# BECE351E – INTERNET OF THINGS

**EXPT NO: 1**

**DATE: 2nd May 2023**

**NAME: Syed Urooj Ul Hasan**

**REG. NO.: 21BRS1244**

## Gas Sensor Detector with Arduino UNO in Tinker CAD

### AIM

To perform and analyze the working of the gas sensor detector with Arduino UNO in Tinker CAD.

### BLOCKS/COMPONENTS REQUIRED

Tinker CAD Block Modules

1. Arduino UNO
2. Gas Sensor

### ALGORITHM

**Step-1:** Open TinkerCAD.

**Step-2:** In the Design Suite, Drag and drop the required blocks and its associated components into their respective places.

**Step-3:** Connect as per the circuit diagram given in the block diagram. Follow the image and place each part as shown. Notice that the breadboard is used to connect the components together.

It contains a series of columns and two rails on each side. As you click from pin to pin, you'll create wires, each of which can be color-coded for easy identification.

**Step-4:** In code section, appropriate Arduino codes are fed. Create the next block, an "if" statement, which is a type of control block that makes a decision.

**Step-5:** Once you're done, select "Start Simulation" on the toolbar to turn on your Arduino Uno.

**Step-6:** Troubleshooting: If your program doesn't behave as expected, check your wiring and programming. Ensure that all pins are properly connected and that each block is written correctly.

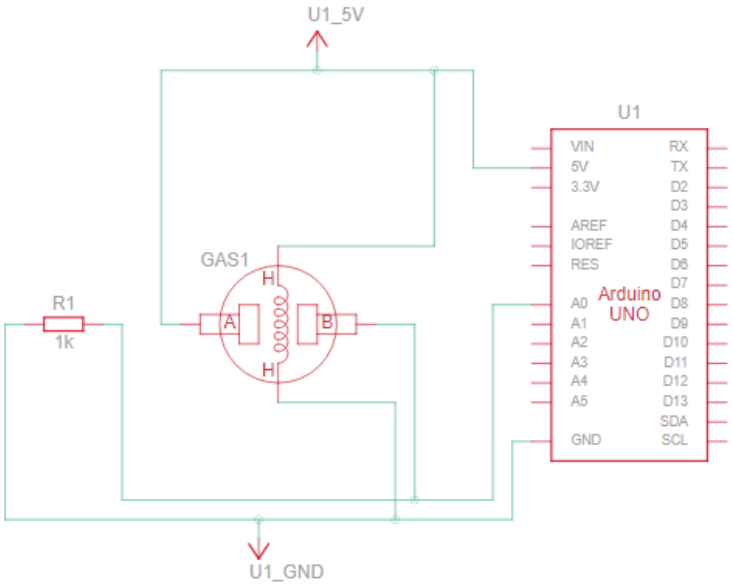
## THEORY ABOUT GAS SENSOR DETECTOR

The gas sensor has a built-in potentiometer that allows you to adjust the sensor digital output (D0) threshold. This threshold sets the value above which the digital pin will output a HIGH signal. The voltage that the sensor outputs changes accordingly to the smoke/gas level that exists in the atmosphere. The sensor outputs a voltage that is proportional to the concentration of smoke/gas. In other words, the relationship between voltage and gas concentration is the following:

- The greater the gas concentration, the greater the output voltage
- The lower the gas concentration, the lower the output voltage

The output can be an analog signal (A0) that can be read with an analog input of the Arduino or a digital output (D0) that can be read with a digital input of the Arduino.

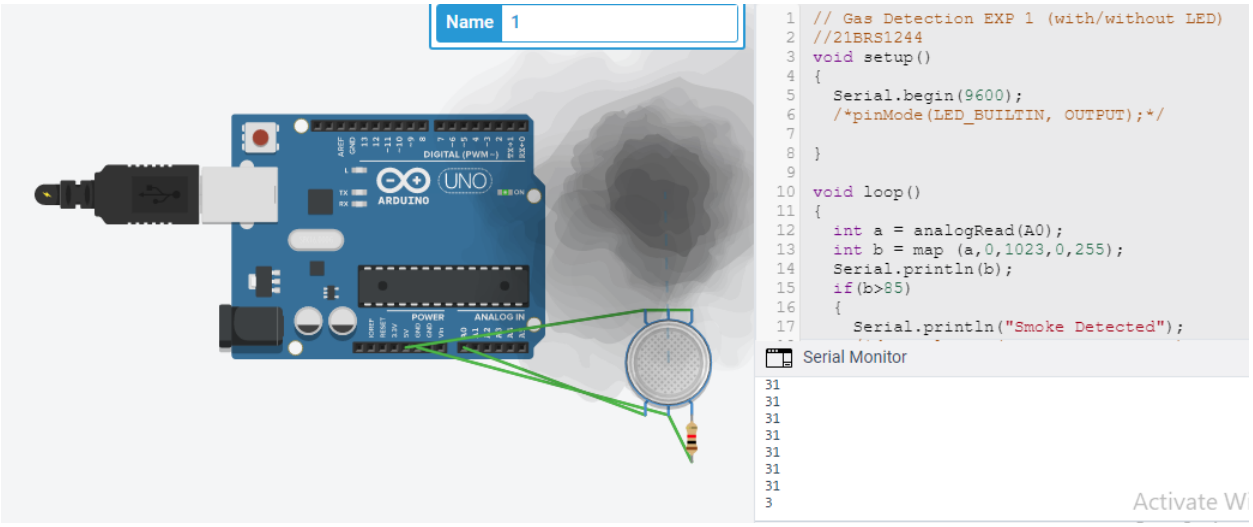
## BLOCK DIAGRAM

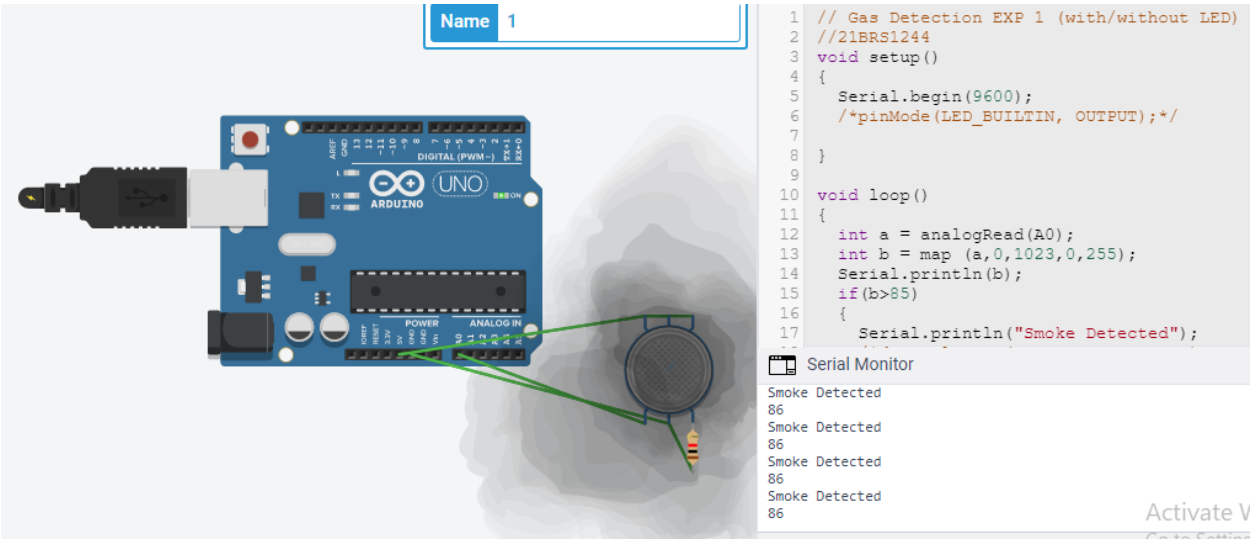



## ARDUINO PROGRAM

```
// Gas Detection EXP 1 (with/without LED)
//21BRS1244
void setup()
{
  Serial.begin(9600);
  /*pinMode(LED_BUILTIN, OUTPUT);*/

}

void loop()
{
  int a = analogRead(A0);
  int b = map (a,0,1023,0,255);
  Serial.println(b);
  if(b>85)
  {
    Serial.println("Smoke Detected");
    /*digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(1000);*/
  }
}
```

## SNAP SHOT OF THE OUTPUT




## INFERENCE

The serial monitor let us know whenever the gas sensor senses something in it's proximity.

## RESULT

The working of the gas sensor detector with Arduino UNO in Tinker CAD is analyzed and the variations in the gas detection with respect to distance is observed.

# BECE351E – INTERNET OF THINGS

**EXPT NO: 2**

**DATE: 6th May 2023**

**NAME: Syed Urooj Ul Hasan**

**REG. NO.: 21BRS1244**

## SOIL MOSITURE MONITORING WITH BUZZER USING ARDUINO UNO– Tinker CAD and

**AIM**

To perform and analyze the working of the soil moisture monitoring

(i) Using sensor with Buzzer and LED a breadboard connected with Arduino UNO in Tinker CAD.

(ii) Using Arduino Uno Hardware

**BLOCKS/COMPONENTS REQUIRED**

(i) Software:

Tinker CAD Block Modules

1. Arduino UNO
2. Soil Moisture Sensor

(ii) Hardware:

1. Arduino UNO
2. Soil Moisture Sensor

3. Sensor Interface

**THEORY ABOUT SOIL MOISTURE SENSOR**

Soil moisture is basically the content of water present in the soil. This can be measured using a soil moisture sensor which consists of two conducting probes that act as a probe. The fork-shaped probe with two exposed conductors acts as a variable resistor (similar to a potentiometer) whose resistance varies with the soil's moisture content. This resistance varies inversely with soil moisture:

The more water in the soil, the better the conductivity and the lower the resistance.

The less water in the soil, the lower the conductivity and thus the higher the resistance.

The sensor produces an output voltage according to the resistance, which by measuring we can determine the soil moisture level. It can measure the moisture content in the soil based on the change in resistance between the two conducting plates. The resistance between the two conducting plates varies in an inverse manner with the amount of moisture present in the soil.

**(i) USING TINKER CAD:**

**ALGORITHM**

**Step-1:** Open TinkerCAD

**Step-2:** In the Design Suite, Drag and drop the required blocks and its associated components into their respective places.

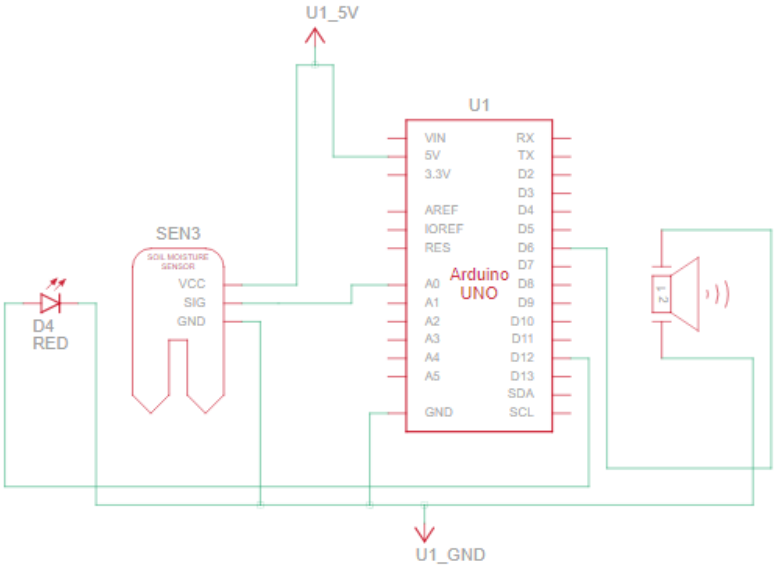
**Step-3:** Connect as per the circuit diagram given in the block diagram. Follow the image and place each part as shown. Notice that the breadboard is used to connect the components together. It contains a series of columns and two rails on each side. As you click from pin to pin, you'll create wires, each of which can be color-coded for easy identification.

**Step-4:** In code section, appropriate Arduino codes are fed. Create the next block, an "if" statement, which is a type of control block that makes a decision.

**Step-5:** Once you're done, select "Start Simulation" on the toolbar to turn on your Arduino Uno.

**Step-6:** Troubleshooting: If your program doesn't behave as expected, check your wiring and programming. Ensure that all pins are properly connected and that each block is written correctly.

