Pedestrian Traffic signal system

Mohana Greeshma Gudhimalla (.002)

Date: December 10, 2021

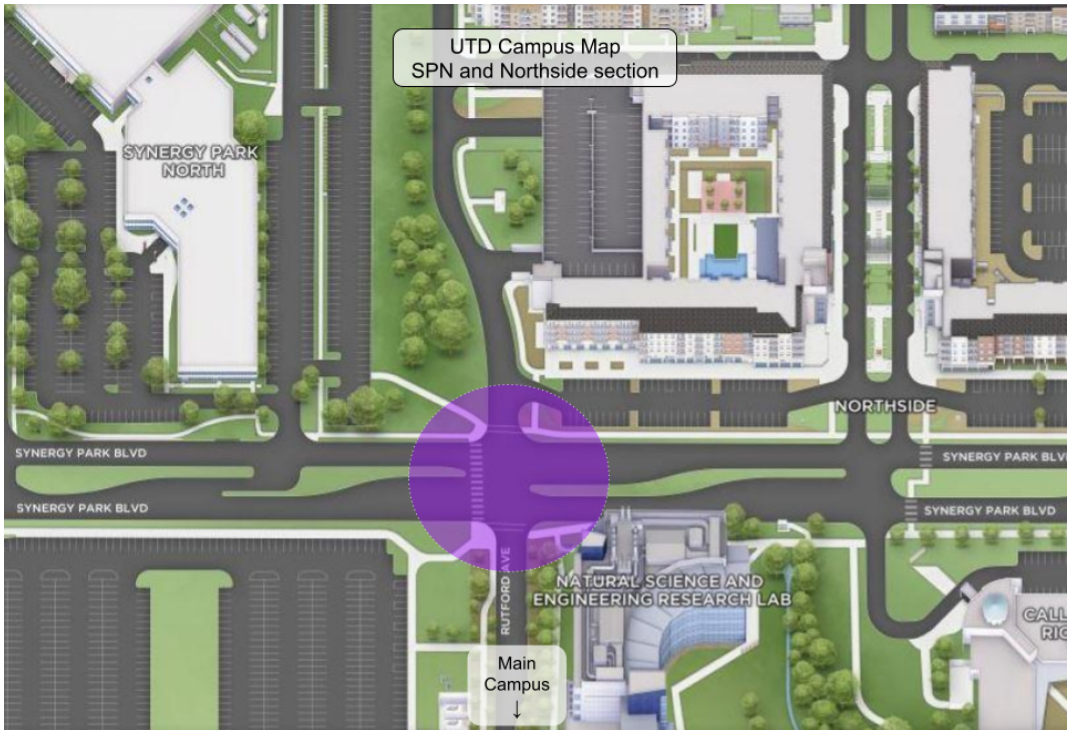
Abstract

This lab shows a short demonstration of how a one-way traffic system works with a crosswalk and its signal light, which allows pedestrians to walk across the road. we used the Arduino IDE to develop and perform my base code on how the lights work, and I implemented the system in TinkerCAD to build a program and test running it.

Introduction

This project is a development based on our previous traffic light project, which controlled two sets of lights at an intersection. For this project, we modified the other design to include one set of lights rather than two, and we incorporated a button and light system to replicate the button of a crosswalk signal.

Figure 1



Our motivation for this project was to replicate a traffic light scenario we experience often. The light and crosswalk situation is based on the walking route on campus, between the SPN building and the main campus. To take this route, we walk across Synergy Park Blvd, which is a main road controlled by a set of traffic lights. This specific section of the road is the crosswalk running parallel to Rutford Ave shown in Figure 1, highlighted by the purple circle.

