LAB1:

Traffic Light Controller Implementation on Arduino UNO

By: Pranati R Trivedi

ECE 442

Lab Date: 03-01-2019

Due Date: 03-08-2019

Acknowledgment: I acknowledge all of the work (including figures and codes) belongs to me and/or persons who are referenced.

Signature : Pranati Trivedi

Introduction:

Purpose: The task of this experiment is to be implement the traffic signal lights using Arduino Uno R3 board. It will be implemented on a breadboard using basic circuitry components like LEDs, jumper wires, push buttons etc. the aim of the project is to get acquainted with the basics of the Arduino microcontroller board and its IDE. The programming in Arduino is very simple that can be easily done by laymen too. It involves getting knowledge of the components, designing the algorithm for traffic lights, implementing the circuit on a bread board and also control the traffic lights according to the density of the traffic.

Scope: The limitations of this project are that it teaches the basics of Arduino and C programming language for it. But, it is limited up to that extent as it just gives the brief idea of the microcontroller board and does not cover the extremities of Arduino board. Despite that, this is a good experiment as it involves all the basic tasks and the logic implementation too.

a. Theory:

The experiment is divided into two sub-tasks

1) Simple two-way intersection

Two directions namely north/south and east/west are considered. The north/south road is comparatively busier than the east/west road. The three colored traffic lights green, yellow and red are used for directions. It is totally time-controlled and does not involve and traffic sensor. As the north/south road is busier it is given 10 time units of time while the east/west road is given 5 time units of time. A transition time of 1 unit is given by the yellow light to change from green to red.

2) Two-way intersection with traffic sensors

Due to the dependence of time in the previous method, it becomes unreliable as it may lead to longer waits for the light to green even if there no vehicles on the other road. Also, if any of the road let's say east/west has very high traffic as compared to other than 5 time unit might not be sufficient to deal this traffic. So, this method proposes an idea where the traffic density can be detected using sensors and that can help decide the flow of the traffic and accordingly assign traffic lights. Automobile sensors are placed on the side of the roads and when it gets activated, it is assumed to have traffic on that road. Then, the corresponding traffic light will turn green and allow the flow to move on. In this experiment, instead of using automobile sensors, push buttons are used. When a push button is pressed for more then 2 seconds, the corresponding light turns green.

