on

delay(2000); // wait 2 seconds

digitalWrite(carYellow, LOW); // yellow off

digitalWrite(carRed, HIGH); // red on

delay(1000); // wait 1 second till its safe

digitalWrite(pedRed, LOW); // ped red off

digitalWrite(pedGreen, HIGH); // ped green on

delay(crossTime); // wait for preset time period

// flash the ped green

for (int x=0; x<10; x++) {

digitalWrite(pedGreen, HIGH);

delay(250);

digitalWrite(pedGreen, LOW);

delay(250);

}

// turn led red on

digitalWrite(pedRed, HIGH);

delay(500);

digitalWrite(carYellow, HIGH); // Yellow will switch on

digitalWrite(carRed, LOW); // red will switch off

delay(1000);

digitalWrite(carGreen, HIGH);

digitalWrite(carYellow, LOW); // Yellow will switch off

// record the time since last change of lights

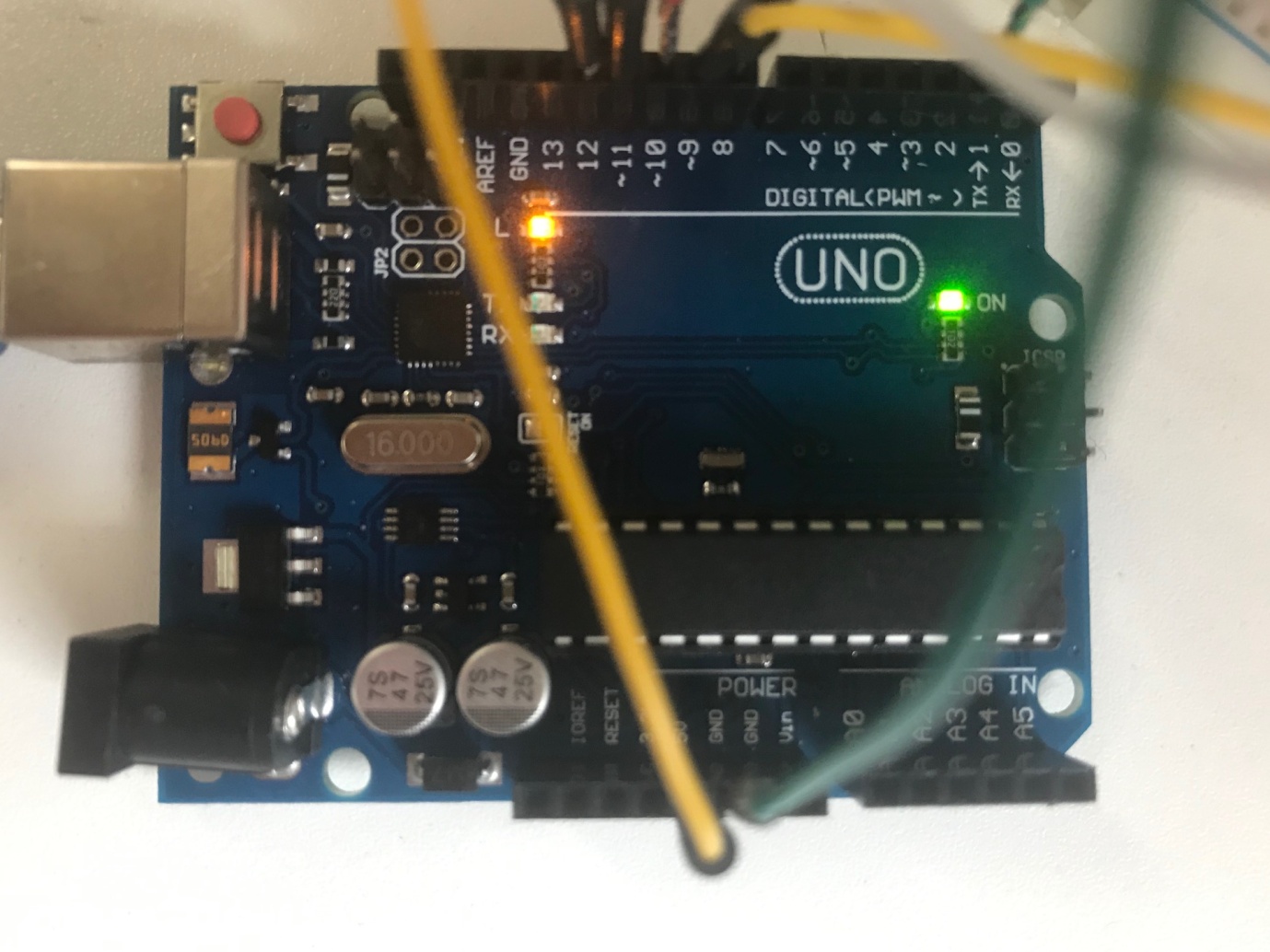
changeTime = millis();