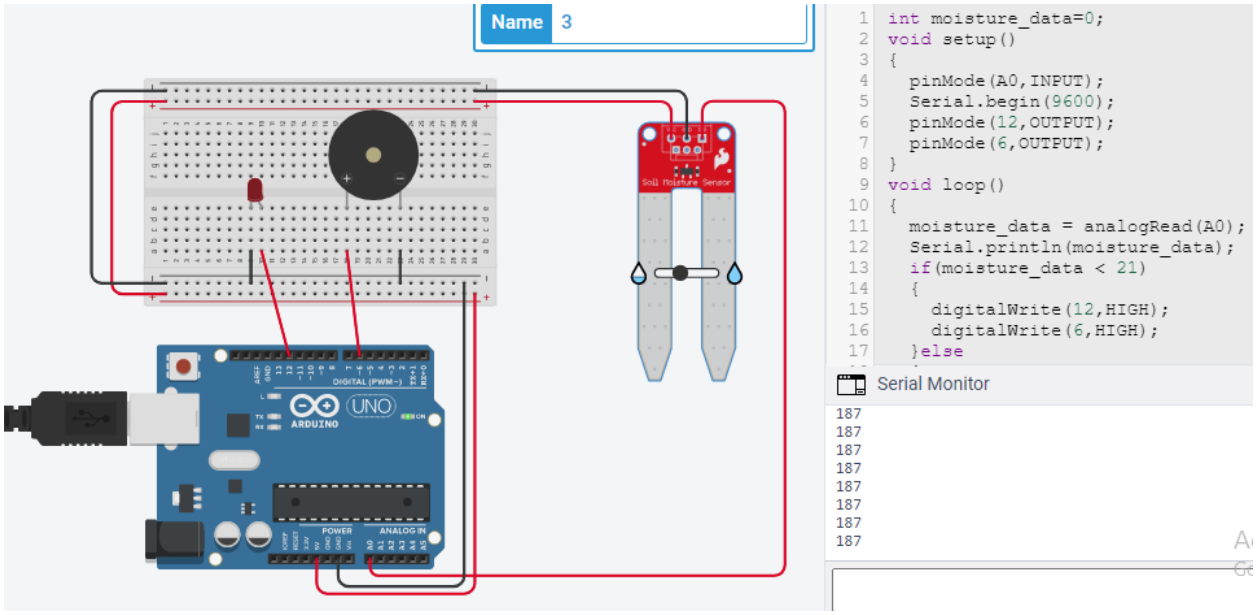
## BLOCK DIAGRAM

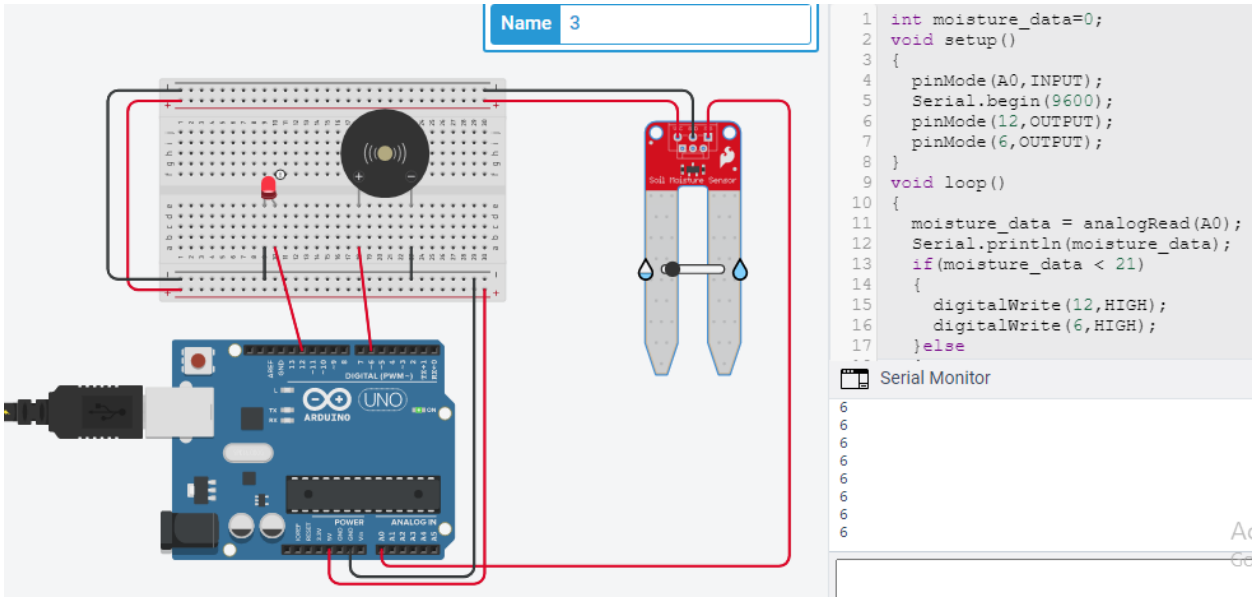



## ARDUINO PROGRAM

```
//21BRS1244
int moisture_data=0;
void setup()
{
  pinMode(A0,INPUT);
  Serial.begin(9600);
  pinMode(12,OUTPUT);
  pinMode(6,OUTPUT);
}
void loop()
{
  moisture_data = analogRead(A0);
  Serial.println(moisture_data);
  if(moisture_data < 21)
  {
    digitalWrite(12,HIGH);
    digitalWrite(6,HIGH);
  }else
  {
    digitalWrite(12,LOW);
    digitalWrite(6,LOW);
  }
  delay(10);
}

```

## SNAP SHOT OF THE OUTPUT



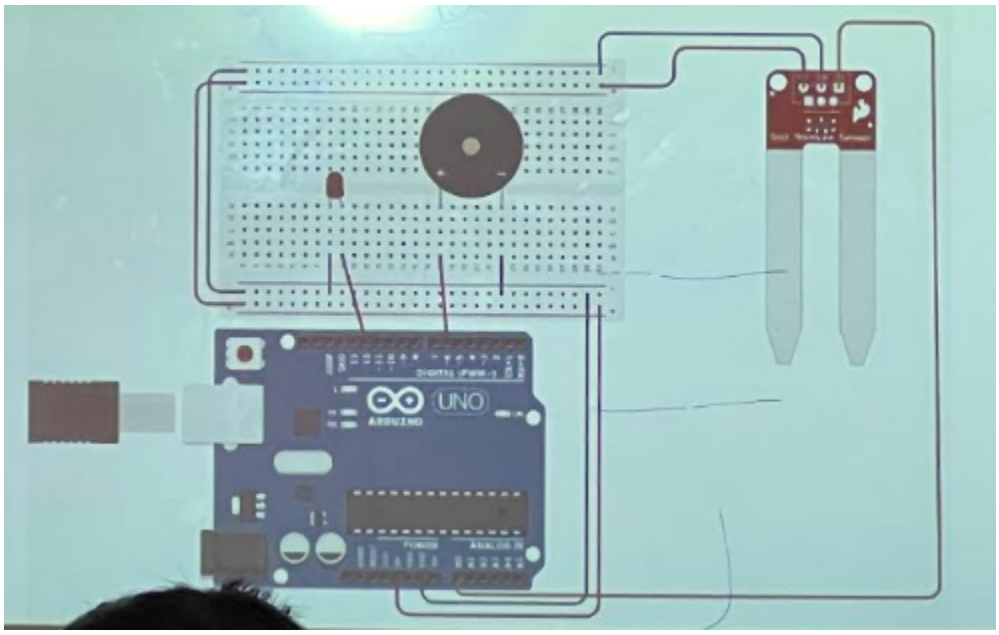




## INFERENCE

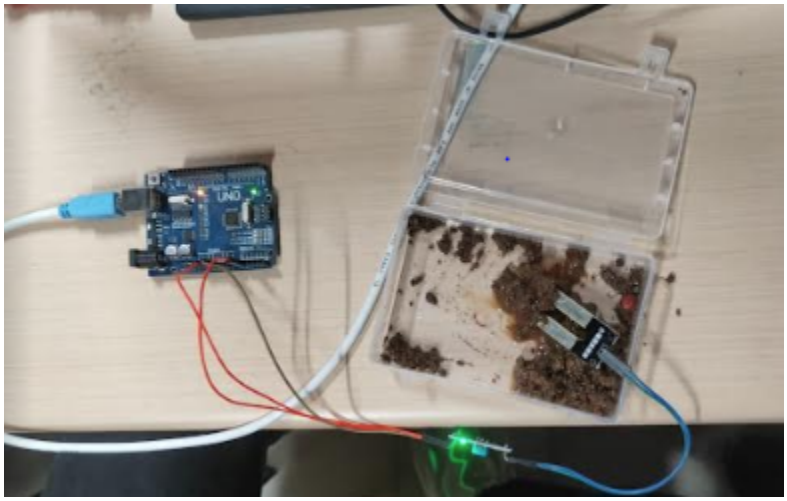
The moisture content of the soil is measured by the sensor and the LED let us know whether the moisture is high or low.

**(ii) USING ARDUINO UNO HARWARE:**

**HARDWARE CIRCUIT BLOCK DIAGRAM**




**HARDWARE CONNECTIONS**



**MEASUREMENT OF SOIL MOISTURE USING ARDUINO UNO**

Analog output of soil moisture sensor is processed using ADC.

The moisture content in terms of percentage is displayed on the serial monitor.

The output of the soil moisture sensor changes in the range of ADC value from 0 to 1023.

This can be represented as moisture value in terms of percentage using formula given below.

(1)

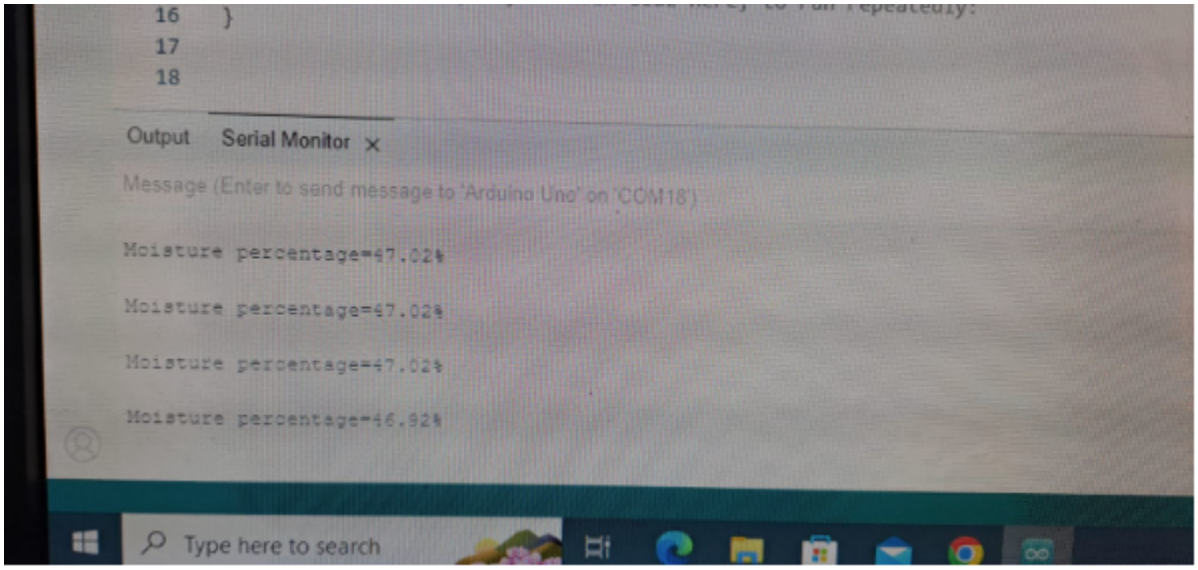
Moisture in percentage = 100 – (Analog output * 100)

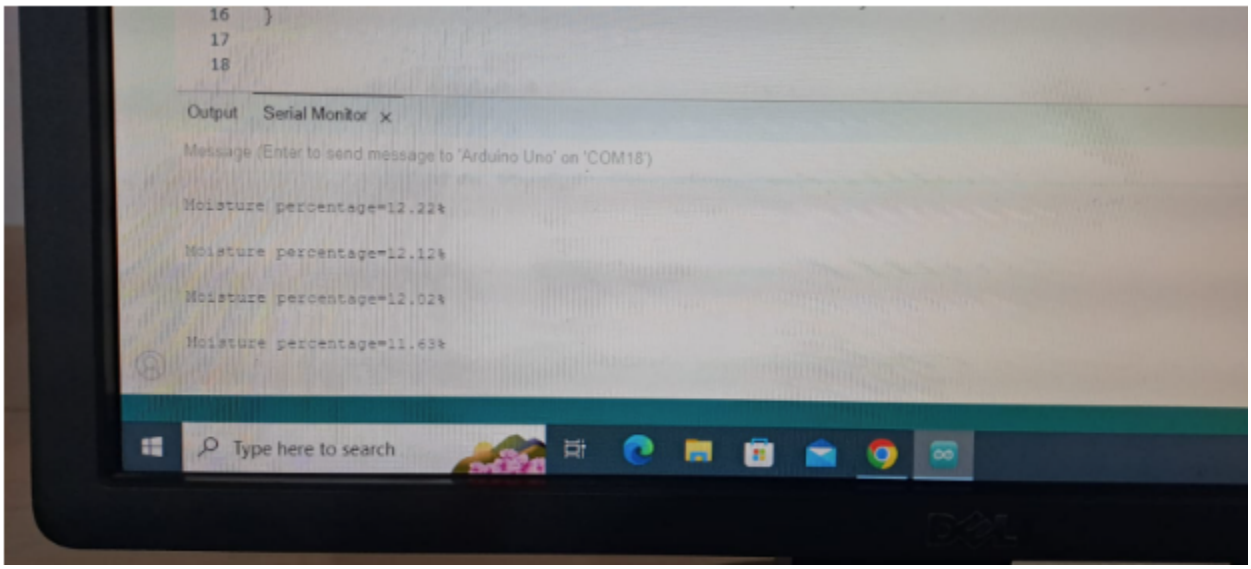
For zero moisture, we get maximum value of 10-bit ADC, i.e. 1023. This, in turn, gives 0% moisture.

**ARDUINO PROGRAM**

```
const int sensor_pin = A1;      /* Soil moisture sensor O/P pin */

void setup() {

  Serial.begin(9600);   /* Define baud rate for serial communication */

}

void loop() {

  float moisture_percentage;

  int sensor_analog;

  sensor_analog = analogRead(sensor_pin);

  moisture_percentage = ( 100 - ( (sensor_analog/1023.00) * 100 ) );

  Serial.print("Moisture Percentage = ");

  Serial.print(moisture_percentage);

  Serial.print("%\n\n");

  delay(1000);

}
```

**SNAP SHOT OF THE OUTPUT**




**INFERENCE**

The moisture content of the soil is measured by the sensor and the LED let us know whether the moisture is high or low.

**RESULT**

The working of the soil moisture monitoring is performed

(i) using sensor with Buzzer and LED a breadboard connected with Arduino UNO in Tinker CAD is analysed and the variations in the moisture content is observed.

(ii) Using the sensor and Arduino Board and the variations in the moisture content is observed.

# BECE351E – INTERNET OF THINGS

**EXPT NO: 3**

**DATE: 09/05/2023**

**NAME: SYED UROOJ UL HASAN**

**REG. NO.: 21BRS1244**

## Smart street light monitoring with Arduino UNO and in Tinker CAD

### AIM

To perform and analyze the working of the photo/LDR sensor detector with Arduino UNO in Tinker CAD and in the hardware kit.

### BLOCKS/COMPONENTS REQUIRED

Tinker CAD Block Modules

1. Arduino UNO
2. Photo/LDR sensor

### ALGORITHM

**Step-1:** Open TinkerCAD.

**Step-2:** In the Design Suite, Drag and drop the required blocks and its associated components into their respective places.

**Step-3:** Connect as per the circuit diagram given in the block diagram. Follow the image and place each part as shown. Notice that the breadboard is used to connect the components together.

It contains a series of columns and two rails on each side. As you click from pin to pin, you'll create wires, each of which can be color-coded for easy identification.

**Step-4:** In the code section, appropriate Arduino codes are fed. Create the next block, an “if” statement, which is a type of control block that makes a decision.

**Step-5:** Once you’re done, select “Start Simulation” on the toolbar to turn on your Arduino Uno.

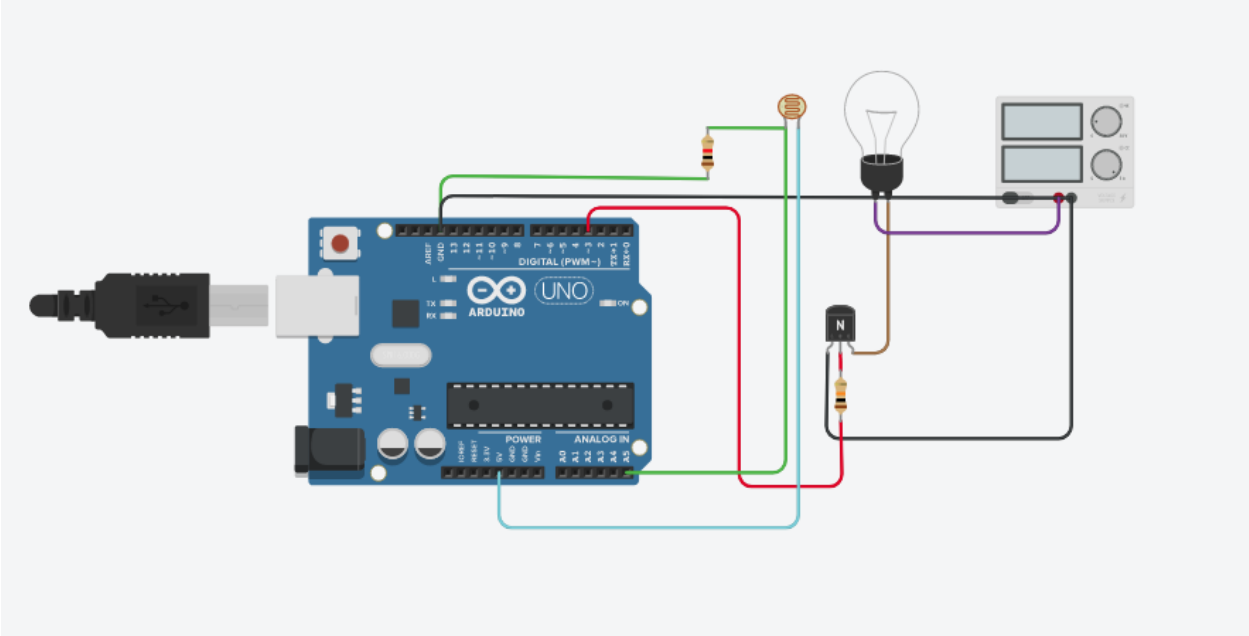
**Step-6:** Troubleshooting: If your program doesn’t behave as expected, check your wiring and programming. Ensure that all pins are properly connected and that each block is written correctly.