b. Preliminary work

1. Arduino circuit to be simulate on TinkerCAD for traffic lights.

I designed the circuit on the TinkerCAD as follows-

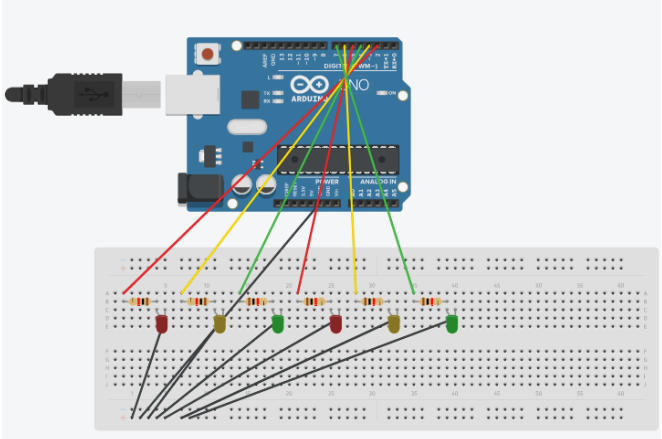


Fig.1 schematic of simple two-way intersection

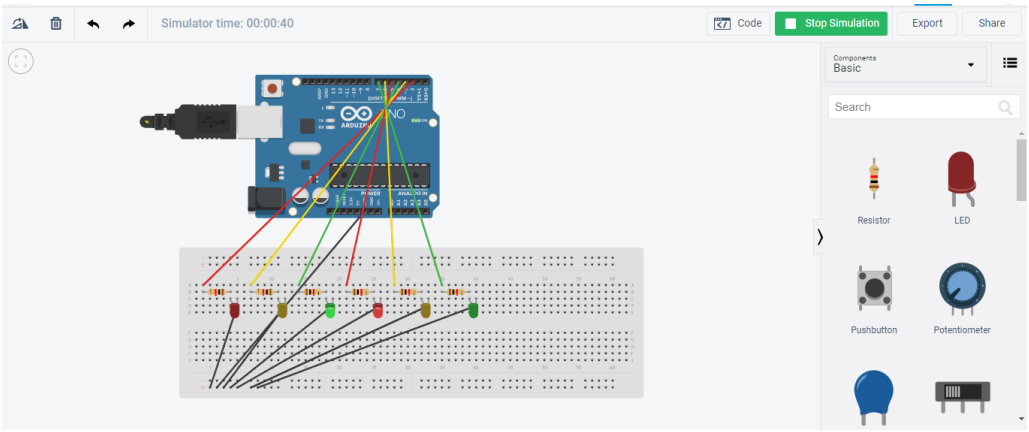


Fig.2 when north/south light is green and east/west is red

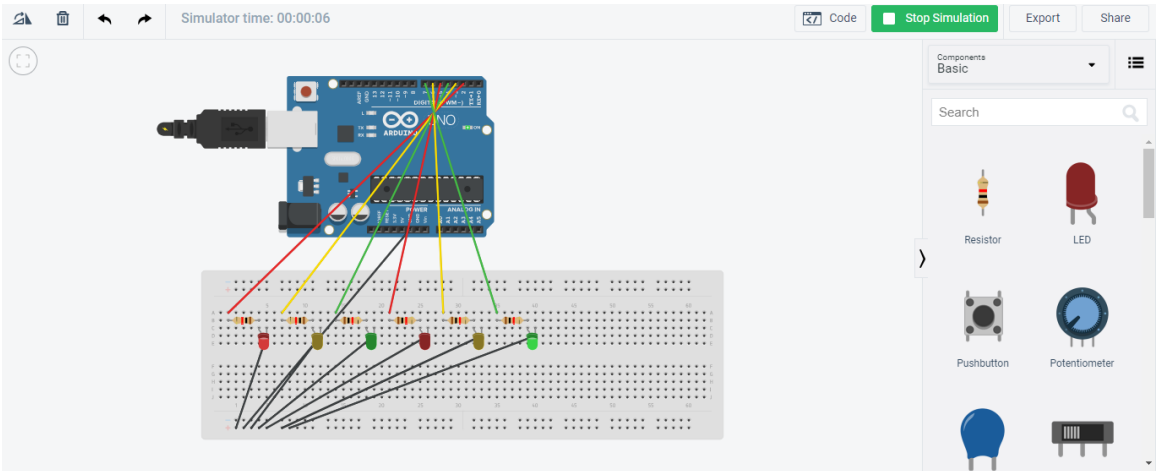


Fig.3 when north/south light is red and east/west is green

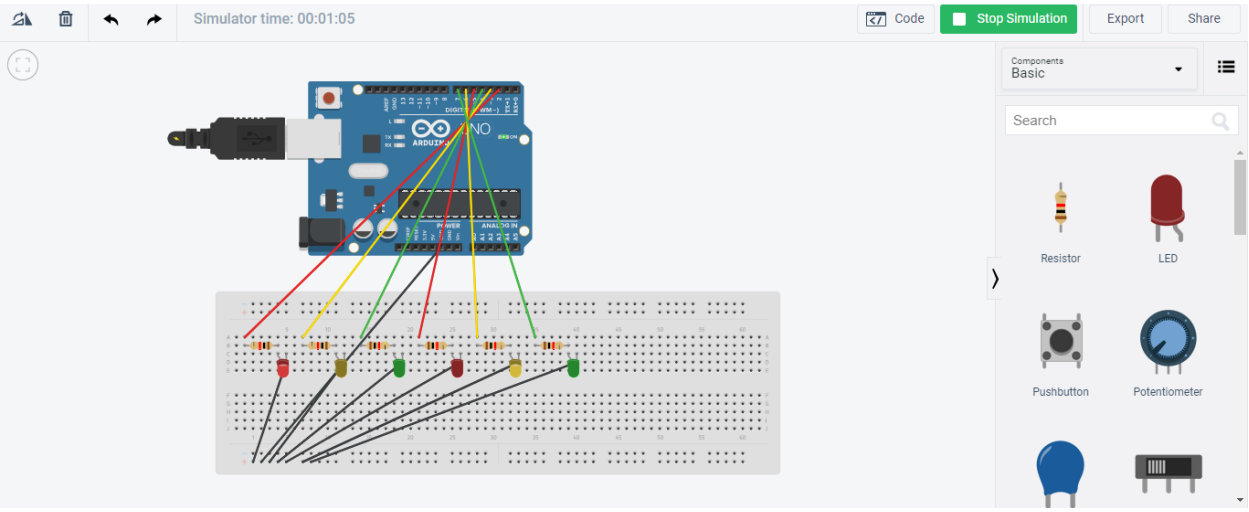


Fig.4 yellow light transition from green to red

Refer appendix A1 for the code.

2. TinkerCAD for Two-way intersection with traffic sensors

The schematic and results are as follows-

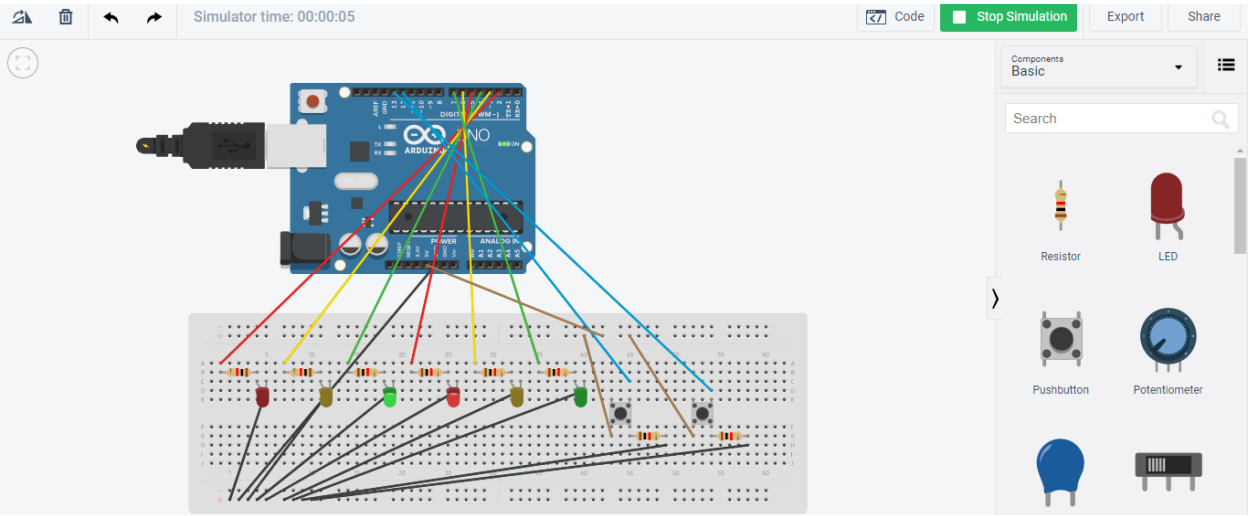


Fig.5 schematic and working of two-way intersection using push buttons

Refer to Appendix A2 for the code.