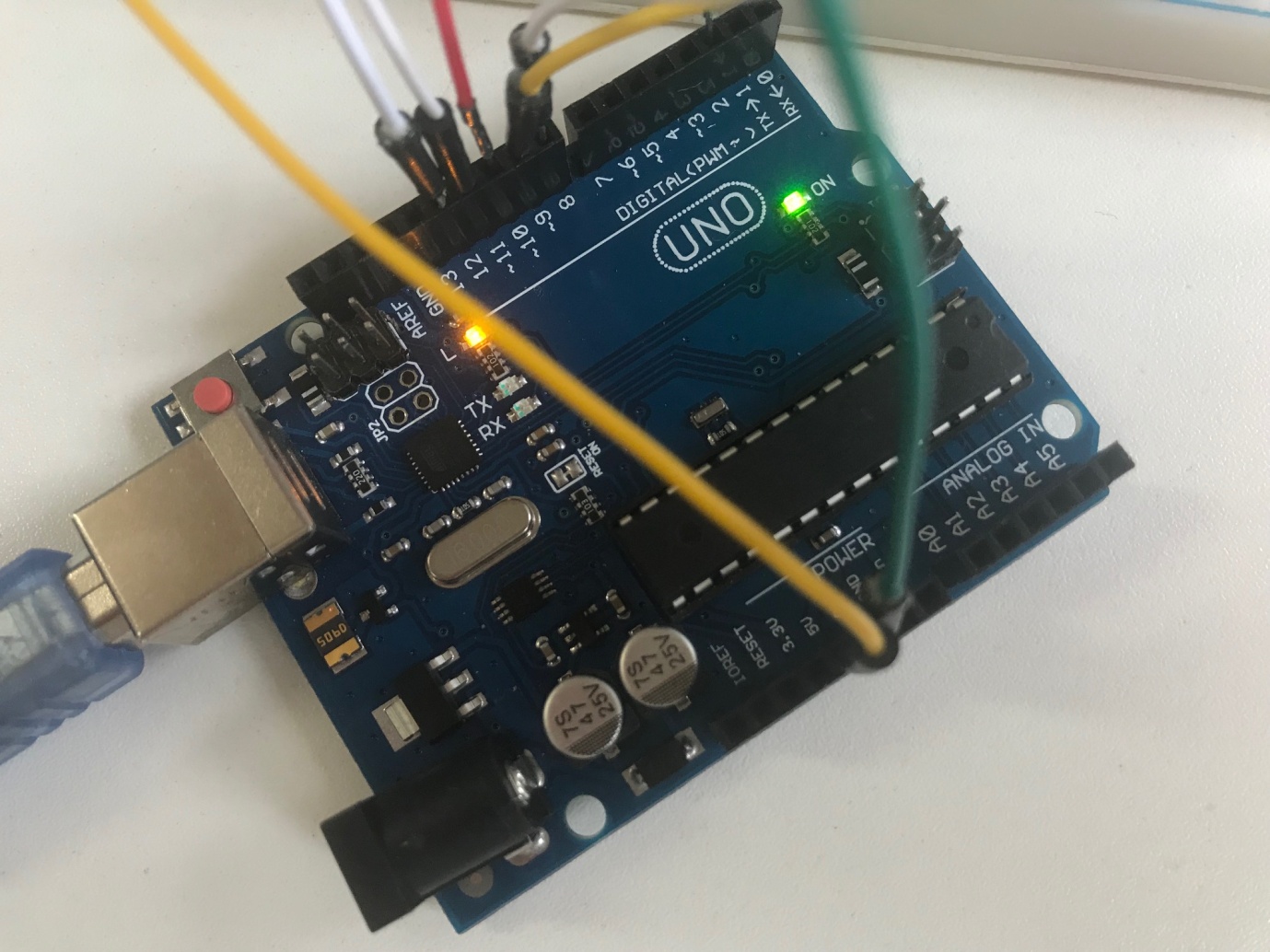
// Return / Loop