## THEORY ABOUT PHOTO SENSOR DETECTOR

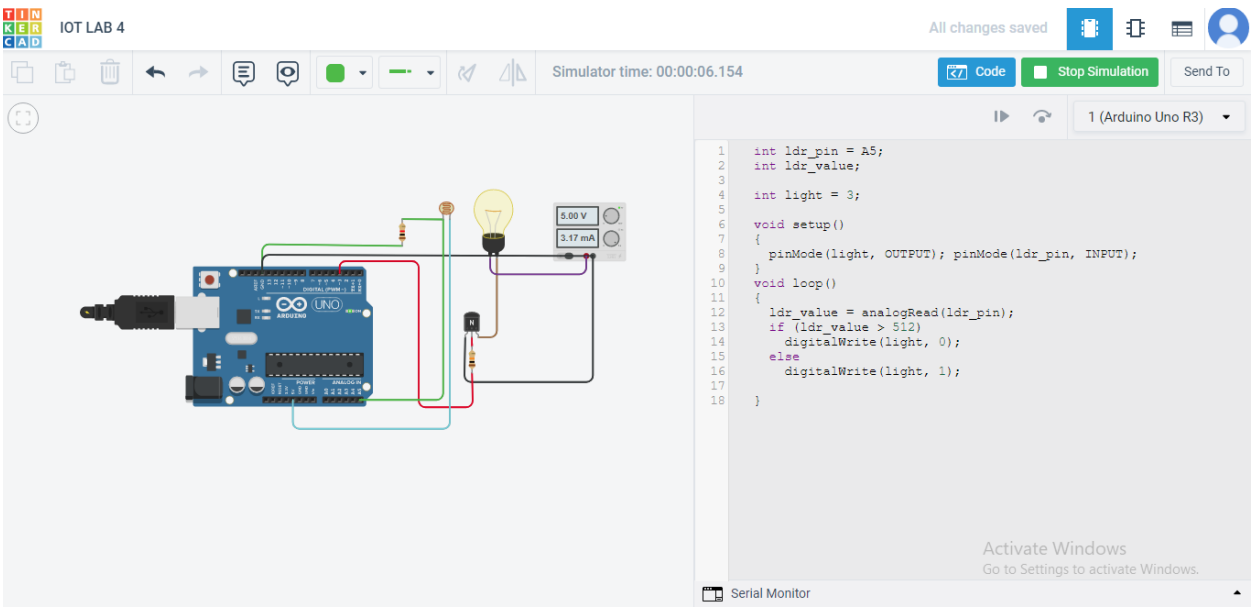
Photoelectric sensors detect objects, changes in surface conditions, and other items through a variety of optical properties.

## CODE:

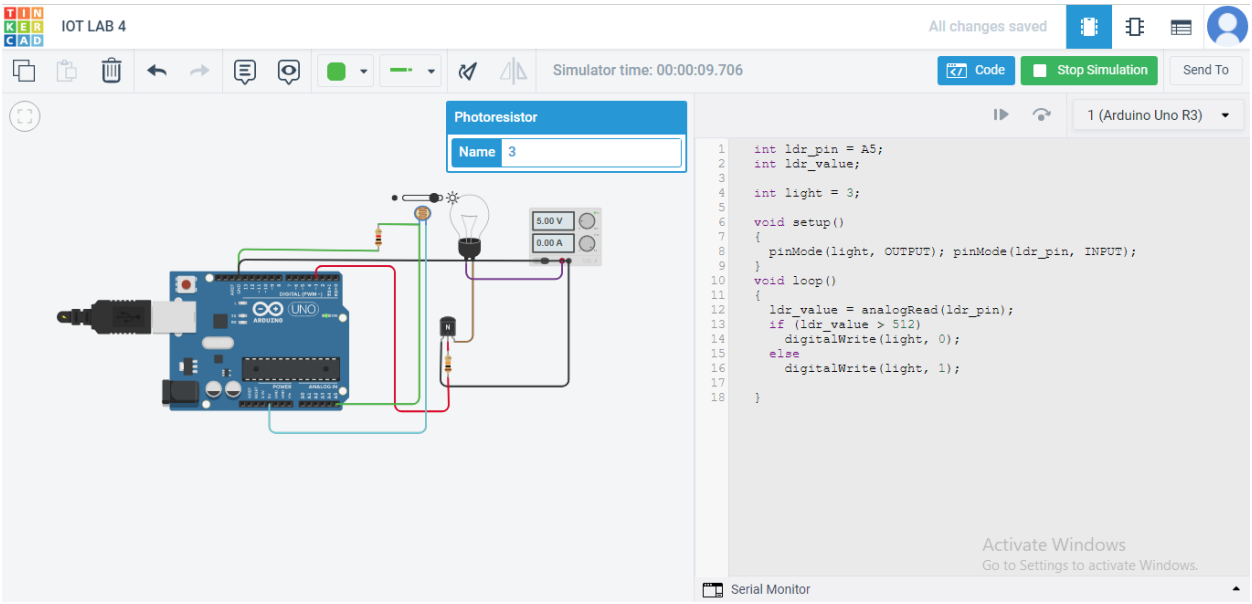
```
 int ldr_pin = A5;
  int ldr_value;

  int light = 3;

  void setup()
  {
    pinMode(light, OUTPUT);
pinMode(ldr_pin, INPUT);
  }
  void loop()
  {
    ldr_value = analogRead(ldr_pin);
    if (ldr_value > 512)
      digitalWrite(light, 0);
    else
      digitalWrite(light, 1);

  }
```




OUTPUT:
CASE 1:




CASE 2:




## INFERENCE

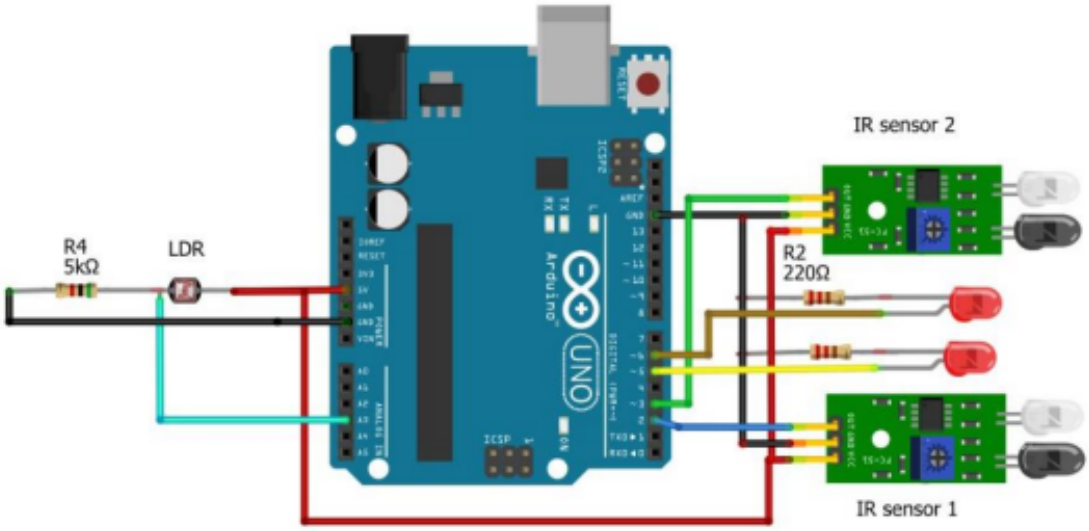
In this experiment, the vehicles are detected using the photo/LDR sensor. After that depending upon the intensity of the headlights of the corresponding vehicles we are detecting if it's daytime then no lights will be on.

## RESULT

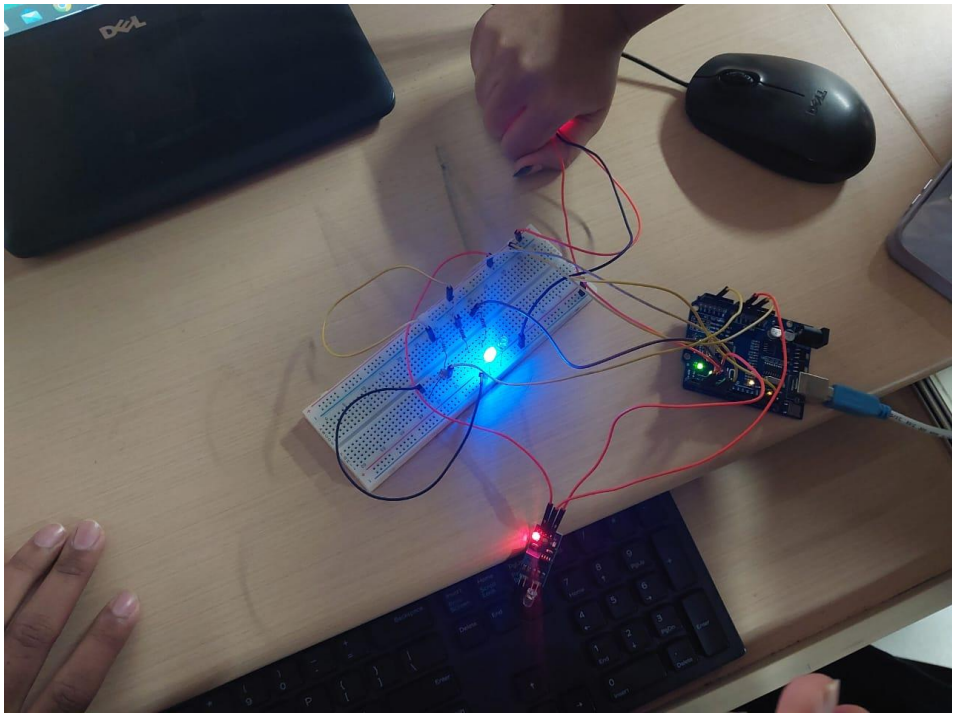
The working of the photo/LDR sensor detector with Arduino UNO in Tinker CAD is analyzed and the variations in the glowing of the light bulb with respect to intensity of the photo/LDR sensor.