Experimental procedure

a. Schematic

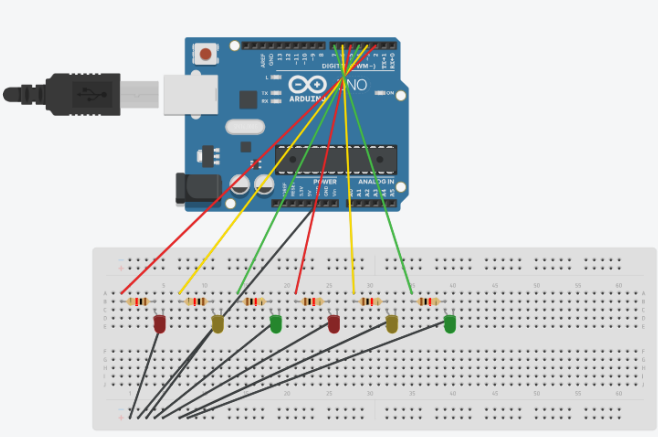


Fig.6 schematic for traffic light using Arduino

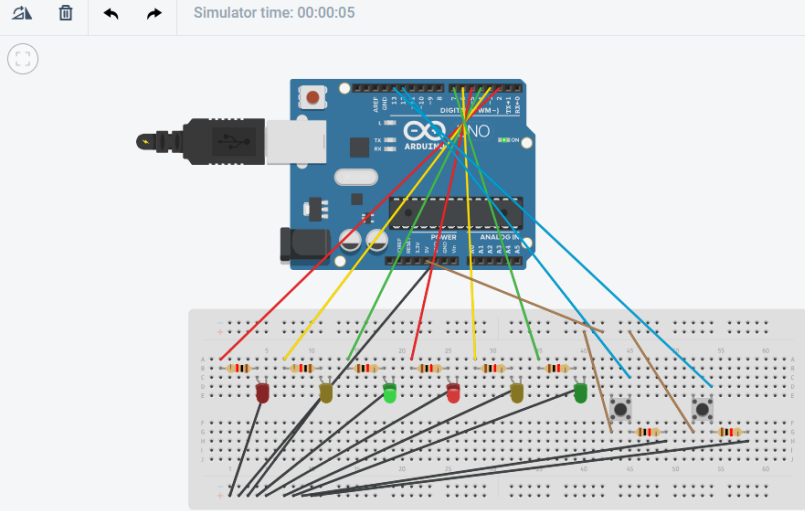


Fig.7 schematic for traffic lights using Arduino and push buttons as automobile sensors

b. Procedure:

The connection to the 6 LEDs, 2 of each red, green and yellow were done. The pin connection was as follows-

Pin 2	Red ns
Pin 3	Yellow ns
Pin 4	Green ns
Pin 5	Red ew
Pin 6	Yellow ew
Pin 7	Green ew
Push button ns	Vcc-vcc, gnd-gnd, data-pin12
Push button ew	Vcc-vcc, gnd-gnd, data-pin 13

To activate the north/south road, push button for ns is pressed for more than 2 seconds so as to detect traffic on that road. If detected than that road gets green light. Similarly, push button for east/west road was used.

c. <u>All apparatus</u>

1) Arduino Uno R3
2) Red, green, yellow LEDs
3) Resistors 1k ohm.
4) Push button
5) Jumper wires

d. <u>Results:</u>

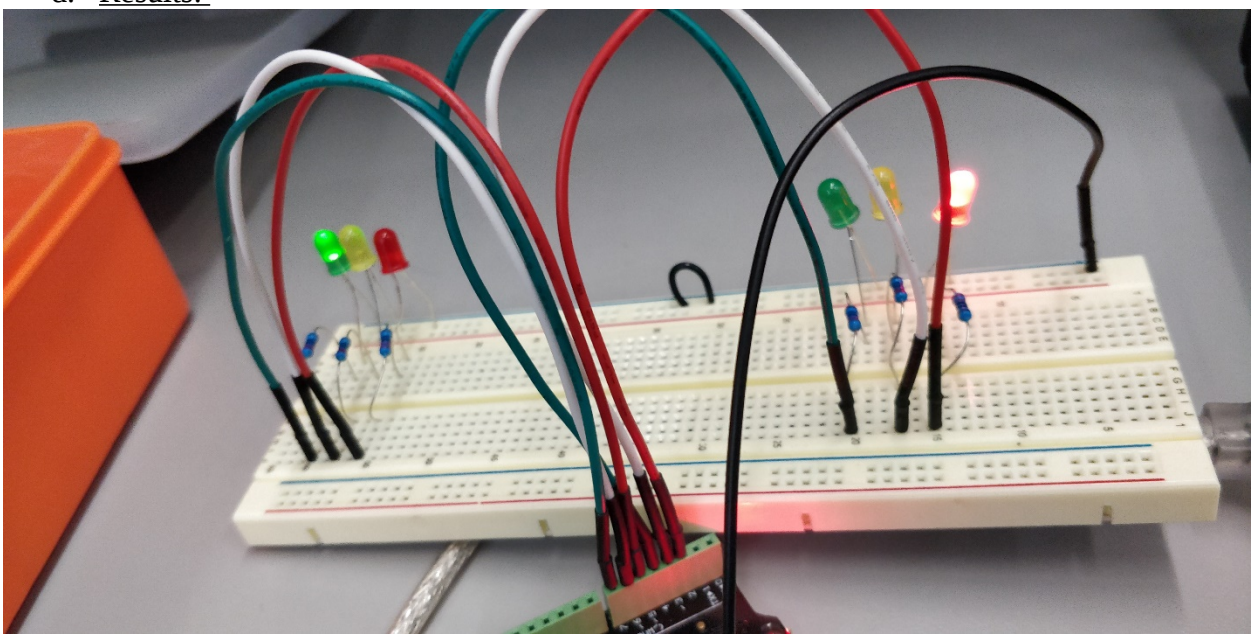


Fig.8 case when north/south light is green and east/west is red. (simple two-way)