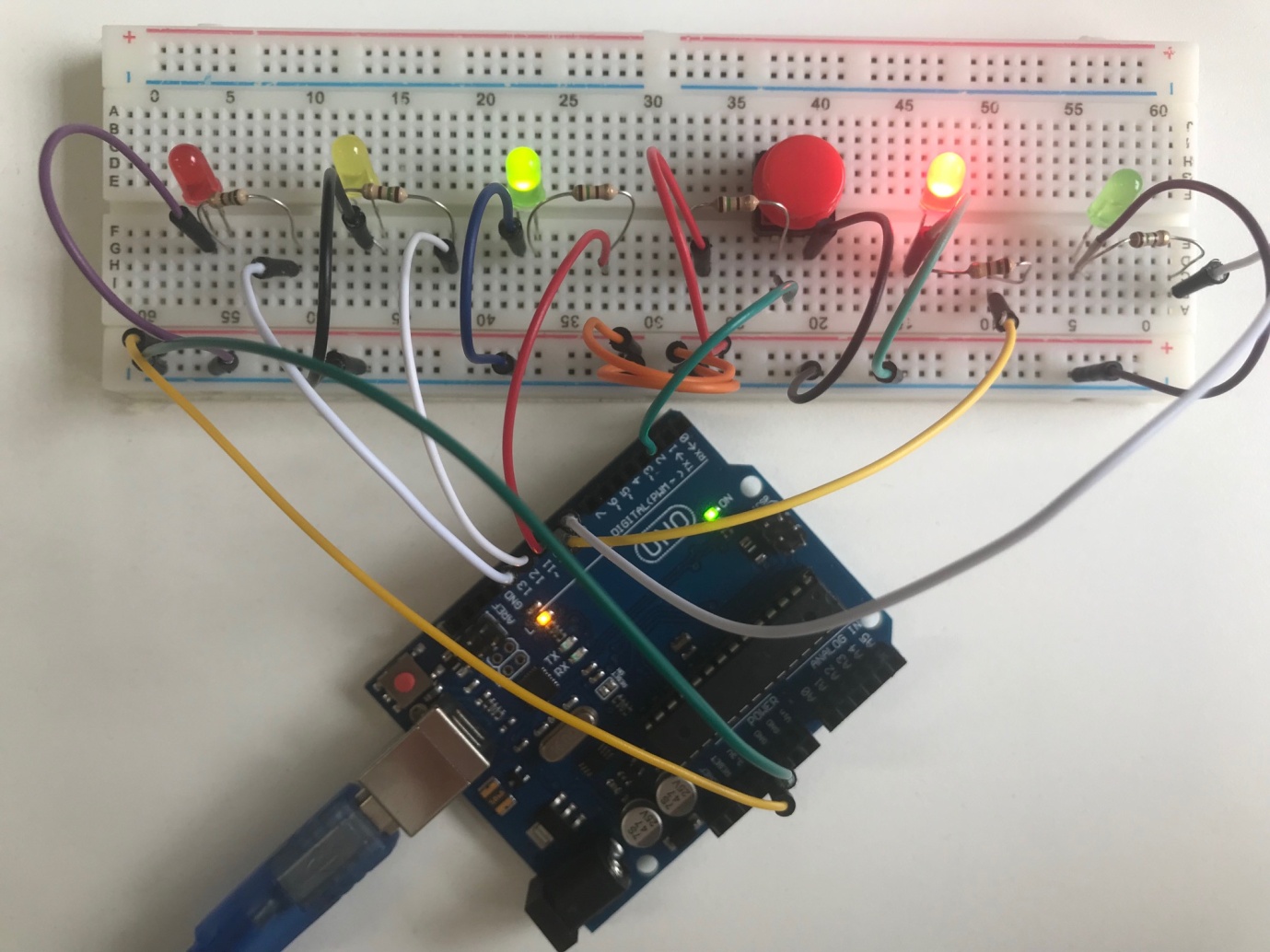
}

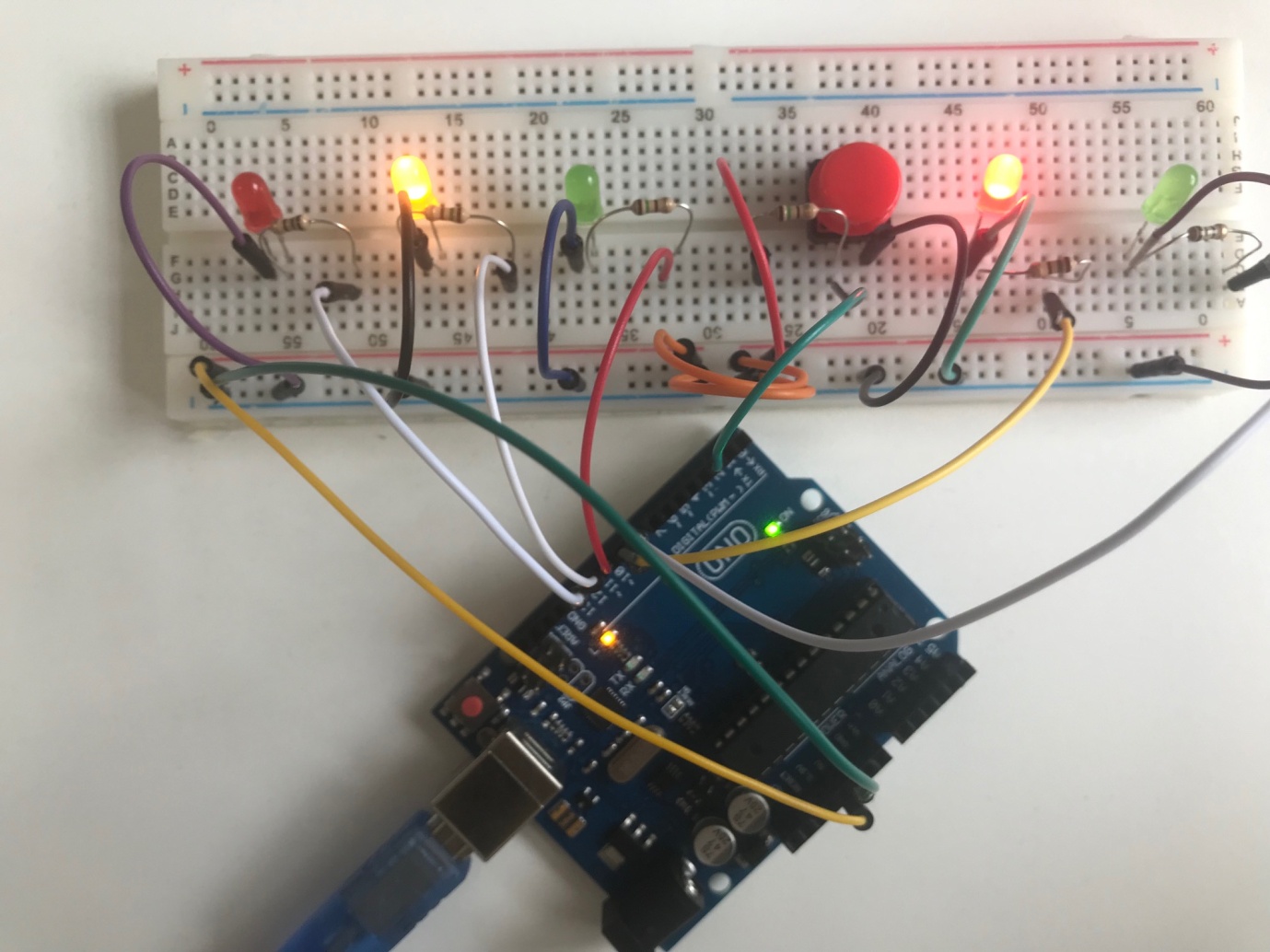