## USING ARDUINO UNO HARWARE:

### HARDWARE CIRCUIT BLOCK DIAGRAM




### HARDWARE CONNECTIONS

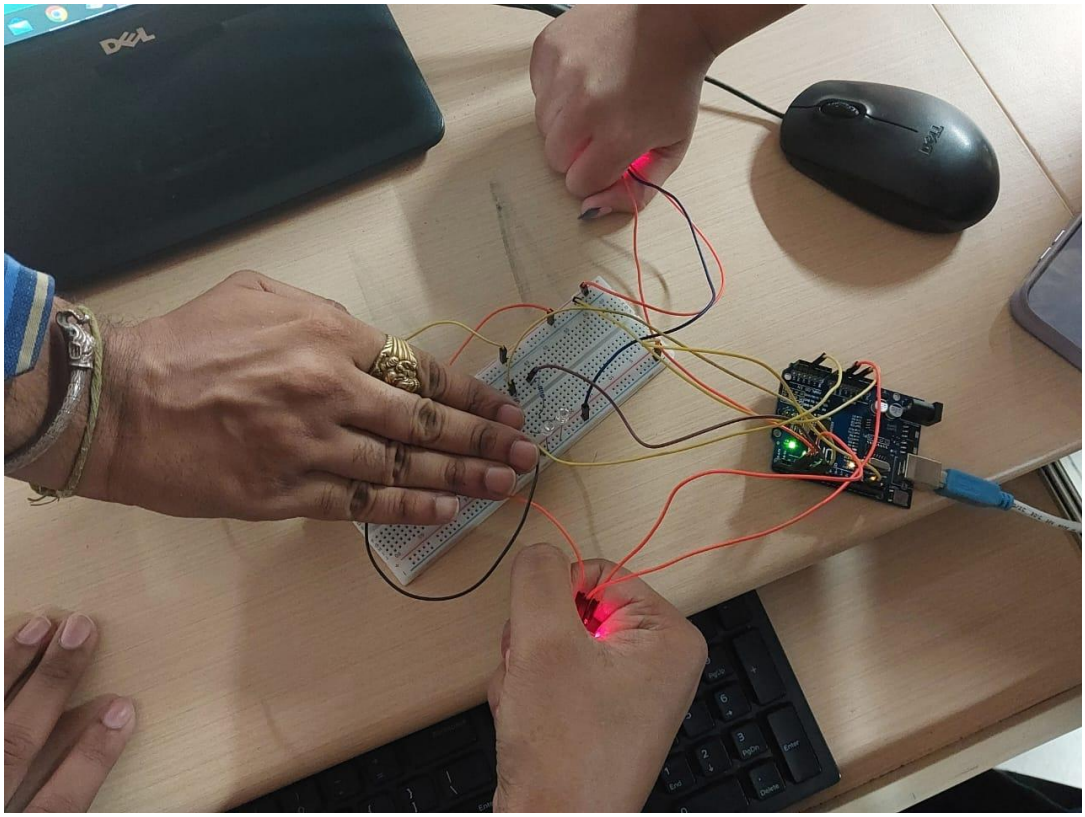


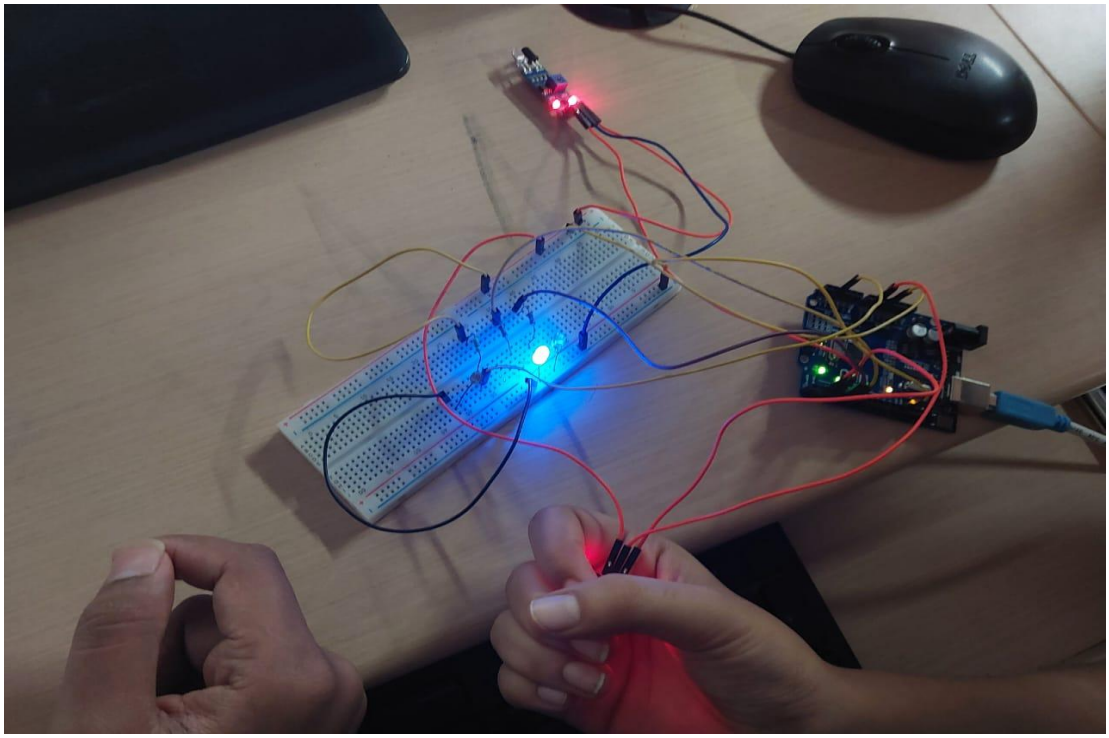
**ARDUINO PROGRAM**

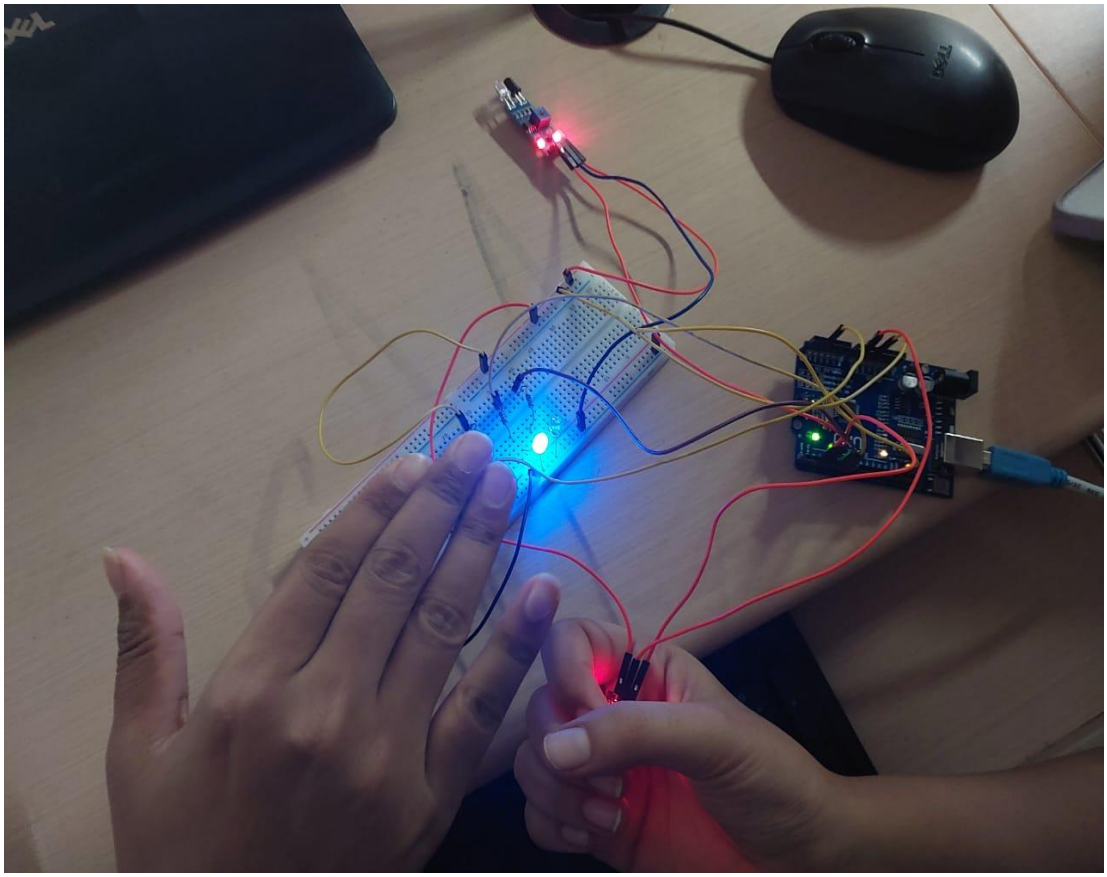
```
int IR1 = 2;
int IR2 = 3;
int LED1 = 5;
int LED2 = 6;
int LDR = A3;
void setup() {
Serial.begin(9600);
pinMode(LED1, OUTPUT);
pinMode(LED2, OUTPUT);
pinMode(IR1, INPUT);
pinMode(IR2, INPUT);
pinMode(LDR, INPUT);
}
void loop() {
int LDRValue = analogRead(LDR);
Serial.print("sensor = ");
Serial.print(LDRValue);
delay(500);
digitalWrite(LED1, LOW);
digitalWrite(LED2, LOW);
Serial.print("It's bright Outside; Lights Status: OFF");
if(LDRValue >500 && digitalRead(IR1) == HIGH)
{
```

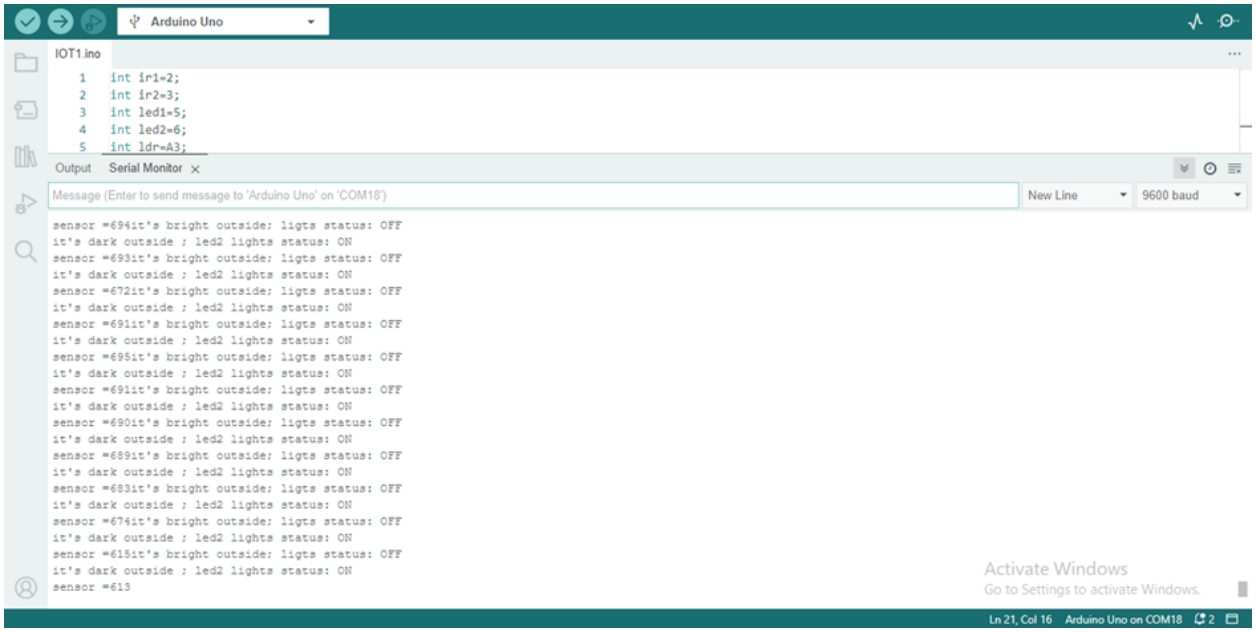
```
digitalWrite(LED1, HIGH); Serial.println("It's Dark Outside; LED! Lights Status: ON");


}

if(LDRValue < 1000 && digitalRead(IR2) == HIGH)

{

digitalWrite(LED2, HIGH); Serial.println("It's Dark Outside; LED2 Lights Status: ON");

}

}
```

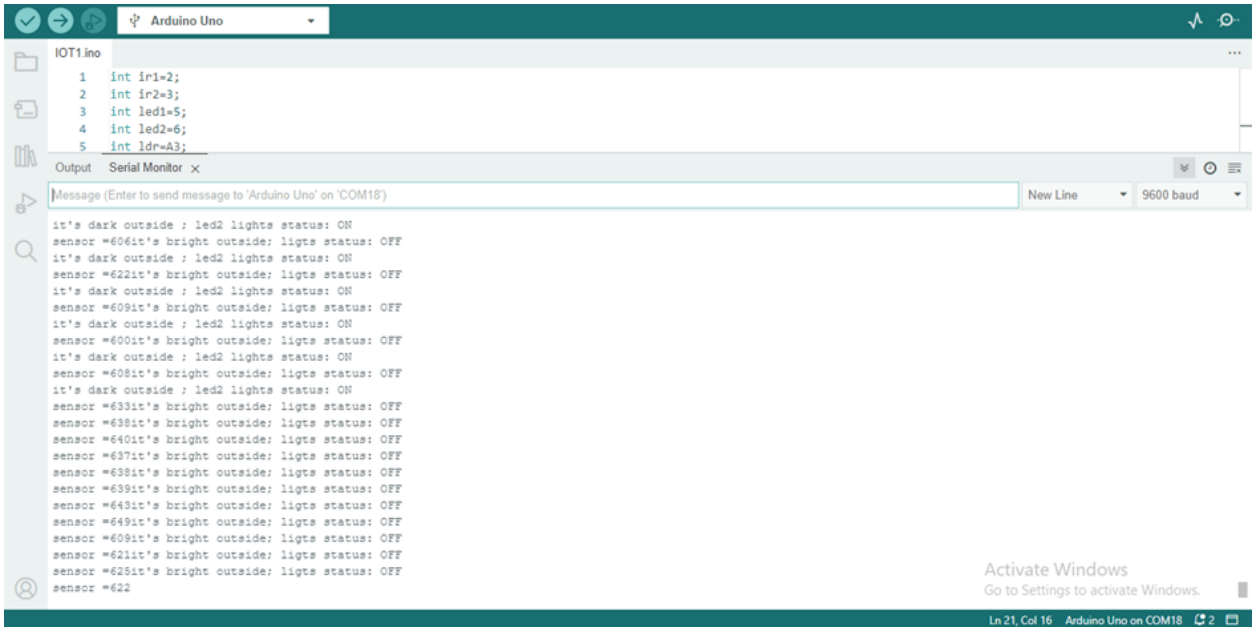
**SNAP SHOT OF THE OUTPUT**










## INFERENCE

In this project, the vehicles are detected using the photo/LDR sensor. After that depending upon the intensity of the headlights of the corresponding vehicles we are detecting if it's daytime then no lights will be on.

## RESULT

The working of the photo/LDR sensor detector with Arduino UNO in Tinker CAD is analyzed and the variations in the glowing of the light bulb with respect to intensity of the photo/LDR sensor.

# BECE351E – INTERNET OF THINGS

**EXPT NO: 4**

**DATE: 16/05/2023**

**NAME: SYED UROOJ UL HASAN**

**REG. NO.: 21BRS1244**

## Environmental monitoring Arduino Uno with DHT 11 Sensor using Node-Red

**AIM**

To perform and analyze the working measures relative humidity and Temperature

(i) Using a DTH 11
(ii) Using Arduino Uno Hardware

**BLOCKS/COMPONENTS REQUIRED**

(i) Hardware:

1. Arduino UNO
2. DTH 11 Sensor

(ii) Software:

1. Node-Red
2. Gauges - Humidity and Temperature
3. Function Node - DHT11 data transform

### THEORY ABOUT DHT11 SENSOR

The DHT11 measures relative humidity.

Relative humidity is the amount of water vapor in air vs. the saturation point of water vapor in air.

At the saturation point, water vapor starts to condense and accumulate on surfaces forming dew. The saturation point changes with air temperature.

Cold air can hold less water vapor before it becomes saturated, and hot air can hold more water vapor before it becomes saturated.

The DHT11 is a commonly used Temperature and humidity sensor.

The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values

of temperature and humidity as serial data.

The sensor is also factory calibrated and hence easy to interface with other microcontrollers.

The formula to calculate relative humidity is:

Relative humidity is expressed as a percentage.

At 100% RH, condensation occurs, and at 0% RH, the air is completely dry.

**(i) USING NODE-RED:**

**ALGORITHM**

**Step-1:** You can use the node-red command to start “Node-RED” in your terminal.

**Step-2:** You can then access the Node-RED editor by pointing your browser at http://localhost:1880.

**Step-3:** The node named Arduino is a "serial in" node which you can find in the network section of the node library. If you don't see the serial in node in the network library then you have to install it first.

Double click on the serial in node to open its properties window.

Then select the COM port your Arduino is connected to which in this case is COM17. Also optionally you can name the node which here is Arduino.

**Step-4:** Function node named DHT11 Data Transform. This function node is available in the function section of the node library.

Drag the function node and place it after the serial in node. Then double click it to open its configuration window.

```
const m = msg.payload.split(',');

const H = {payload:parseFloat(m[0])};

const T = {payload:parseFloat(m[1])};

return [H,T];
```

**Step-5:** Connect the two output nodes fro the function node to these two gauge nodes.

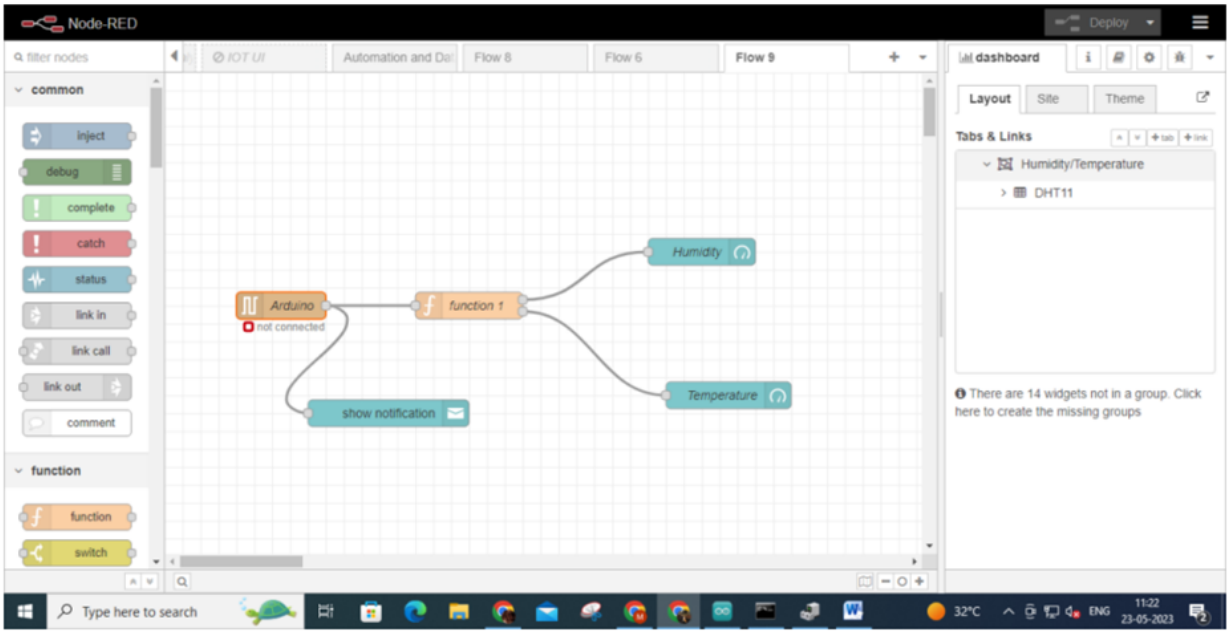
**Step-6:** The dashboard nodes requires tab and group names so that they can be manipulated for display. On the node red configuration palette click on dashboard.