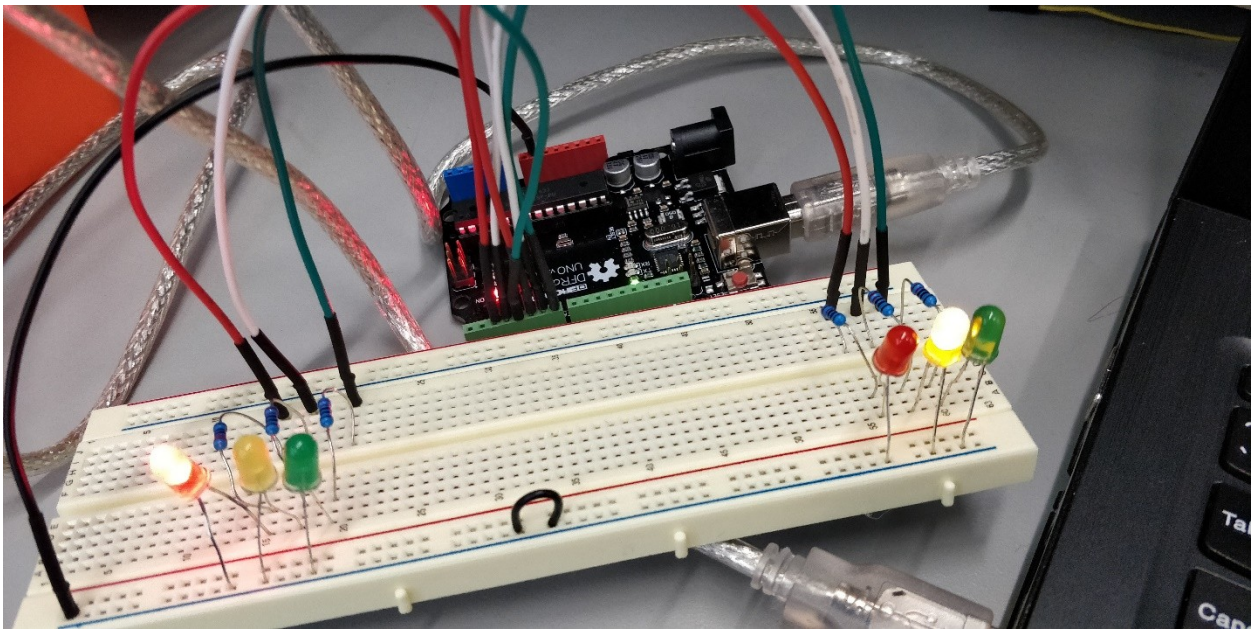


Fig.9 case of yellow transition from green to red. (simple two-way)

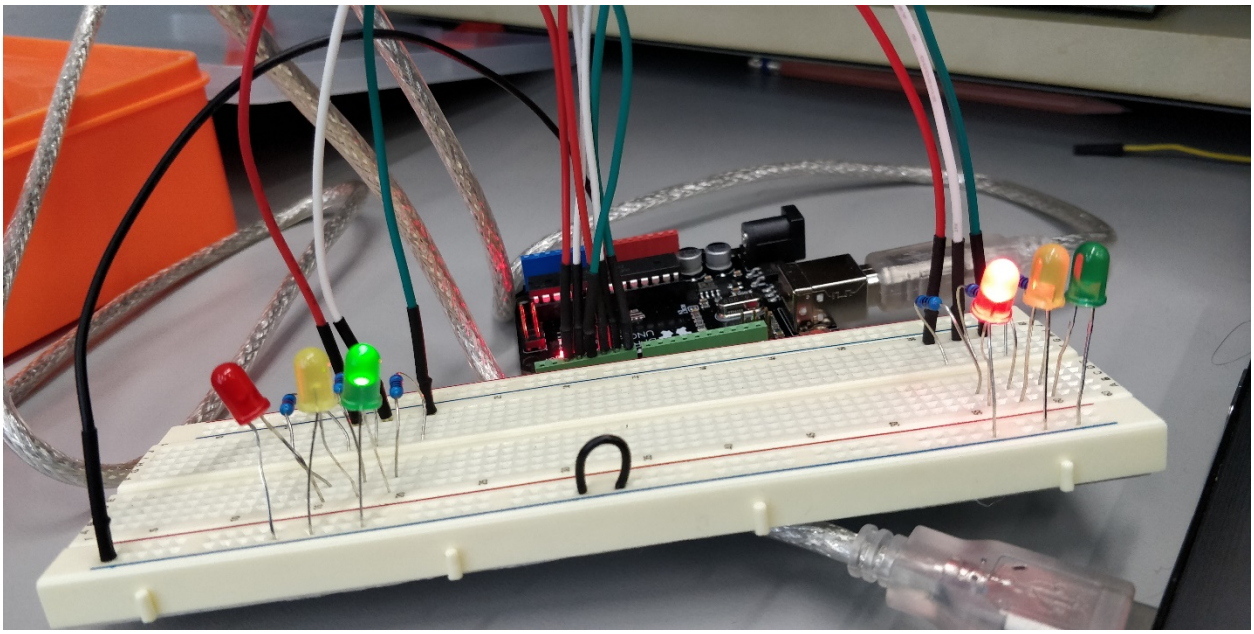


Fig.10 case when east/west light is green and north/south is red. (simple two-way)