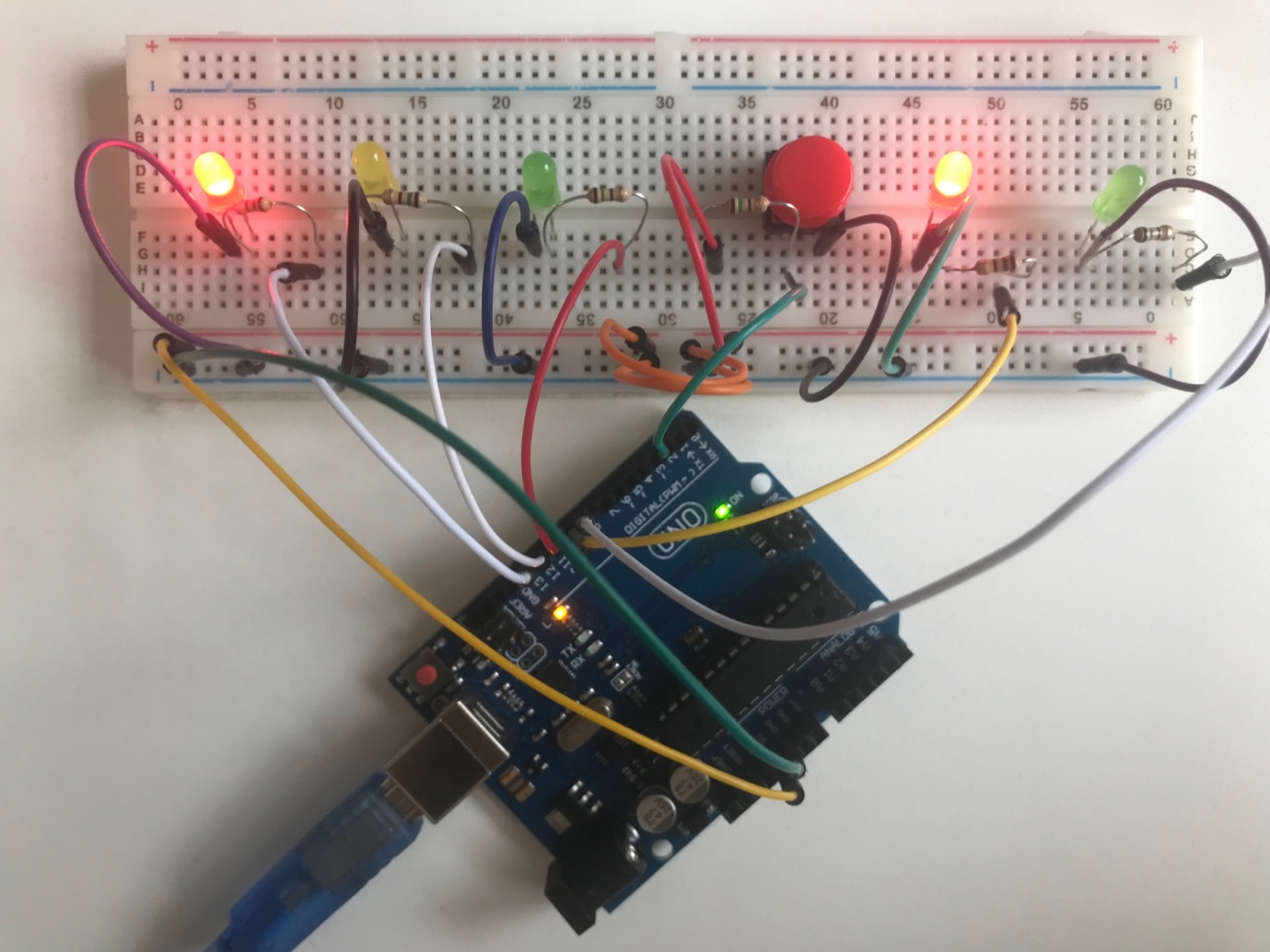
double click to view movie of the traffic lights in action^