In the group edit window, type in the name for the group such as DHT11 here. Humidity/Temperature should be selected in the Tab and type in 8 in the Width section.

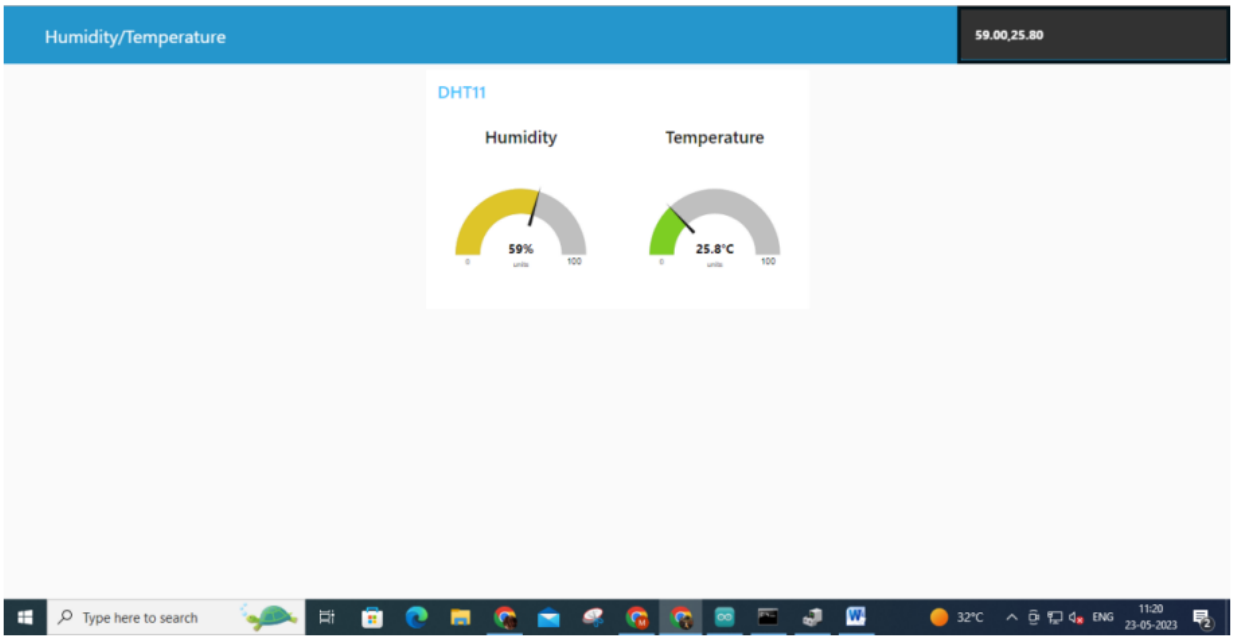
**Step-7:** localhost:1880/ui to open the dashboard or use the dashboard open link as

shown below.

## BLOCK DIAGRAM:




## OUTPUT:

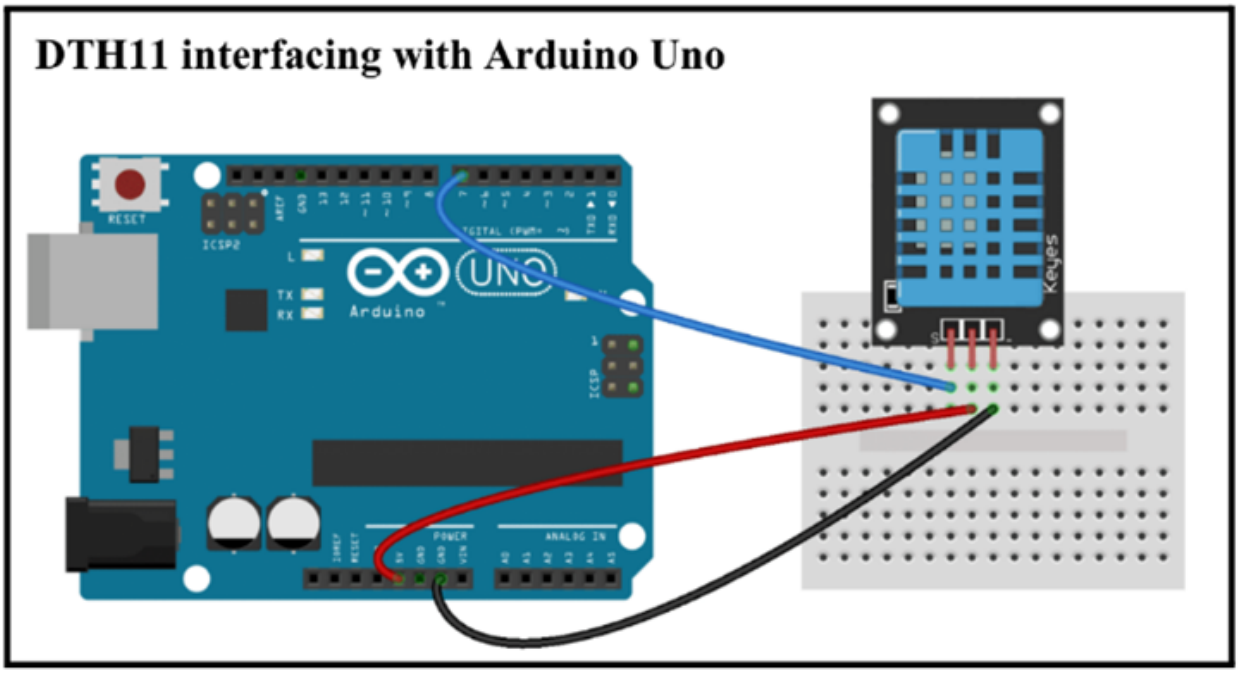



## INFERENCE

The ambient humidity and temperature keeps changing when the DHT11 sensor senses its surroundings.

## (ii) USING ARDUINO UNO HARDWARE:

## HARDWARE CIRCUIT BLOCK DIAGRAM



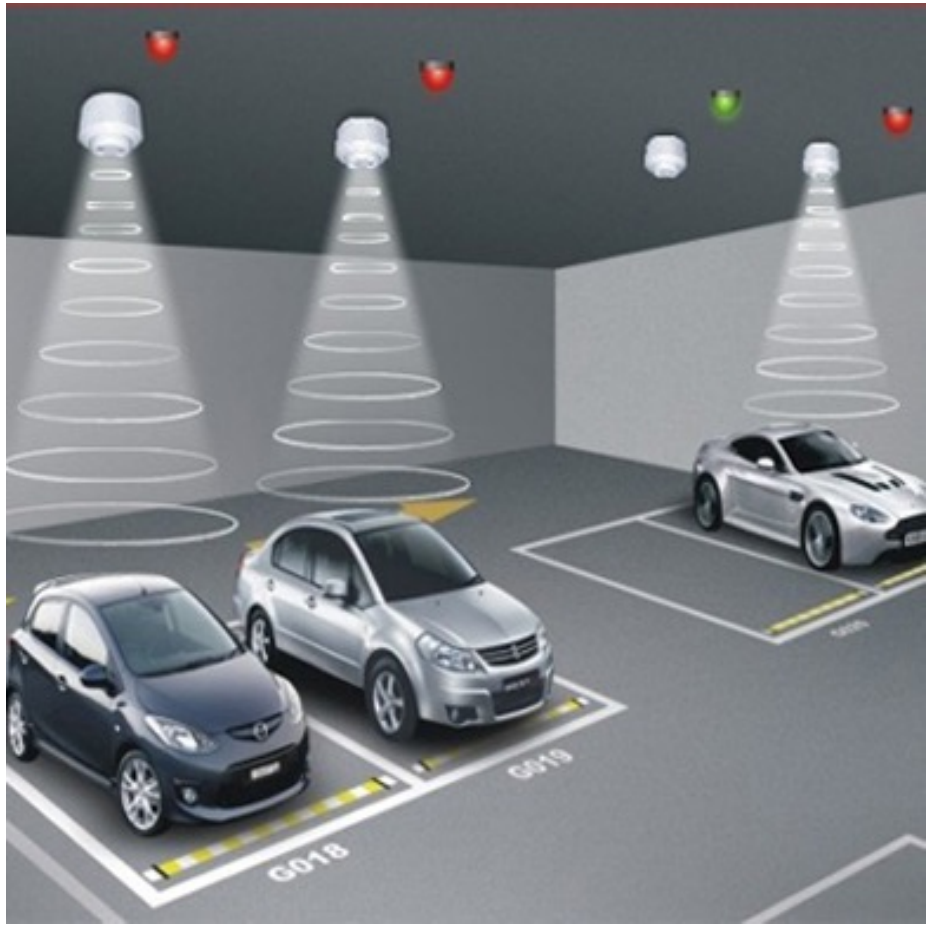

## ARDUINO PROGRAM

```
#include <DHT.h> // Include Adafruit DHT11 Sensors

#define DHTPIN 7 // DHT11 Output Pin connection
#define DHTTYPE DHT11 // DHT Type is DHT11 DHT
DHT dht(DHTPIN, DHTTYPE); // Initialize DHT sensor

void setup() {
    Serial.begin(9600); // To see data on serial monitor
    dht.begin();
}

void loop() {
    float H = dht.readHumidity(); // Read Humidity
    float T = dht.readTemperature(); // Read temperature as Celsius
    if (isnan(H) || isnan(T)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }

    // Combine Humidity and Temperature into single string
    String dhtData = String(H) + "," + String(T);
    Serial.println(dhtData);
    delay(2000); // Wait two seconds between measurements
}
```

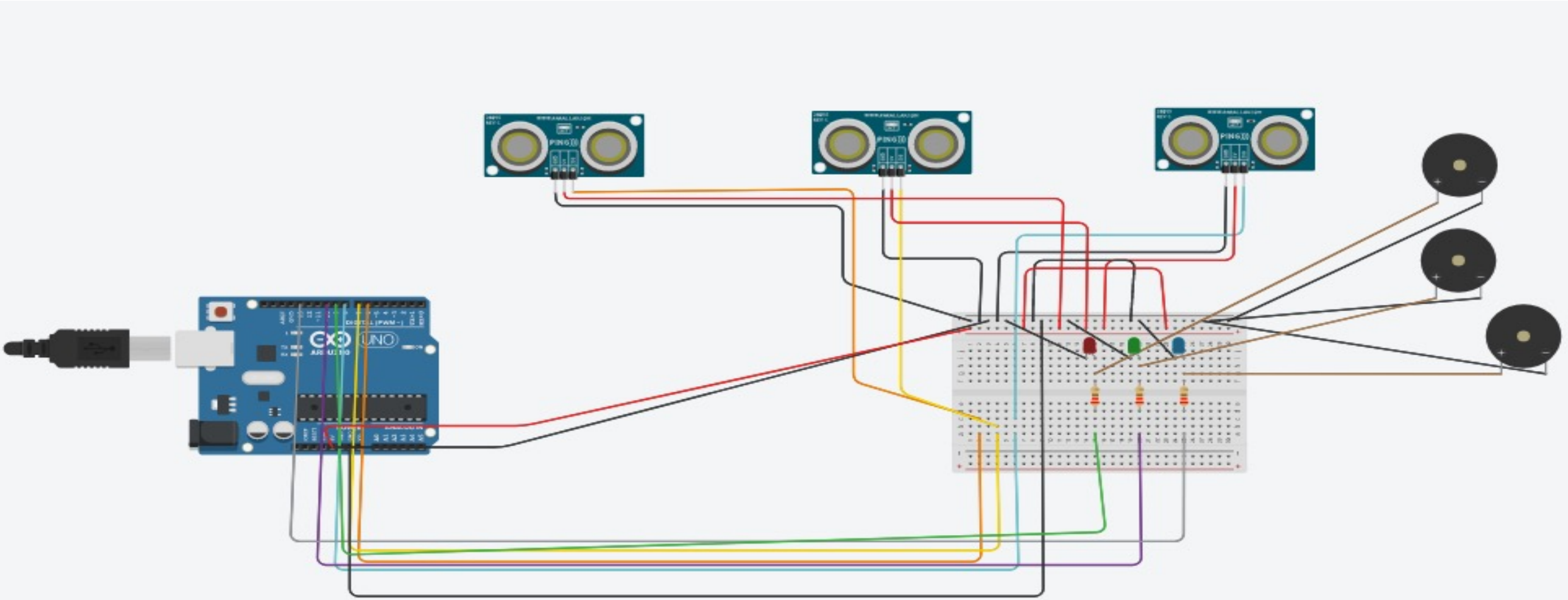
**RESULT**

We have successfully setup an Environmental monitoring Arduino Uno with DHT11 Sensor using Node-Red.