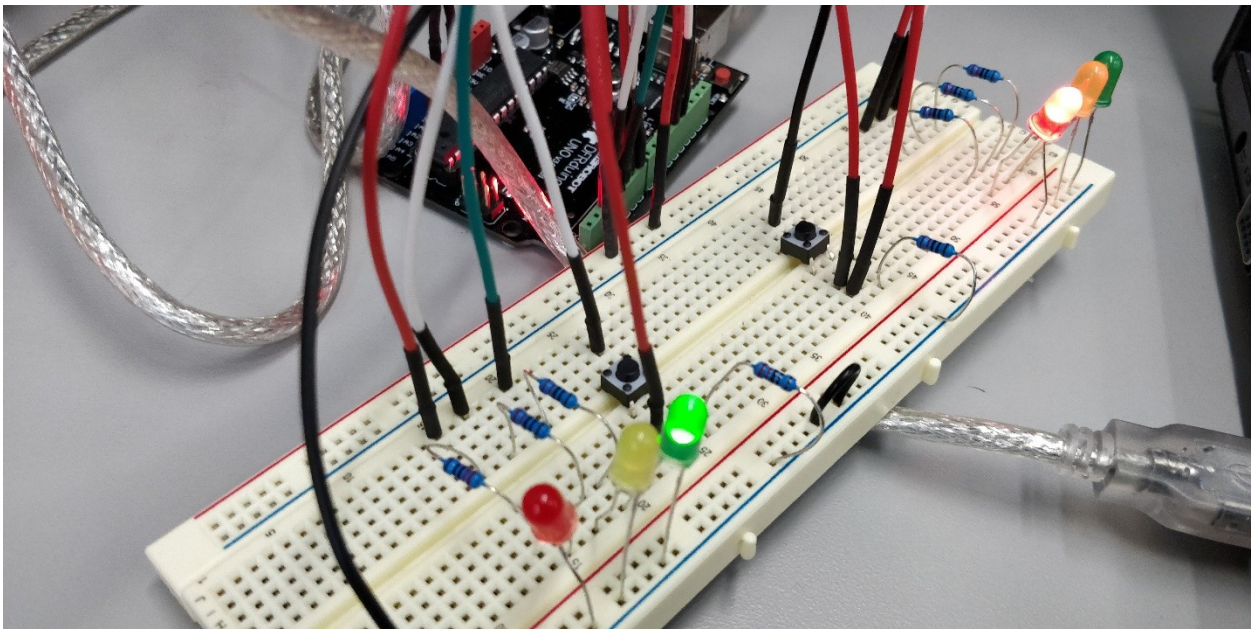


Fig.11 normal execution for two-way intersection with traffic sensors

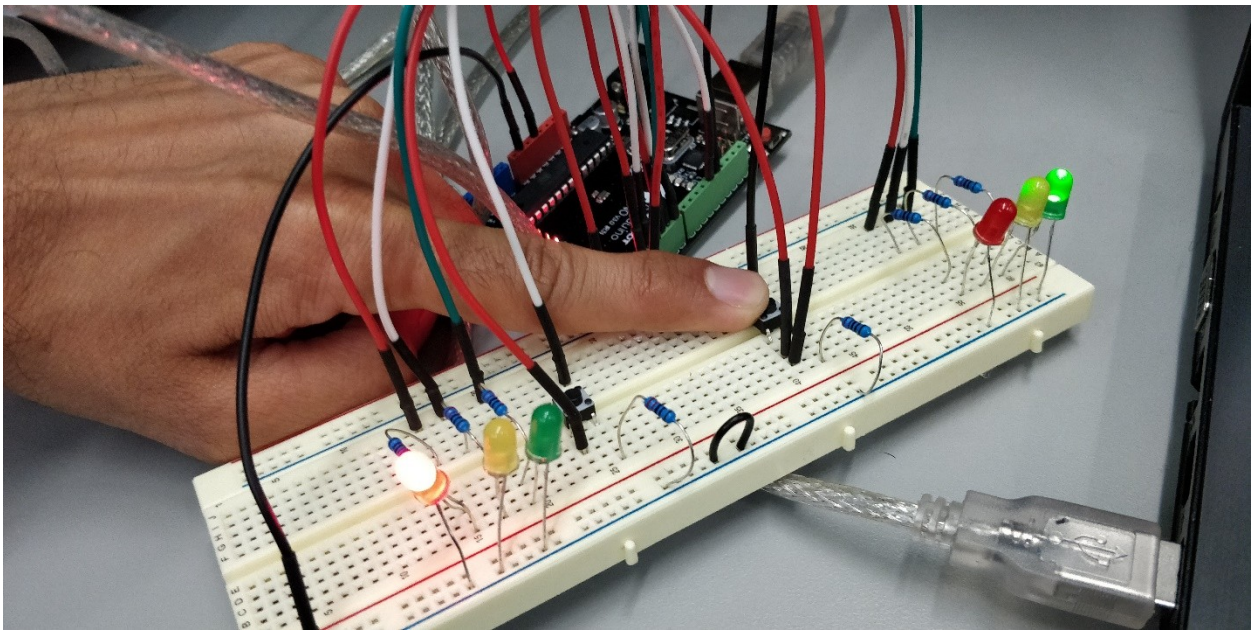


Fig.12 green for ns when north/south push button pressed for more than two seconds

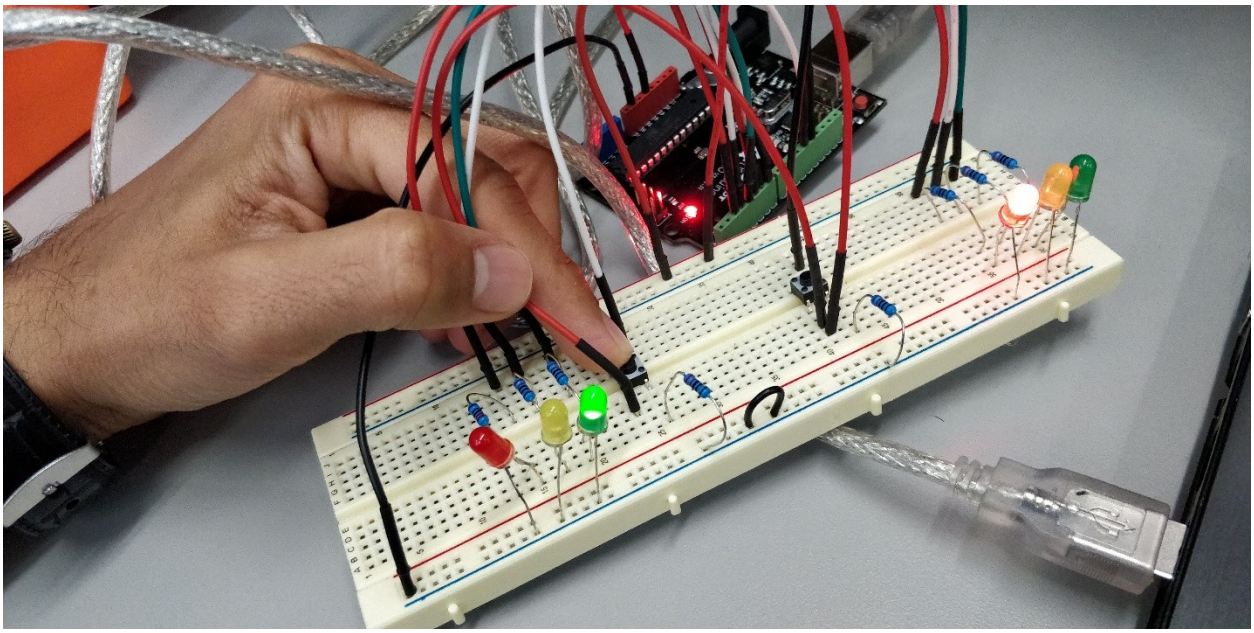


Fig.13 green for ew when east/west push button pressed for more than two seconds

e. Discussion:

1) Schematic:

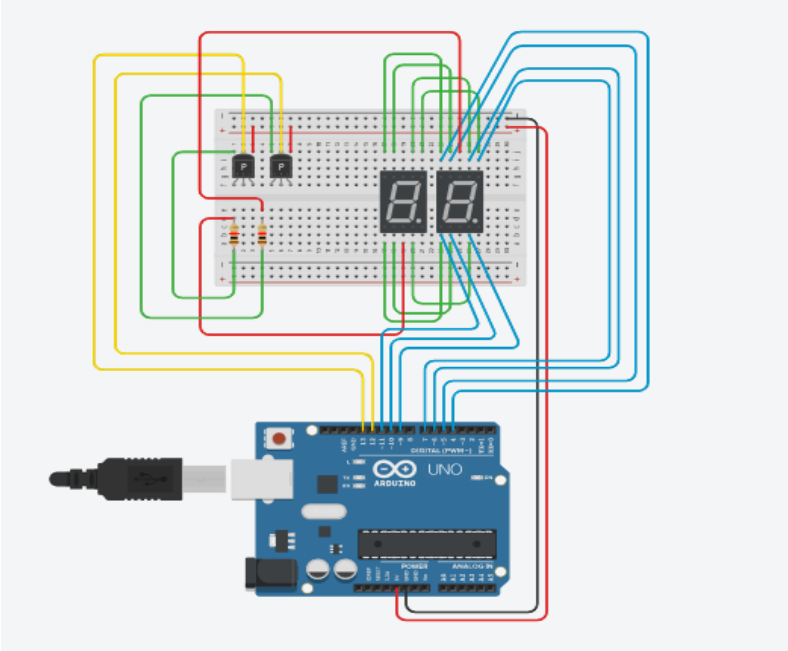
Please refer to the schematic shown above in experimental procedure (a).

Refer to appendix A1 and A2 for fully commented code

2) (a) Arduino is a microcontroller for embedded systems. It is an open source platform to carry small embedded related tasks and projects. It has several digital input-output pins, analog input-output pins. These pins can be used to get data from the sensor. This data is manipulated, and results are stored. The obtained data can be used further for processing and give output to other sensors, actuators and devices. It is a low-voltage computing system which can be used for computing low-powered sensors and devices.it has the support for SPI, I2C interfaces. Also, other support like USB, ethernet, wifi can be given to the board by attaching different shields on top of it. The programming is similar to C programming and easy too. Arduino Uno R3 has 8-bit AtMega328P microcontroller, 16 MHz clock frequency, 32 KB of flash memory and operates on 5V.

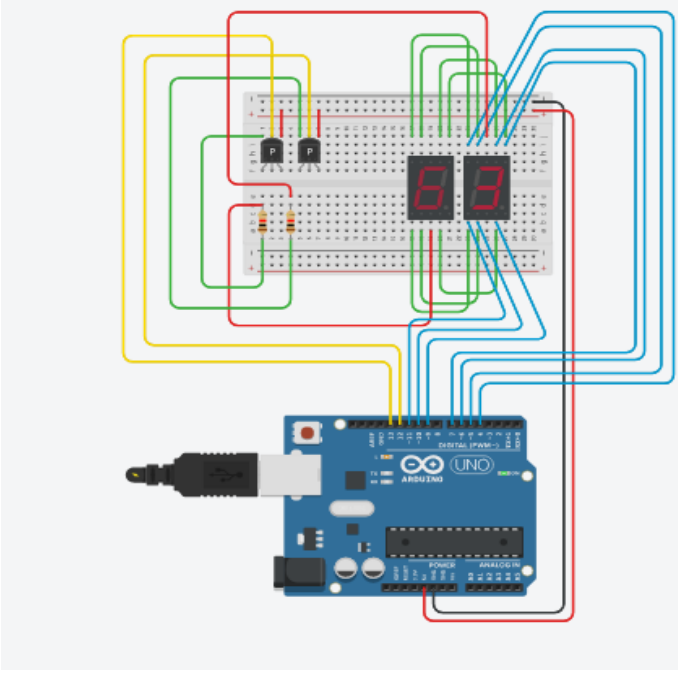
   (b) Arduino can be used for many different applications like health monitoring system, temperature controller, robotics, internet of things etc. Different sensors like temperature, pressure, infrared, ultrasonic, accelerometer, etc. can be interfaced with Arduino for making different projects. The main advantage is that it is low-powered and so all these sensors can be interfaced at such a low voltage and it becomes easy to obtain data from sensors. But, at the same time it has several limitations as it is only good for low-powered devices and cannot draw current more than 20mA. So, it cannot operate motors without the help of shield to amplify the current. Also, certain other devices like camera, internet, video are not directly supported by Arduino and it requires many shields to be interfaced with the main Arduino in order to operate it. This makes the design and computation more complex. Even the computation power is low in case of camera. It is not that suitable for real time operations. It gives a delay and cannot be relied on for real time applications.

3) Implement a system that will count from 0 to 99 and display on two 7 segment LED displays in TinkerCAD. A schematic diagram of your hardware design and a fully commented listing of the program need to be included in the lab report.