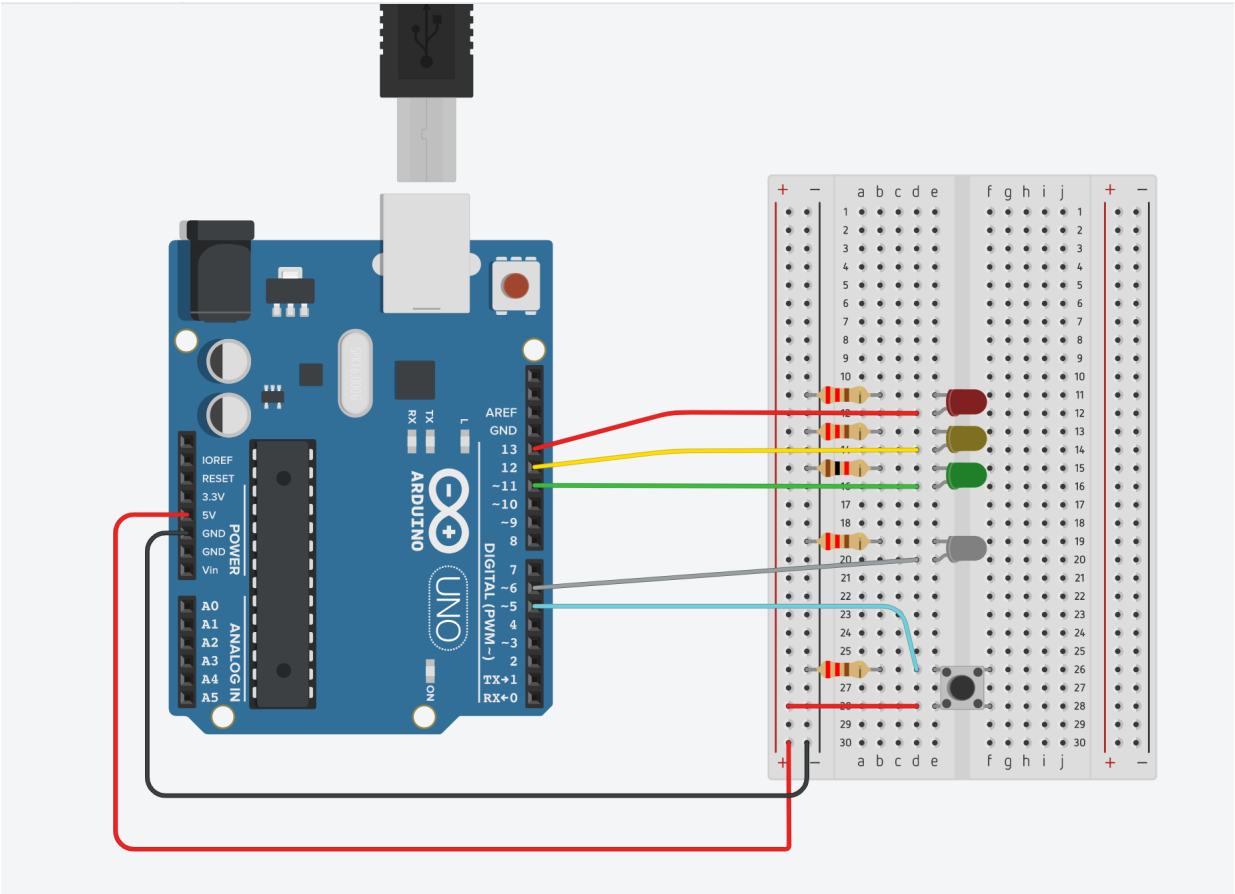
With this project, we expected the general operation to replicate the operation of both the traffic lights and the signal light when someone presses it to cross the road. When the button is pressed, the signal LED should light up, similar to the walk signal given by the green walking person. At the same time, the main traffic light should turn red and remain that way for the duration of the crosswalk signal's operation, then resume the loop that controls cars on the same road.

Design and Analysis

In this lab, we are trying to design a one-way traffic light that includes a horizontal pedestrian light. We have developed a code that performs this exercise. In this code, the one-way traffic lights light up for a specific amount of time in a loop that goes from green to yellow to red and then again to green. If the pedestrian walking light is pressed a couple of times, the red light lights up in the one-way traffic and the pedestrian walking light lights up at the same time. After a couple of seconds, the pedestrian light turns off. The red light still stays on another two seconds to give some extra time for the remaining pedestrians to cross the road. After that, the green light lights up, resuming the loop of one-way traffic lights.

The pin number that connects the button to the Arduino is pin number 5. The number that connects pedestrian streetlight is pin number 6. Red light connects to pin 13, yellow light is connected to pin 12 and green light connects to pin 11. When the void loop starts, the green lights up for 2 seconds. It then switches to yellow light for 2 seconds and then it switches to red light for 2 seconds. This loop keeps on continuing again and again. If the pedestrian LED is very low and the button is pressed, the if statement gets activated. In this if statement, we have stated how when pedestrian led gets light up, the red light lights up as well. There is a 5 second delay for pedestrian light and then it turns off. The red-light delays for 7 seconds and then turn off. When both LEDs turn off, the change of lights gets triggered, activating the traffic lights.

Figure 2 The setup of one-way traffic light with pedestrian traffic light.



In this TinkerCAD presentation, we need 4 resistors, one red LED, one green LED, one yellow LED, one LED that is any other color, here we have blue LED. We also need one button and an Arduino. The 5v Arduino supply should be connected to the positive side of the terminal that connects the entire breadboard. The ground should be connected to the negative side of the breadboard. The cathode of red LED is connected to the resistor which is connected to the ground. In a similar manner, yellow light and green light's cathodes should be connected to the resistor. The anode of the red LED is connected to pin number 13. The anode of yellow and green light is connected to 12 and 11 respectively.