# Experiment 5
# SMART Car Parking System using TinkerCAD and Arduino



# Schematic - TINKERCAD




Simulation Component Required:
LED 3, Resistors: 3 Nos 220ohms, Ultrasonic

# TINKER CAD PROGRAM

```
#define trigPin1 6
#define trigPin2 7
#define trigPin3 8
#define echoPin1 6
#define echoPin2 7
#define echoPin3 8
#define led1 9
#define led2 10
#define led3 13
bool bool1=false;
bool bool2=false;
bool bool3=false;
long readUltrasonicDistance(int triggerPin, int
echoPin){
   pinMode(triggerPin, OUTPUT);
   digitalWrite(triggerPin, LOW);
   delayMicroseconds(2);
   digitalWrite(triggerPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(triggerPin, LOW);
   pinMode(echoPin, INPUT);
   return pulseIn(echoPin, HIGH);   }
```

```
void setup() {
  //Serial.begin (9600);
  pinMode(trigPin1, OUTPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(trigPin3, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(echoPin2, INPUT);
  pinMode(echoPin3, INPUT);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  Serial.begin(9600);

void loop() {
  float distance1;
  float distance2;
  float distance3;
   distance1 = (readUltrasonicDistance(6, 6) * 0.034)/2;
   //Serial.println(distance1);
   if (distance1 >=320|| distance1 <= 2){
    digitalWrite(led1, HIGH);
     bool1=false;   }
```

# TINKER CAD PROGRAM

```
else {
    digitalWrite(led1, LOW);
    bool1=true;
  }
    distance2 = (readUltrasonicDistance(7, 7) *
0.034)/2;
   if (distance2 >=320 || distance2 <= 2){
    digitalWrite(led2, HIGH);
     bool2=false;
  }
  else {
    digitalWrite(led2, LOW);
    bool2=true;
  }
    distance3 = (readUltrasonicDistance(8, 8) *
0.034)/2;
   if (distance3 >=320 || distance3 <= 2){
    digitalWrite(led3, HIGH);
     bool3=false;
  }
```

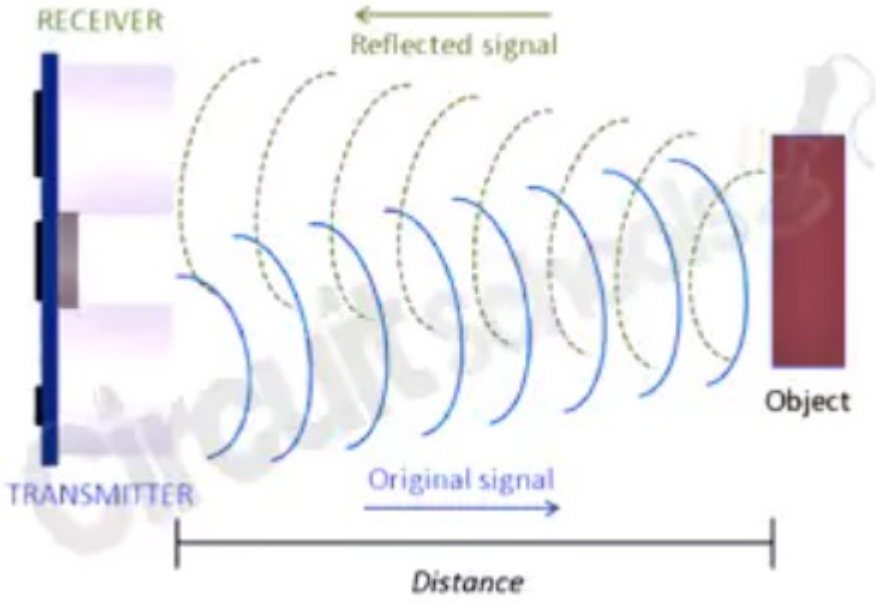
```
else {
  digitalWrite(led3, LOW);
  bool3=true;
 }
if(!bool1&&!bool2&&!bool3)
 {
  Serial.println("All slots empty");
 }
 else if(bool1&&!bool2&&!bool3)
 {
 Serial.println("1 Slot filled");
 Serial.println("Slot 2 3 empty");
 }
 else if(!bool1&&bool2&&!bool3)
 {
  Serial.println("1 Slot filled");
  Serial.println("Slot 1 3 empty");
 }
 else if(bool1&&bool2&&!bool3)
 {
  Serial.println("2 Slot filled");
  Serial.println("Slot 3 empty");
 }
```

# TINKER CAD PROGRAM

```
  else if(!bool1&&!bool2&&bool3)
  {
    Serial.println("1 Slot filled");
    Serial.println("Slot 1 2 empty");
  }
  else if(bool1&&!bool2&&bool3)
  {
    Serial.println("2 Slot filled");
    Serial.println("Slot 2 empty");
  }
  else if(!bool1&&bool2&&bool3)
  {
    Serial.println("2 Slot filled");

    Serial.println("Slot 1 empty");
  }
  else
  {
      Serial.println("All Slots filled");
  }
  delay(500);
}
```

# HC-SR04 ultrasonic sensor

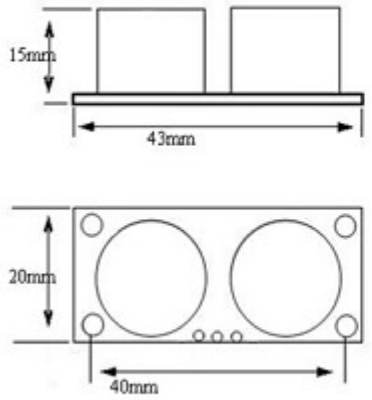
- When is sensor is powered on an ultrasonic sound wave(at higher wavelength, humans cannot hear) is emitted through one of its transducers, and waiting for the sound to bounce off  at some present object, the echo is captured by the second transducer.
- The distance is proportional to the time it takes for the echo to arrive.
- The sensor sends an ultrasonic sound wave through the trigger or trigger, **bounces off the object** and the receiver or echo detects the wave.
- By measuring the time taken for the wave from emission till it gets back we can know the distance.

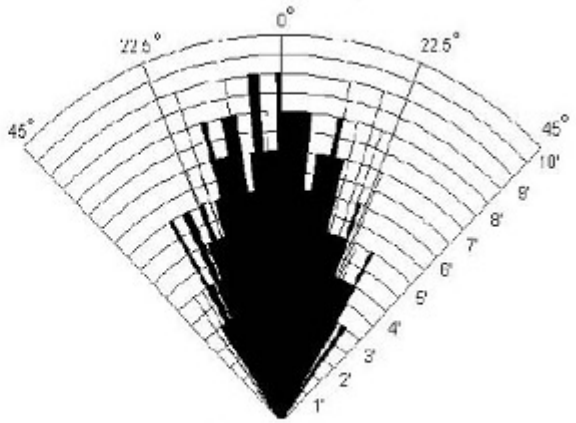



**https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/**

# Working

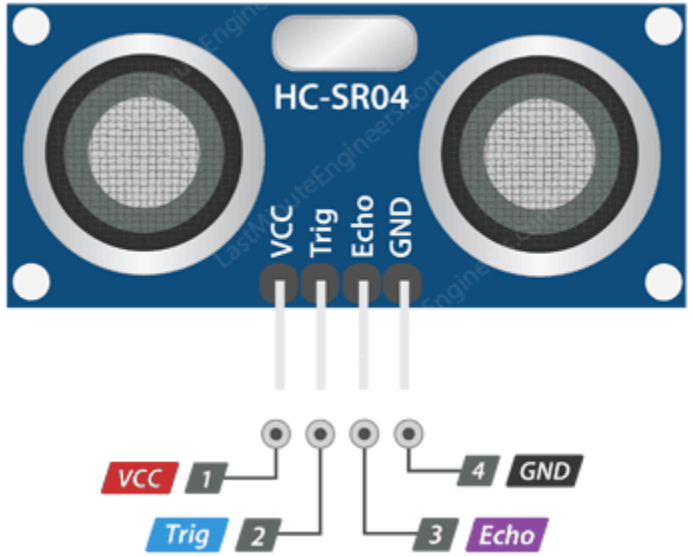
HCSR04 is an Arduino library HCSR04 Sensors




*Practical test of performance,*

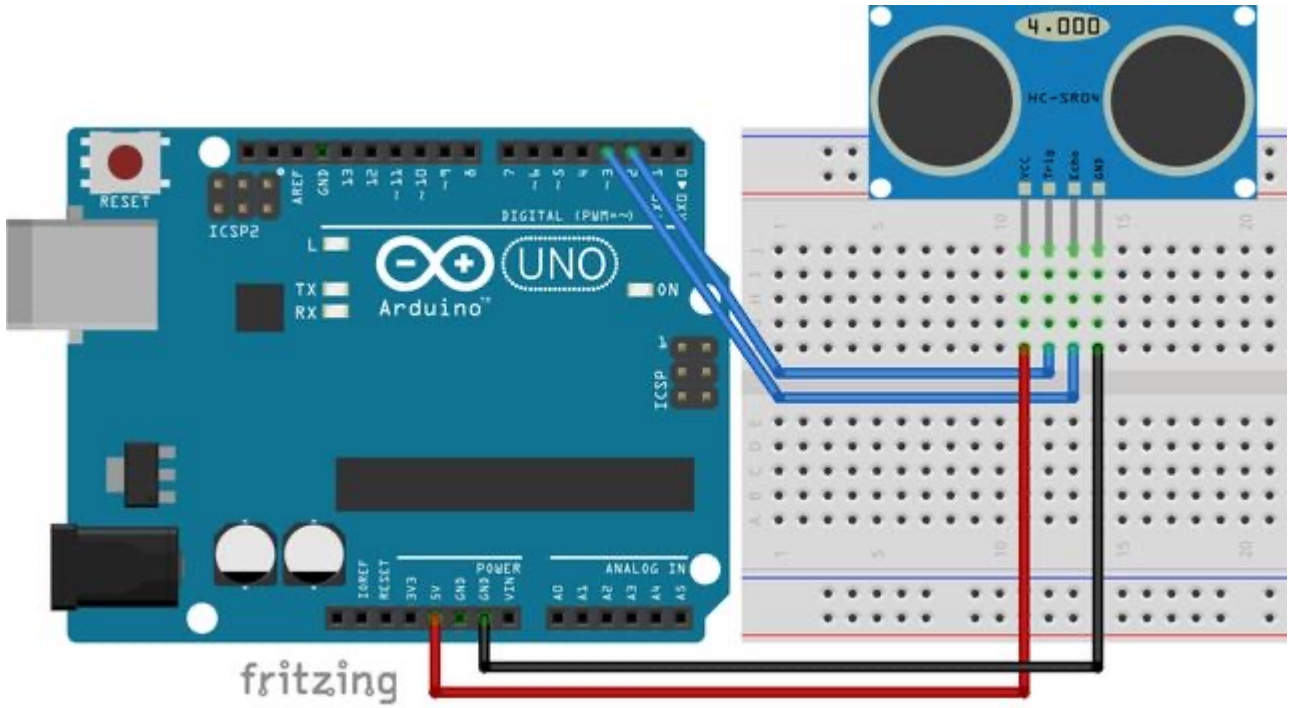
*Best in 30 degree angle*




**Trigger Pin:** pin is used to trigger ultrasonic sound pulses. By setting this pin to HIGH for 10µs, the sensor initiates an ultrasonic burst.

**Echo:** pin goes high when the ultrasonic burst is transmitted and remains high until the sensor receives an echo, after which it goes low. By measuring the time the Echo pin stays high, the distance can be calculated

# HARDWARE CIRCUIT - 1

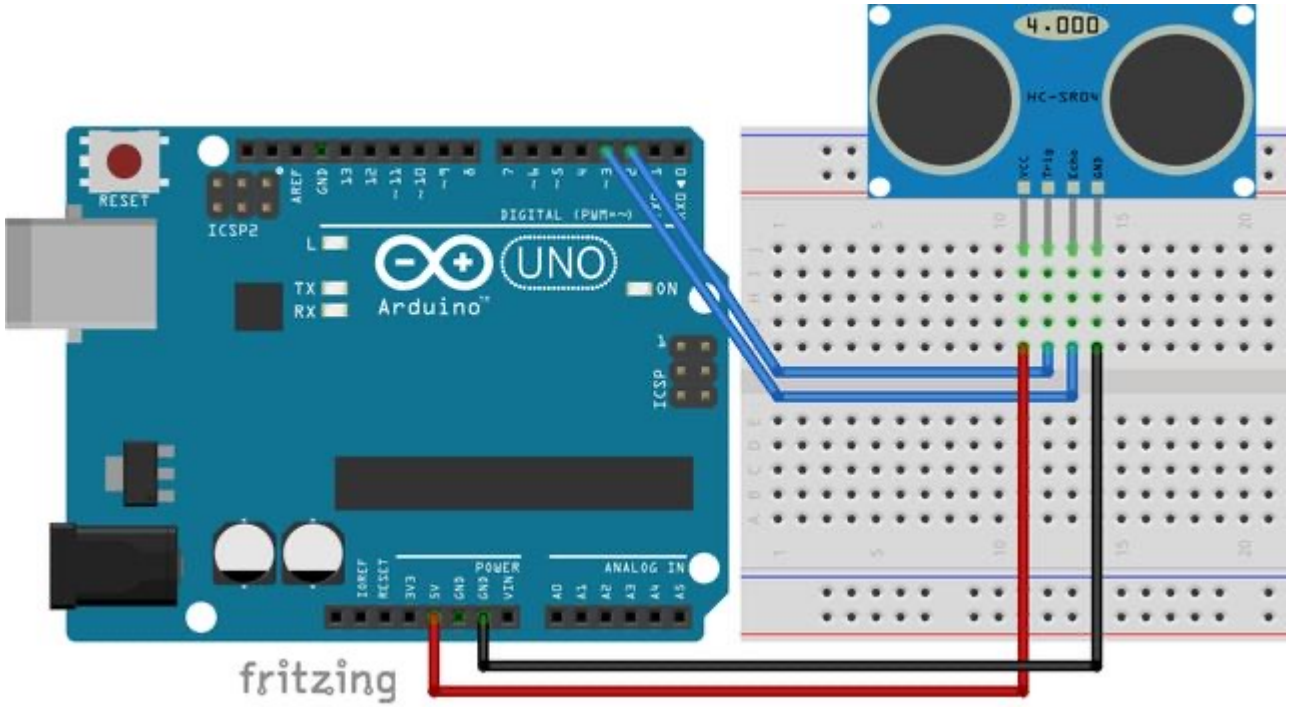



Additionally Connect LED and Buzzer – Only one slot free

# PROGRAM : Automated Car Parking System
# Project using Arduino Uno without Libraries

```
const int trigPin = 2;
const int echoPin = 3;
const int buzzer = 6;
// defines variables
long duration;
int distance;
int safetyDistance;

void setup() {
pinMode(trigPin, OUTPUT); // Sets the
trigPin as an Output
pinMode(echoPin, INPUT); // Sets the
echoPin as an Input
pinMode(buzzer, OUTPUT);
safetyDistance = ultrasonic();
Serial.begin(9600); // Starts the serial
communication
}
void loop() {
int dist = ultrasonic();
if (dist <= safetyDistance){
  tone(buzzer, 2500);
  delay(500);
  tone (buzzer, 2500);
}
else{
  noTone (buzzer);
}
// Prints the distance on the
Serial Monitor
Serial.print("Distance: ");
Serial.println(dist);
}
int ultrasonic(){
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state
for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the
sound wave travel time in
microseconds
duration = pulseIn(echoPin,
HIGH);
// Calculating the distance
distance= duration*0.034/2;
return distance;
}
```

# HARDWARE CIRCUIT- 2




**Additionally Connect LED and Buzzer – Only one slot free**
**HCSR04 hc1(2, 3);**
**//initialisation class HCSR04 (trig pin , echo pin)**
**HCSR04 hc2(9,10);**
**Include the HCSR04 libraries by gamegine from Arduino uno IDE -> Tools -> manage -> libraries**
**Check for sample programs in files->examples->HCSR04 Ultrasonic ->HCSR04**