7 Segment Display

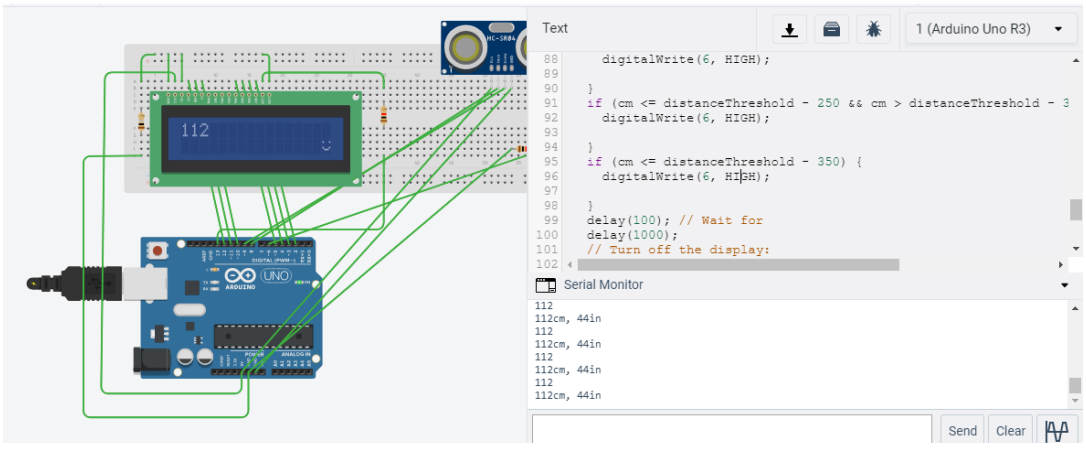
Simulator time: 00:01:09

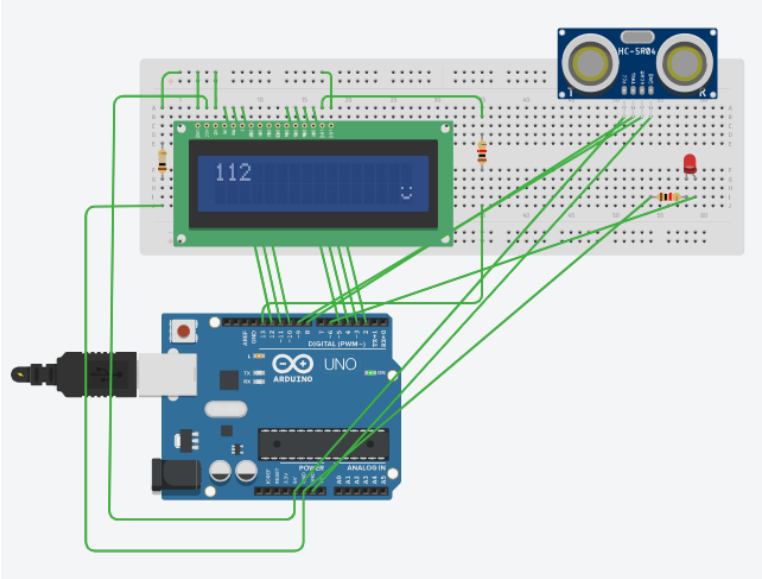


```
int a = 6;
int b = 7;
int c = 9;
int d = 10;
int e = 11;
int f = 5;
int g = 4;                                  // define name for pins

int transistors[] = {13, 12};

int leftCounter = 0;
int rightCounter = 0;

int arr[][7] = {
  {0, 0, 0, 0, 0, 0, 1},
  {1, 0, 0, 1, 1, 1, 1},
  {0, 0, 1, 0, 0, 1, 0},
  {0, 0, 0, 0, 1, 1, 0},
  {1, 0, 0, 1, 1, 0, 0},
  {0, 1, 0, 0, 1, 0, 0},
  {0, 1, 0, 0, 0, 0, 0},
  {0, 0, 0, 1, 1, 1, 1},
  {0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 1, 0, 0}
};

void setup()
{
  pinMode(a, OUTPUT);
  pinMode(b, OUTPUT);
  pinMode(c, OUTPUT);
  pinMode(d, OUTPUT);
  pinMode(e, OUTPUT);
  pinMode(f, OUTPUT);
  pinMode(g, OUTPUT);
  pinMode(transistors[0], OUTPUT);
  pinMode(transistors[1], OUTPUT);
}

void loop() {
  PraTri(leftCounter, rightCounter);
  delay(1000);
  rightCounter++;
  if (rightCounter > 9){
          rightCounter = 0;
```

```
    leftCounter++;
    if(leftCounter > 9)
      leftCounter = 0;
  }

}

void PraTri(int left, int right)
{
  digitalWrite(transistors[0], HIGH);
  digitalWrite(transistors[1], LOW);
  write(right);
  delay(100);

  digitalWrite(transistors[0], LOW);
  digitalWrite(transistors[1], HIGH);
  write(left);
}

void write(int i){
  digitalWrite(a, arr[i][0]);
  digitalWrite(b, arr[i][1]);
  digitalWrite(c, arr[i][2]);
  digitalWrite(d, arr[i][3]);
  digitalWrite(e, arr[i][4]);
  digitalWrite(f, arr[i][5]);
  digitalWrite(g, arr[i][6]);
}
```

4)





Code:

```
int distanceThreshold = 0;

int cm = 0;

int inches = 0;

long readUltrasonicDistance(int triggerPin, int echoPin)
{
  pinMode(triggerPin, OUTPUT);  // Clear the trigger
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  // Sets the trigger pin to HIGH state for 10 microseconds
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
  // Reads the echo pin, and returns the sound wave travel time in microseconds
  return pulseIn(echoPin, HIGH);
}


#include <LiquidCrystal.h>

// Connections:
// rs (LCD pin 4) to Arduino pin 12
// rw (LCD pin 5) to Arduino pin 11
// enable (LCD pin 6) to Arduino pin 10
```

```
// LCD pin 15 to Arduino pin 13
// LCD pins d4, d5, d6, d7 to Arduino pins 5, 4, 3, 2
LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2);

byte smiley[8] = {
  B00000,
  B10001,
  B00000,
  B00000,
  B10001,
  B01110,
  B00000,
};

int backLight = 13;    // pin 13 will control the backlight

void setup()
{
    pinMode(6, OUTPUT);
  pinMode(backLight, OUTPUT);
  digitalWrite(backLight, HIGH); // turn backlight on. Replace 'HIGH' with 'LOW' to turn
it off.
  lcd.begin(16,2);              // use 16 columns,2 rows for a 16x2 LCD, etc.
  lcd.clear();                  // start with a blank screen

  Serial.begin(9600);

}

void loop()
{
  // set threshold distance to activate LEDs
  distanceThreshold = 350;
  // measure the ping time in cm
  cm = 0.01723 * readUltrasonicDistance(8, 9);
  // convert to inches by dividing by 2.54
  inches = (cm / 2.54);
  Serial.print(cm);
  Serial.print("cm, ");
  Serial.print(inches);
  Serial.println("in");
  if (cm > distanceThreshold) {
    digitalWrite(6, LOW);

  }
  if (cm <= distanceThreshold && cm > distanceThreshold - 100) {
```

```
  digitalWrite(6, HIGH);

 }
 if (cm <= distanceThreshold - 100 && cm > distanceThreshold - 250) {
  digitalWrite(6, HIGH);

 }
 if (cm <= distanceThreshold - 250 && cm > distanceThreshold - 350) {
  digitalWrite(6, HIGH);

 }
 if (cm <= distanceThreshold - 350) {
  digitalWrite(6, HIGH);

 }
 delay(100); // Wait for
 delay(1000);
 // Turn off the display:
 lcd.noDisplay();
 delay(500);
 // Turn on the display:
 lcd.display();
 lcd.setCursor(0,0);          // set cursor to column 0, row 0 (the first row)
 lcd.print(cm);    // change text to whatever you like. keep it clean!

 lcd.createChar(0, smiley);
 lcd.setCursor(15, 1);
 lcd.write(byte(0));
 Serial.println(cm);
 delay(1);
}
```