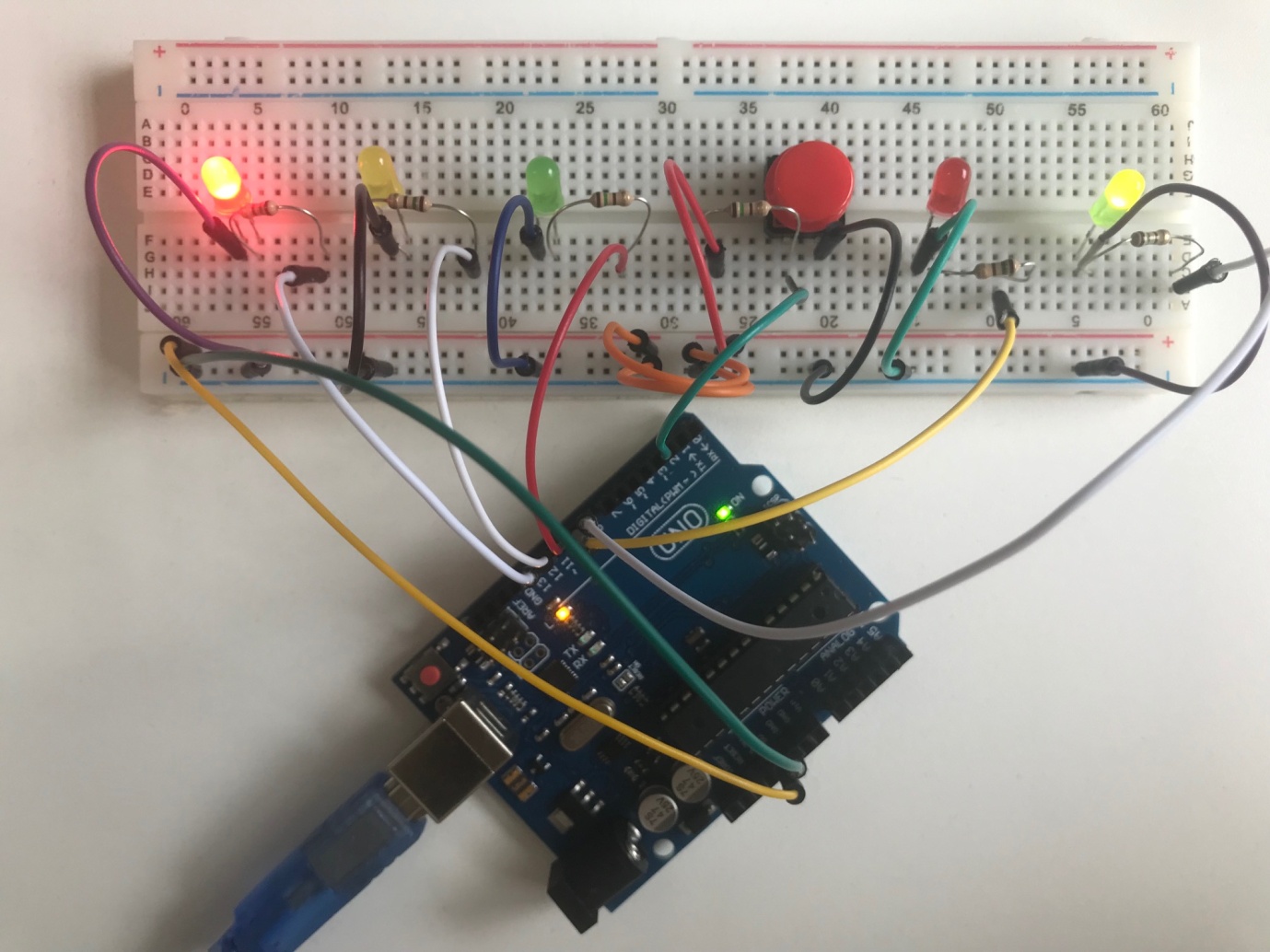


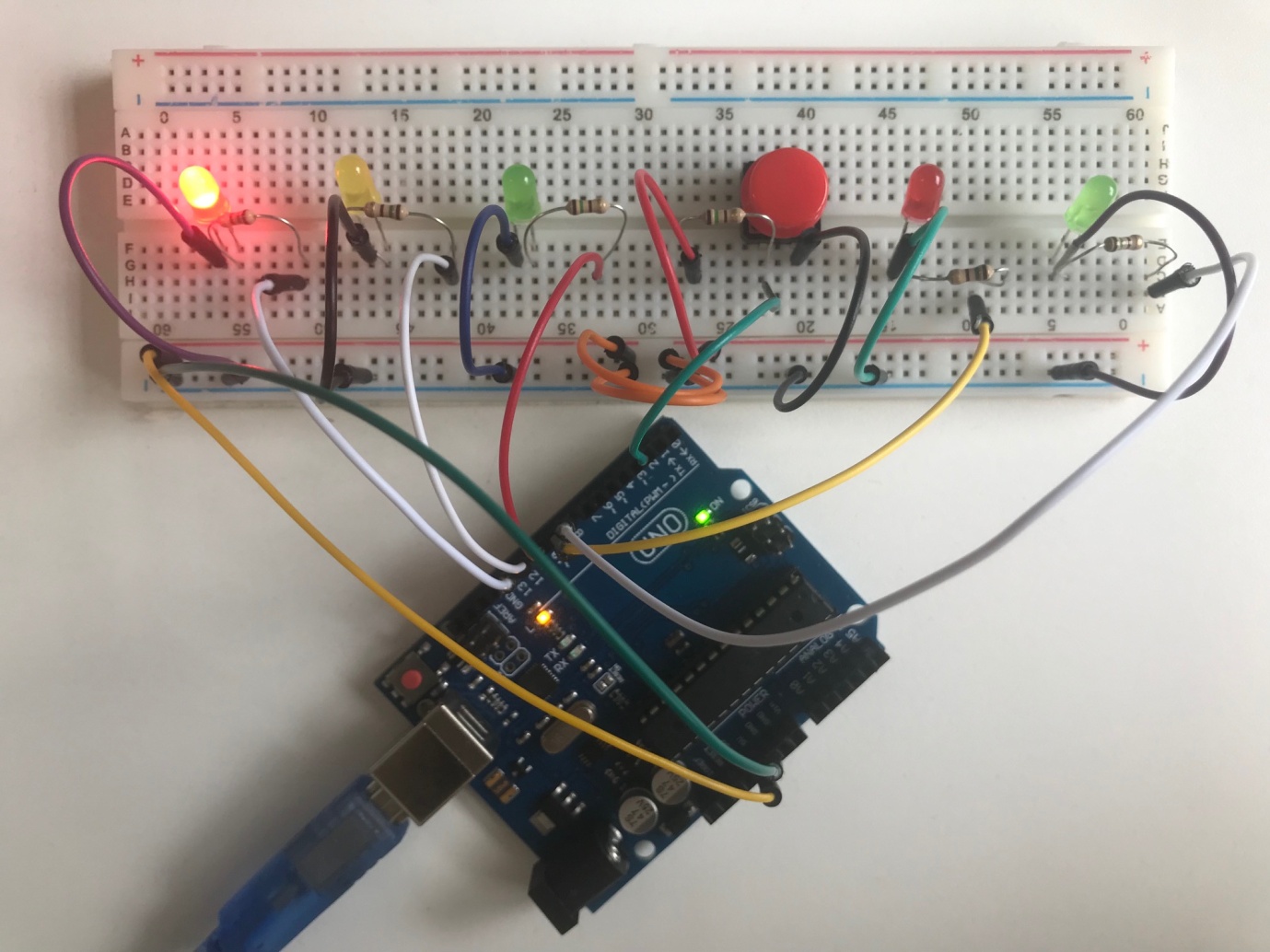




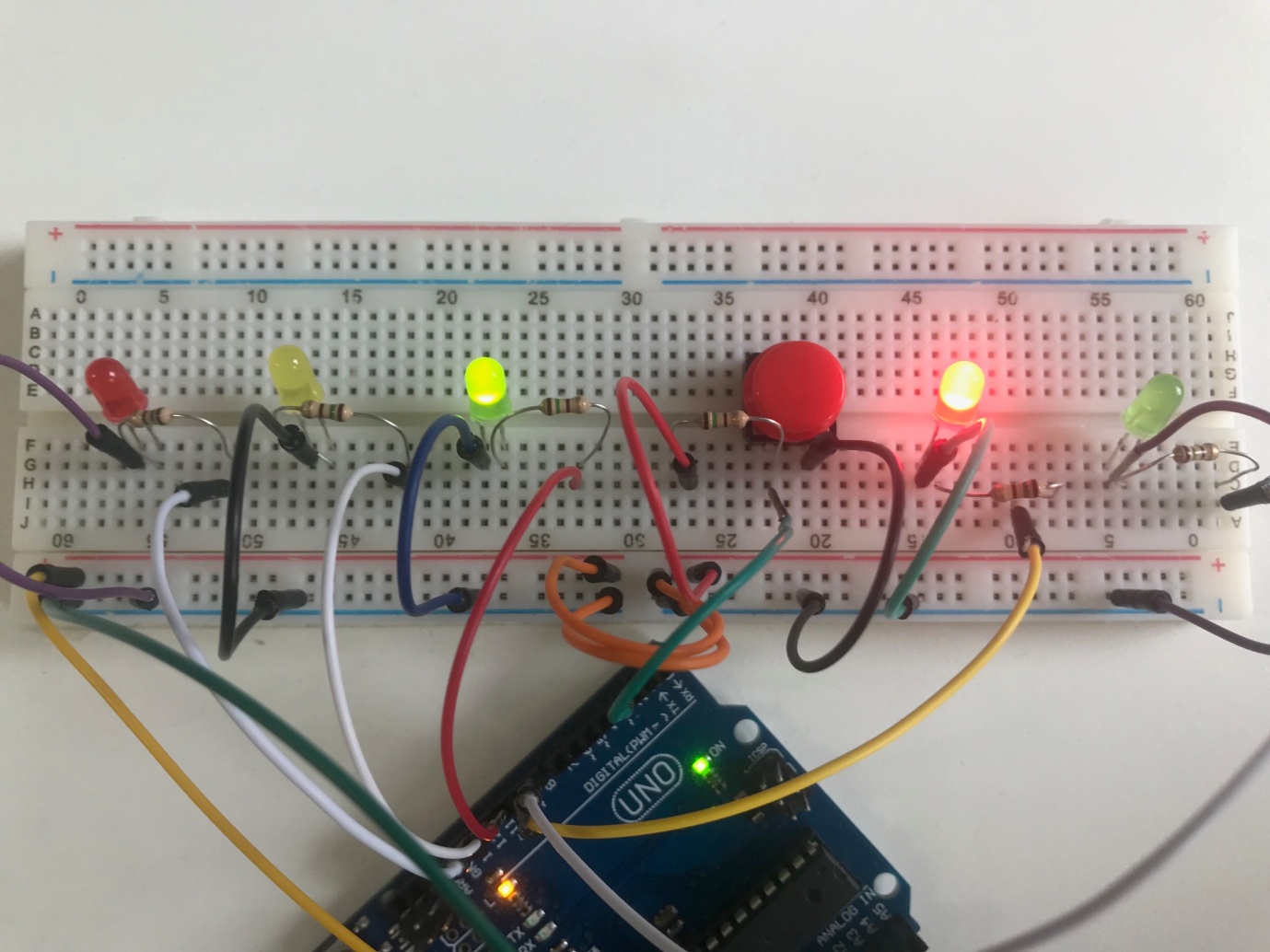












**Discussion**

This interactive traffic lights embedded system was tested by constructing the circuit with the Arduino using the Fritzing design I'd constructed earlier. It was tested visually by means of constructing the circuit together and seeing if it worked consistently when pushing the button, whether all LED's were working in a harmonious fashion. It was tested several times, and different lengths of times of holding the push button to waiver what kind of outcome would occur.

After testing the functioning of the program and the embedded system of the traffic lights I came across a problem. The problem with the original code, for it to work it had to have the button being pressed at exactly the time you got to a five second gap. And if it wasn't happening at precisely that time it wouldn't be read. So what we've done is each press of the button we've saved it to a variable state, which