Sample Testing Program

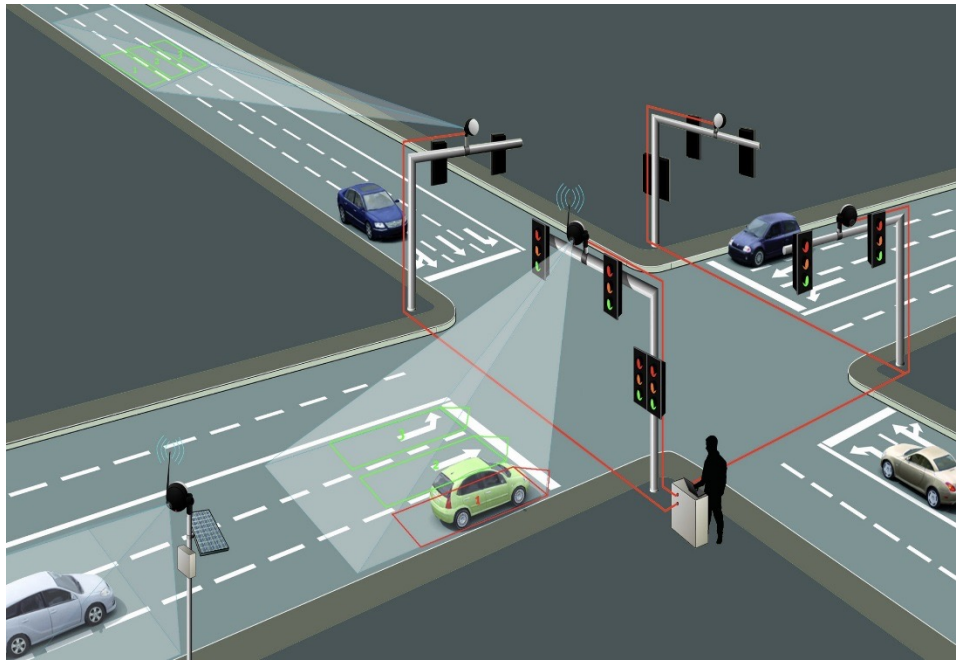
```
#include <HCSR04.h>
HCSR04 hc(5, 6);
//initialisation class HCSR04
(trig pin , echo pin)
void setup() {
Serial.begin(9600); }
 void loop() {
Serial.println( hc.dist() );
//return current distance (cm)
in serial
delay(60);
// we suggest to use over 60ms
measurement cycle, in order to
prevent trigger signal to the
echo signal. }
```

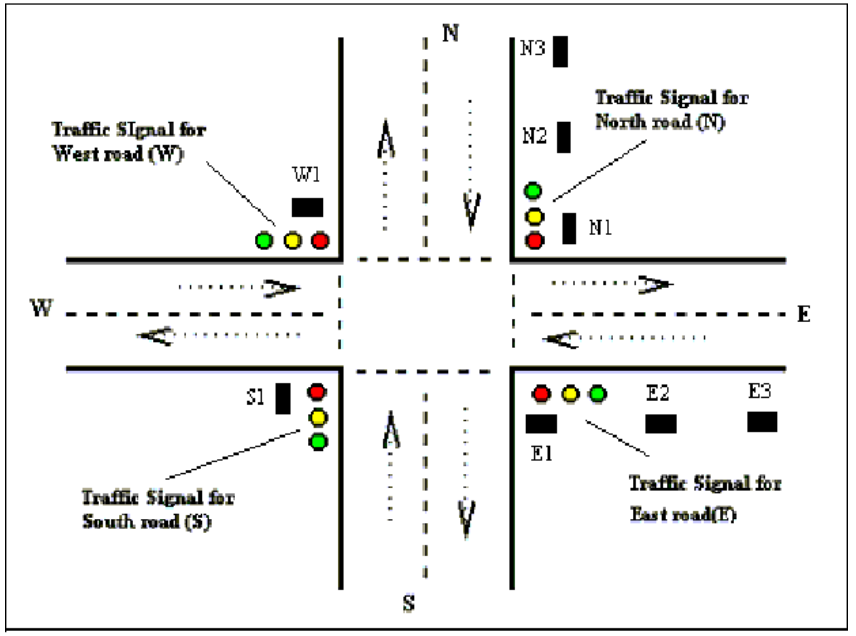
# PROGRAM : Automated Car Parking System Project using Arduino Uno with Libraries – Optimized Code

```
#include <HCSR04.h>
HCSR04 hc1(5, 6);
//initialisation class HCSR04 (trig pin , echo pin)
HCSR04 hc2(9,10);
void setup()
{
Serial.begin(9600);
pinMode(11,OUTPUT);
pinMode(12,OUTPUT);
pinMode(13,OUTPUT);
}
void loop() {
Float dist1,dist2;
 Serial.println(hc1.dist());
// return curent distance in serial
 delay(200);
Serial.println(hc2.dist());
delay(200);
dist1=hc1.dist();
dist2=hc2.dist();
if (dist1<30){
digitalWrite(11,HIGH);
 Serial.println("Parking Slot 1 is full");
digitalWrite(13,HIGH);
}
Else
{
digitalWrite(11,LOW);
 digitalWrite(13,LOW);
 }
 if (dist2<30){
digitalWrite(12,HIGH);
 Serial.println("Parking Slot 2 is full");
digitalWrite(13,HIGH);
}
Else
{
 digitalWrite(12,LOW);
digitalWrite(13,LOW);
 }
}
```

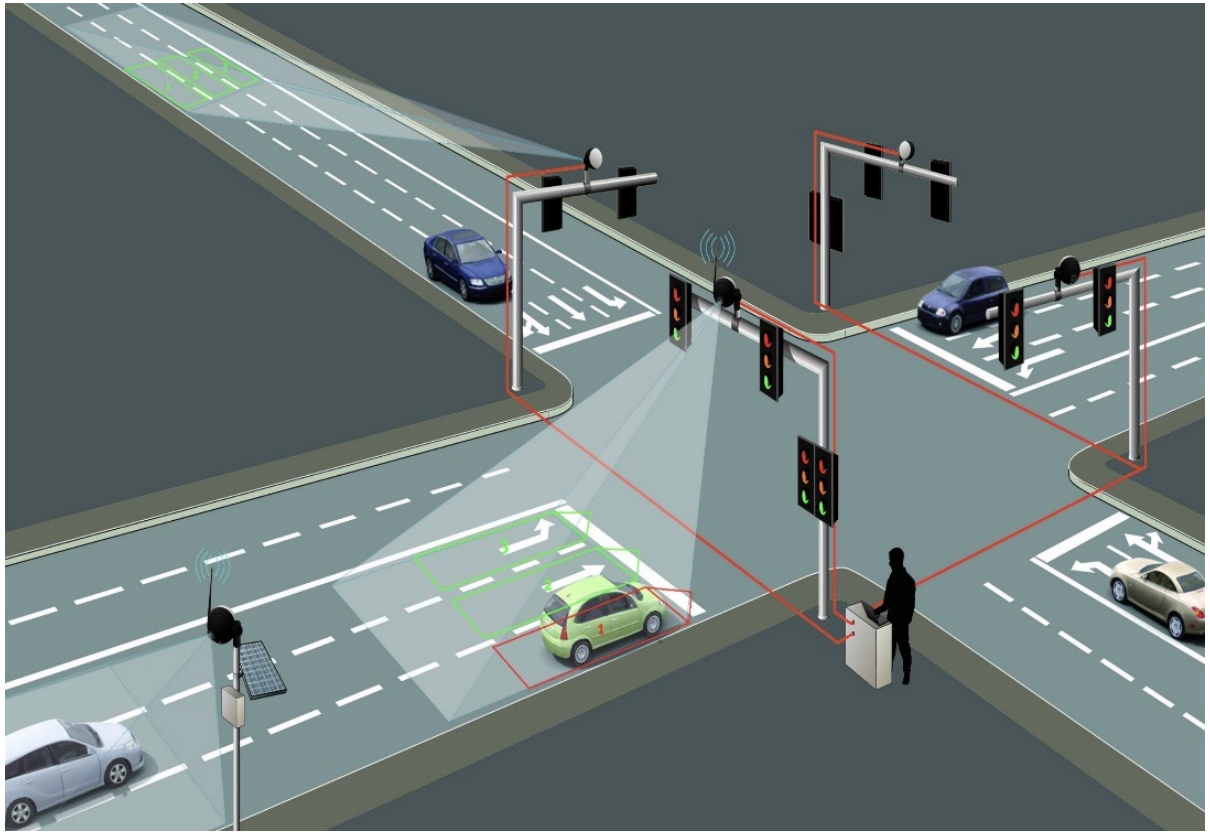
**Case study:**
**Sample for reference**
**https://www.ibm.com/cloud/blog/iot-enabled-smart-parking-meter**

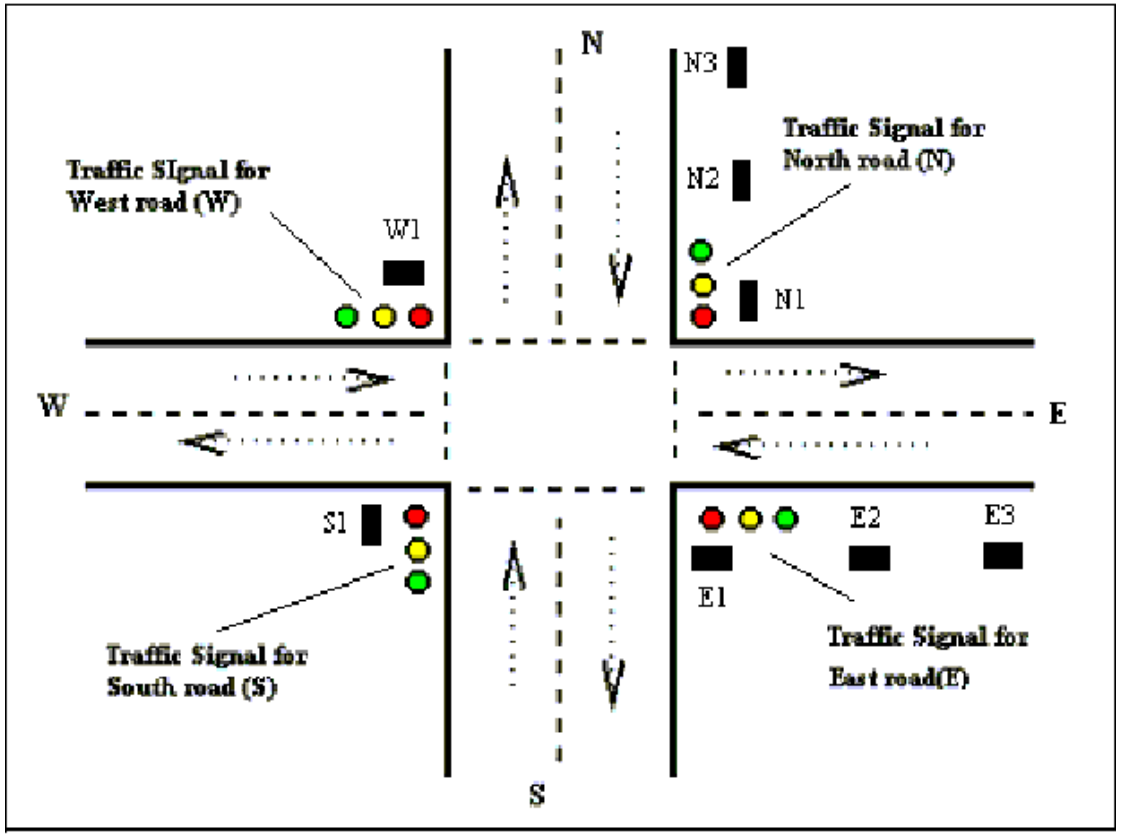
# Experiment 6
# SMART Traffic Light Control System using TinkerCAD and Arduino