Interpretation:

It can be seen from the results that the obtained outputs are much as desired. It is comparable to the results and thus are error-free.

Conclusion:

It can be concluded that using LEDs and other basic components like push buttons, a reliable code can be developed for the traffic lights. This code can be implemented using Arduino board and the desired system for traffic lights is obtained on the interfaced components. The automobile sensor can help in achieving the needs for determining traffic density and change the traffic signals for the lane accordingly. This Arduino board can work as a good and reliable support for this system.

Appendix

Appendix A1
Code for simple two-way traffic light controller

```
// North_South LEDs
#define r_ns 2
#define y_ns 3
#define g_ns 4
// East_West LEDs
#define r_ew 5
#define y_ew 6
#define g_ew 7

#define time_base 500

void setup()
{
pinMode(r_ns, OUTPUT);       // declare as output
pinMode(y_ns, OUTPUT);
pinMode(g_ns, OUTPUT);
pinMode(r_ew, OUTPUT);
pinMode(y_ew, OUTPUT);
pinMode(g_ew, OUTPUT);
}

void ChangeLedValue(byte number)      // to turn on the sequence for LEDs
{
  digitalWrite(r_ns, bitRead(number, 5));
  digitalWrite(y_ns, bitRead(number, 4));
  digitalWrite(g_ns, bitRead(number, 3));
  digitalWrite(r_ew, bitRead(number, 2));
  digitalWrite(y_ew, bitRead(number, 1));
  digitalWrite(g_ew, bitRead(number, 0));
}

void LightSequence()              // to assign sequence in which those 6 LEDs will turn on and off
{
  ChangeLedValue(B001100);         // giving north/south green for 10 unit time
  delay(time_base*10);
  ChangeLedValue(B010100);         // yellow transition to shift from green to red
  delay(time_base*1);
  ChangeLedValue(B100100);         // holding red for one unit time for both the signals
  delay(time_base*1);
  ChangeLedValue(B100001);         // giving east/west green for 5 unit time
```

```
  delay(time_base*5);
  ChangeLedValue(B100010);          // yellow transition to shift from green to red
  delay(time_base*1);
  ChangeLedValue(B100100);          //holding red for one unit time for both the signals
  delay(time_base*1);
}

void loop()
{
  LightSequence();
}
```

Appendix A2
Code for two-way intersection with traffic sensors

```
// North_South LEDs
#define r_ns 2
#define y_ns 3
#define g_ns 4
// East_West LEDs
#define r_ew 5
#define y_ew 6
#define g_ew 7

#define ns 12
#define ew 13

#define time_base 500
//Softwareserial(9600);
void setup()
{
pinMode(r_ns, OUTPUT);        // declare as output
pinMode(y_ns, OUTPUT);
pinMode(g_ns, OUTPUT);
pinMode(r_ew, OUTPUT);
pinMode(y_ew, OUTPUT);
pinMode(g_ew, OUTPUT);
pinMode(ns, INPUT);           // declare as input
pinMode(ew, INPUT);
}

void ChangeLedValue(byte number)      // to turn on the sequence for LEDs
{
  digitalWrite(r_ns, bitRead(number, 5));
  digitalWrite(y_ns, bitRead(number, 4));
  digitalWrite(g_ns, bitRead(number, 3));
  digitalWrite(r_ew, bitRead(number, 2));
  digitalWrite(y_ew, bitRead(number, 1));
  digitalWrite(g_ew, bitRead(number, 0));
}

void LightSequence_ns()             // assign sequence when north/south traffic is high
{
    ChangeLedValue(B001100);
    delay(time_base*10);

    ChangeLedValue(B010100);
```

```
   delay(time_base*1);

   ChangeLedValue(B100100);
   delay(time_base*1);
}

void LightSequence_ew()            // assign sequence when east/west traffic is high
{
   ChangeLedValue(B100001);
   delay(time_base*10);

   ChangeLedValue(B100010);
   delay(time_base*1);

   ChangeLedValue(B100100);
   delay(time_base*1);
}

void LightSequence()               // to assign sequence in which those 6 LEDs will turn on and off
{
  ChangeLedValue(B001100);         // giving north/south green for 10 unit time
  delay(time_base*10);
  ChangeLedValue(B010100);         // yellow transition to shift from green to red
  delay(time_base*1);
  ChangeLedValue(B100100);         // holding red for one unit time for both the signals
  delay(time_base*1);
  ChangeLedValue(B100001);         // giving east/west green for 5 unit time
  delay(time_base*5);
  ChangeLedValue(B100010);         // yellow transition to shift from green to red
  delay(time_base*1);
  ChangeLedValue(B100100);         //holding red for one unit time for both the signals
  delay(time_base*1);
}

void carDetect()                   // function to detect the presence of car
{

  int count=0;
  if(digitalRead(ew)==HIGH)     // condition to check if east/west sensor is activated for more
than 2 seconds
  {

    for (int i=1;i=10;i++)
    {
      int a=digitalRead(ew);
      if(a==HIGH)
```

```
      {
        count++;
      Serial.println(count);
            if(count==10)
        {
        LightSequence_ew();
        break;

        }
      }
    }

  }
    }

void loop()
{
  digitalWrite(2, HIGH);
  digitalWrite(7,HIGH);
  carDetect();

}
```