Coming to the pedestrian LED, the cathode is connected to the resistor which is connected to the ground from the negative terminal. The anode is connected to pin 6. The button is connected in such a way where a resistor is placed on one side, it can be connected with either side of the button. With the resistor, there is a wire that connects the button and Arduino that is on the same side of the resistor, as we can see in the picture above. The other side of the button is connected to the 5v supply positive terminal. A TinkerCAD model of the overall setup can be seen in Figure 2 above.

Results

In the lab, we were able to set up the hardware and have all lights operate as intended. We based the hardware, shown in Figure 3, on the TinkerCAD model in Figure 2. The setup includes the crosswalk signal light and its pushbutton, and the traffic lights that loop normally and switch to red when the button is pressed. In our Arduino setup, we used red, yellow, and green LEDs for the traffic lights and a blue LED for the crosswalk signal.

We tested the button multiple times, and it is successful in both turning the blue signal LED on and triggering the loop to stop and switch the red LED on. The light system is shown in Figures 4 and 5, from two different angles, where the button has been pressed and the red LED and signal LED are both on.

While the overall operation was successful, we did find that the button needed to be pressed multiple times or held down for a second or two before it would trigger the intended actions. However, once the loop was triggered, the signal system would work properly for each test we did.

Figure 3

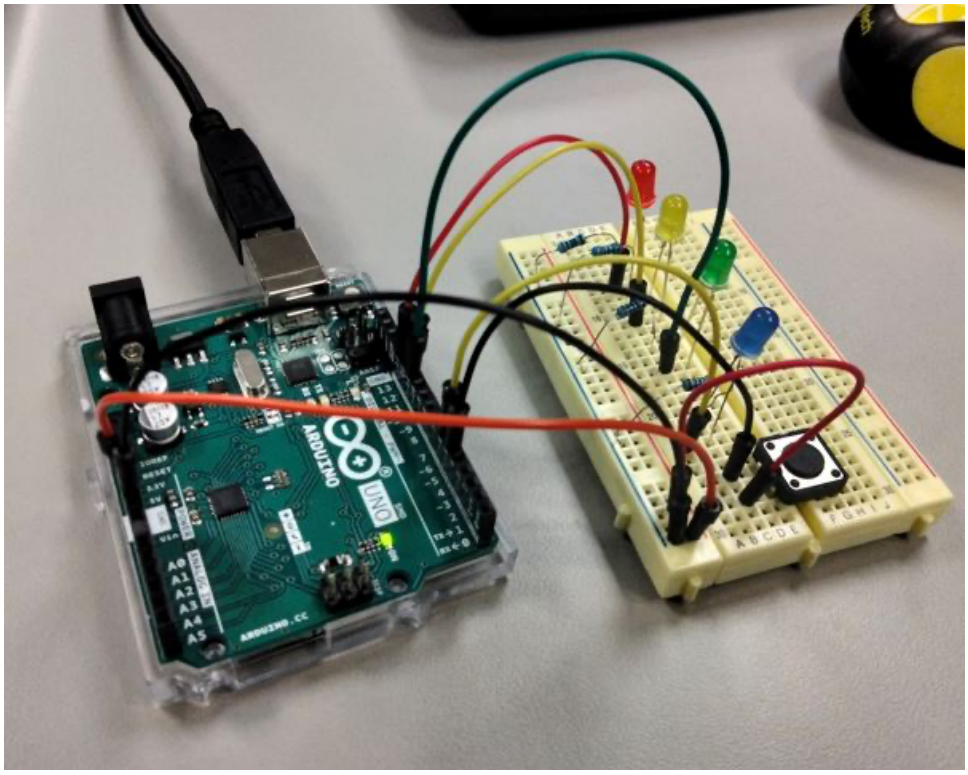


Figure 4

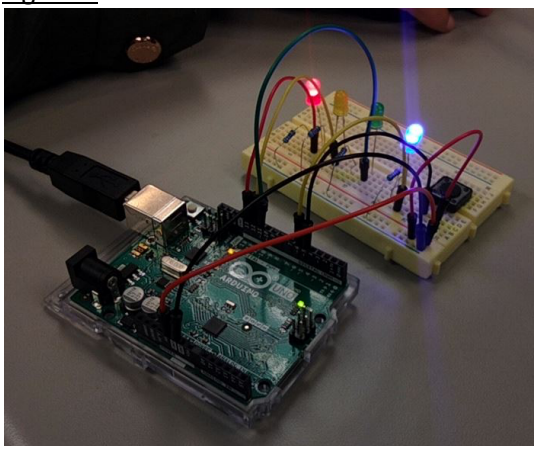
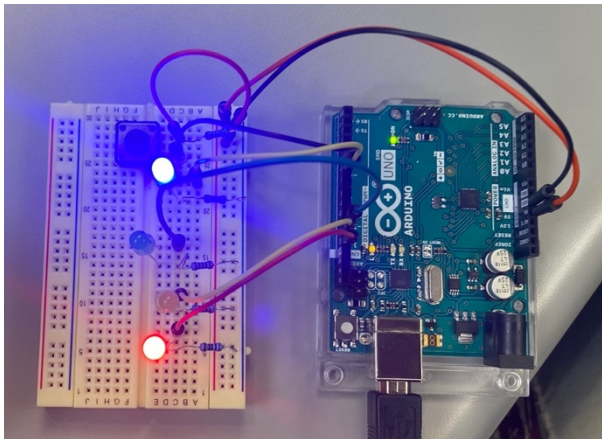