has to be a global variable so that it doesn't get changed each time you go round. If the state is high (which means the button is unpressed) but we find that the button here is being pressed then we make the state high. What makes it high again is having taken an action from the button being pressed by the pedestrians.

The way in which I'd initially wired the LED's and Arduino together the LED's were not initially working. After a little adjusting and making my way round the circuit, I found the source that was preventing it from working, and re-assembled the circuit resulting it to work. It works harmoniously and synergistically to result in an effective interactive traffic lights for both cars and pedestrians. Each step helps the next; it would not have been possible to have created this without the use of the push button actuator, nor the resistors, and every component of the circuit.

**Conclusion**

In this mini project we learned and explored how to create an interactive embedded system for traffic lights for both cars and pedestrians, as well as utilising the C programming language in which to accomplish it. We learned how to effectively control this circuit of LED lights by connecting them to specific ports on the Arduino, and calling them within the Arduino programming code. In addition we learned how to make use of actuators within the embedded system, and how they proved to be a vital part in ensuring the traffic lights worked harmoniously.

This traffic lights mini embedded project could be further improved by implementing a crossroads traffic lights, so a four-way traffic lights, each with their own corresponding pedestrian lights. Whilst it sounds complicated, I'm confident it would be easy to set up as I have the basics sorted. It would make for a nice challenge.

**References**

Amazon. (2018). *Arduino Uno.* [Online]. <https://www.amazon.com/Arduino-Uno/dp/B0044X2E5S>. [Accessed 29 April 2019].

Components 101. (2018). *Arduino Uno.* [Online]. <https://components101.com/microcontrollers/arduino-uno>. [Accessed 14 May 2019].