Figure 5



Discussion and Conclusions

Our project was successful overall. The lights operated in the loop as intended and the signal system worked to break the loop and turn on the red and blue LEDs when the button was pushed.

For improvements, one main change we would work on would be to consistently keep the red LED on for a longer duration after the blue LED turns off. Since this project simulates traffic lights and a crosswalk, this would be the equivalent of the crossing signal switching to red and indicating that nobody should cross, then the traffic light stayed red for an extra moment before resuming the loop. This would give an added layer of safety to the system, since it would make sure that pedestrians receive the signal not to cross before traffic starts flowing again.

Another improvement would be to modify the button operation to trigger the signal after one press, rather than having to press and hold it. This is an issue we had with our setup, and it would take some modification of the code and retesting the hardware to isolate the issue and resolve it.

For the intentions of our project, this does also simulate the real-world operation of the signal to cross Synergy Park Blvd. The button pedestrians use to signal on this road also sometimes take multiple presses for the system to recognize that we are waiting to use the crosswalk. In addition to refining the system we created, these improvements would have a benefit to the real-world situation that inspired our project.

We also considered what would make our project more realistic, if we used this system and refined it to better suit the replication of Synergy Park Blvd's crosswalk. One main addition we would have to make would be ways to control traffic coming perpendicular to the main road. In Figure 1, this would be the stretch of Rutford Ave from Northside to the main campus. Considering the traffic from these roads would be essential to demonstrate the full scenario of using the crosswalk we simulated; traffic on this road is less direct than Synergy Park Blvd in relation to the crosswalk, but anyone who turns from these areas into the main road would need to be factored in when considering what cars need to be controlled for pedestrians to cross safely.

References/Appendices

Final Complete Code:

```
//Code written by Greeshma Gudhimalla:

int pin = 5;  // button pin number
int led = 6; // pedestrian led pin number

void setup()
 {

  pinMode(LED_BUILTIN, OUTPUT); // output green light

  pinMode(12,OUTPUT);     // output yellow light

  pinMode(11, OUTPUT);       // output green light

  pinMode(pin, INPUT_PULLUP);      // input button state

  pinMode(led, OUTPUT);      //output pedestrian led light

}

long offAt = 0;
 void loop()
{
   if (digitalRead (pin) == LOW)  // if the button is turned off or not pressed, the traffic lights continue to
work.

  {

 digitalWrite(11, HIGH);

delay(2000);

digitalWrite(12, HIGH);

digitalWrite(11, LOW);

delay(2000);

digitalWrite(12,LOW);

digitalWrite(LED_BUILTIN, HIGH);

delay(2000);
```

```
digitalWrite(LED_BUILTIN, LOW);
 }

 if ((digitalRead(led) == LOW ) &&  (digitalRead(pin) == HIGH)  )
//if LED is off and button is pressed [low because it has pullup resistor]

 {

  digitalWrite(LED_BUILTIN, HIGH);

  digitalWrite(led, HIGH);

  digitalWrite(13, HIGH); //RED LIGHT HIGH

  offAt = millis() + 5000; //store var of now + 5 seconds

  delay(5000);

  digitalWrite (led, LOW);
  delay(2000);
  digitalWrite(LED_BUILTIN, LOW);

  changeLights();

 }



 else (digitalRead(led) == HIGH) ;  //if led is on and the command to turn it off after a delay of 5
seconds.

 {

    if(millis() >= offAt)   //see if it's time to turn off LED

    {
      digitalWrite(led, LOW);  //it's time. this also re-enables the button

    }
 }
}

void changeLights ()  // if the led is turns off after 5 seconds, the traffic lights resume their work and
start looping again.

{

digitalWrite(LED_BUILTIN , LOW);

digitalWrite(11, HIGH);
```

```
delay(2000);

digitalWrite(12, HIGH);

digitalWrite(11, LOW);

delay(2000);

digitalWrite(12,LOW);

digitalWrite(LED_BUILTIN, HIGH);

delay(2000);

digitalWrite(LED_BUILTIN, LOW);

}

// end of the code.
```