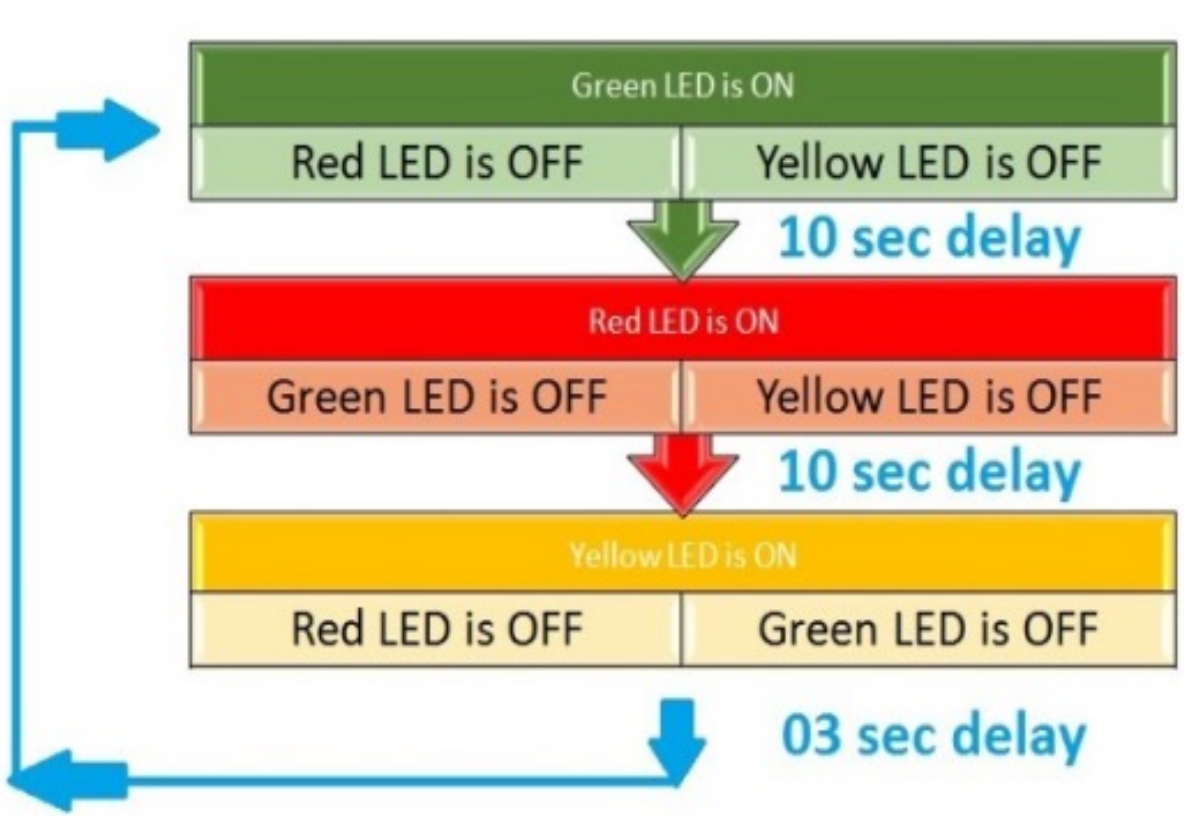


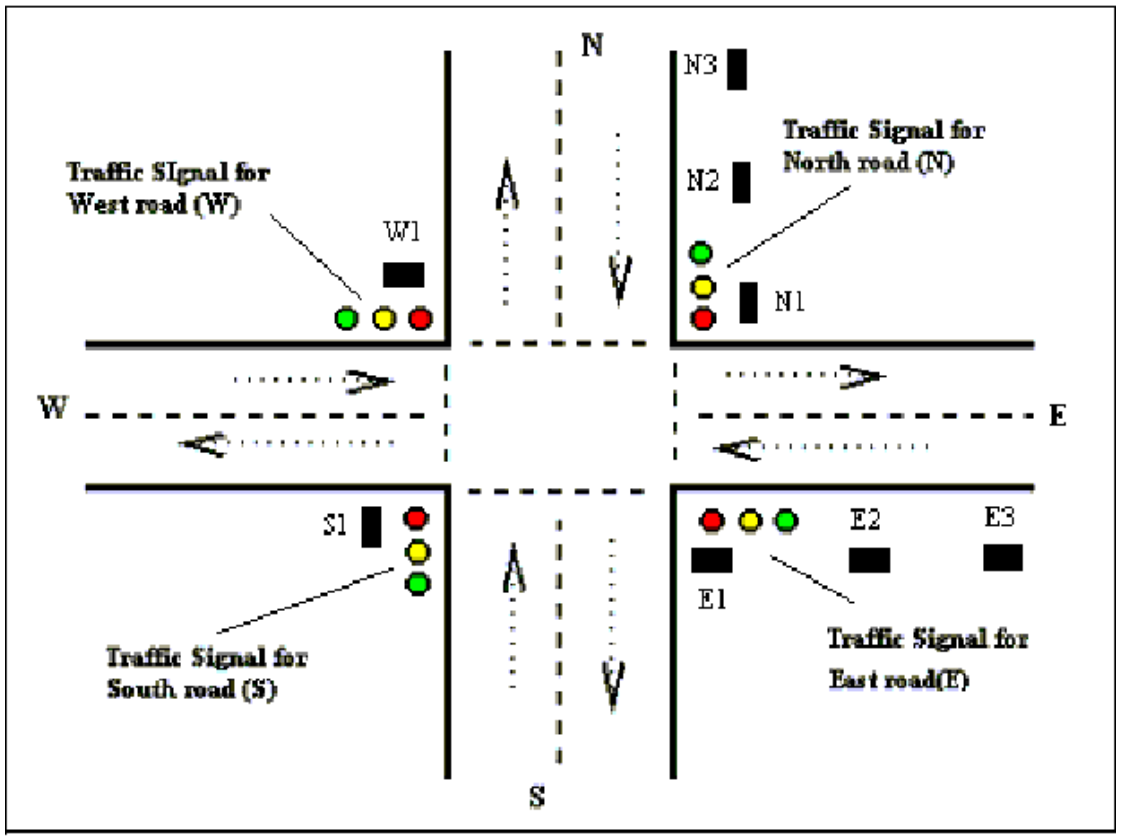
# SMART Traffic Light Control System





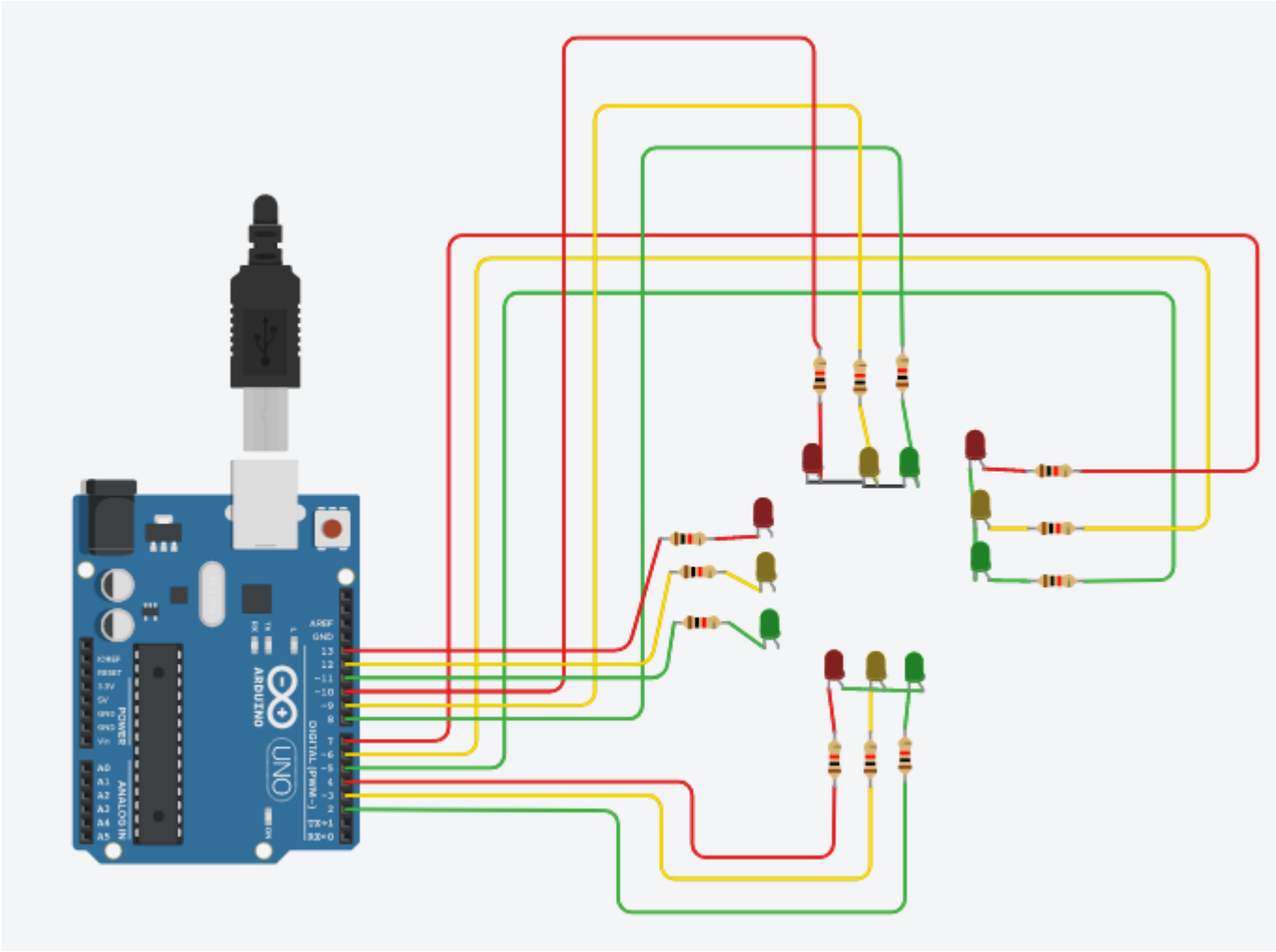
# SMART Traffic Light Control System



# Schematic 1- TINKERCAD

**Debug the Circuit to complete the Simulation**



Simulation Component Required:
LED 12, Resistors: 12 Nos 220ohms,

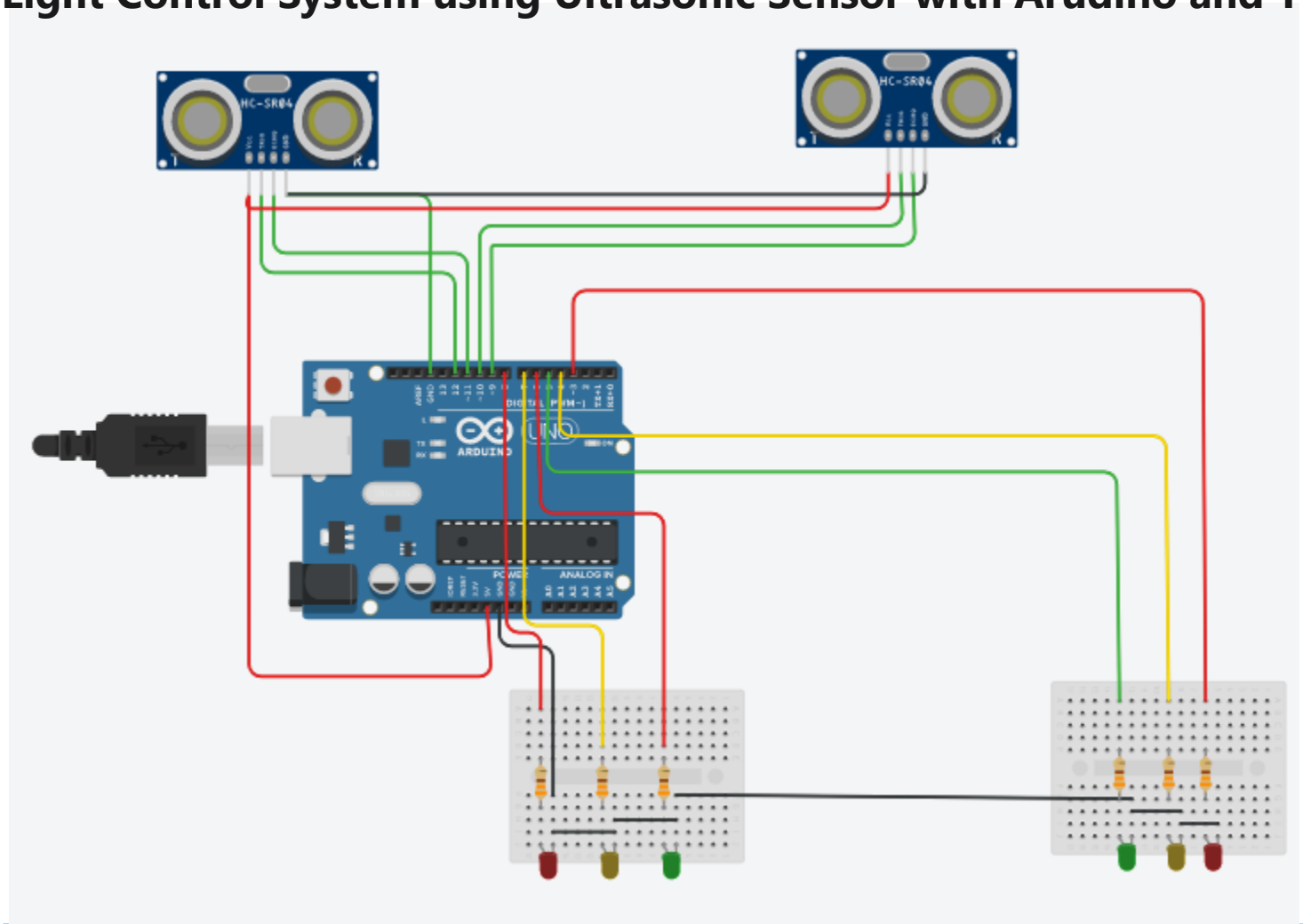
# TINKER CAD PROGRAM 1

```
int Lane1[] = {13,12,11}; // Lane 1 Red, Yellow and Green
int Lane2[] = {10,9,8};// Lane 2 Red, Yellow and Green
int Lane3[] = {7,6,5};// Lane 3 Red, Yellow and Green
int Lane4[] = {4,3,2};// Lane 4 Red, Yellow and Green
void setup()
{
 for (int i = 0; i < 3; i++)
 {
  pinMode(Lane1[i], OUTPUT);
  pinMode(Lane2[i], OUTPUT);
  pinMode(Lane3[i], OUTPUT);
  pinMode(Lane4[i], OUTPUT);
 }
 for (int i = 0; i < 3; i++)
 {
  digitalWrite(Lane1[i], LOW);
  digitalWrite(Lane2[i], LOW);
  digitalWrite(Lane3[i], LOW);
  digitalWrite(Lane4[i], LOW);
 }}
```

```
void loop()
 {
digitalWrite(Lane1[2], HIGH);
 digitalWrite(Lane3[0], HIGH);
 digitalWrite(Lane4[0], HIGH);
 digitalWrite(Lane2[0], HIGH);
 delay(7000);
 digitalWrite(Lane1[2], LOW);
 digitalWrite(Lane3[0], LOW);
 digitalWrite(Lane1[1], HIGH);
 digitalWrite(Lane3[1], HIGH);
 delay(3000);
 digitalWrite(Lane1[1], LOW);
 digitalWrite(Lane3[1], LOW);
 digitalWrite(Lane1[0], HIGH);
 digitalWrite(Lane3[2], HIGH);
 delay(7000);
 digitalWrite(Lane3[2], LOW);
 digitalWrite(Lane4[0], LOW);
 digitalWrite(Lane3[1], HIGH);
 digitalWrite(Lane4[1], HIGH);
 delay(3000);
 digitalWrite(Lane3[1], LOW);
 digitalWrite(Lane4[1], LOW);
 digitalWrite(Lane3[0], HIGH);
 digitalWrite(Lane4[2], HIGH);
 delay(7000);
 digitalWrite(Lane4[2], LOW);
 digitalWrite(Lane2[0], LOW);
 digitalWrite(Lane4[1], HIGH);
 digitalWrite(Lane2[1], HIGH);
 delay(3000);
 digitalWrite(Lane4[1], LOW);
 digitalWrite(Lane2[1], LOW);
 digitalWrite(Lane4[0], HIGH);
 digitalWrite(Lane2[2], HIGH);
 delay(7000);
 digitalWrite(Lane1[0], LOW);
 digitalWrite(Lane2[2], LOW);
 digitalWrite(Lane1[1], HIGH);
 digitalWrite(Lane2[1], HIGH);
 delay(3000);
 digitalWrite(Lane2[1], LOW);
 digitalWrite(Lane1[1], LOW); }
```

# Schematic 2- TINKERCAD

## SMART Traffic Light Control System using Ultrasonic Sensor with Arudino and TinkerCAD




Simulation Component Required:
LED 6, Resistors: 6 Nos 220ohms, Ultrasonic 2Nos

# TINKER CAD PROGRAM 2

```
int red1=8;
int green1=6;
int yellow1=7;
int red2=3;
int green2=5;
int yellow2=4;
const int pingPin = 10; // Trigger Pin of Ultrasonic
Sensor
const int echoPin = 9; // Echo Pin of Ultrasonic Sensor
const int pingPin2 = 12; // Trigger Pin of Ultrasonic
Sensor
const int echoPin2 = 11; // Echo Pin of Ultrasonic
Sensor
void setup() {
  // put your setup code here, to run once:
pinMode(red1,OUTPUT);
pinMode(red2,OUTPUT);
pinMode(yellow1,OUTPUT);
pinMode(yellow2,OUTPUT);
pinMode(green1,OUTPUT);
pinMode(green2,OUTPUT);
}
void loop() {
  // put your main code here, to run
repeatedly:
  int distance1,distance2;
  distance1=calculatedistance(pingPin ,
echoPin);
  distance2=calculatedistance(pingPin2
, echoPin2);
 if(distance1>=distance2){
    digitalWrite(red1,LOW);
digitalWrite(green2,LOW);
digitalWrite(red2,LOW);
digitalWrite(green1,LOW);
digitalWrite(yellow1,HIGH);
digitalWrite(yellow2,HIGH);
delay(200);
   while(distance1>distance2){
  distance1=calculatedistance(pingPin
, echoPin);
distance2=calculatedistance(pingPin2
, echoPin2);
digitalWrite(red1,HIGH);
digitalWrite(green2,HIGH);
digitalWrite(red2,LOW);
digitalWrite(green1,LOW);
digitalWrite(yellow1,LOW);
digitalWrite(yellow2,LOW);
  } }


if(distance2>distance1){
   digitalWrite(red1,LOW);
digitalWrite(green2,LOW);
digitalWrite(red2,LOW);
digitalWrite(green1,LOW);
digitalWrite(yellow1,HIGH);
digitalWrite(yellow2,HIGH);
delay(200);
 while(distance2>distance1){
```

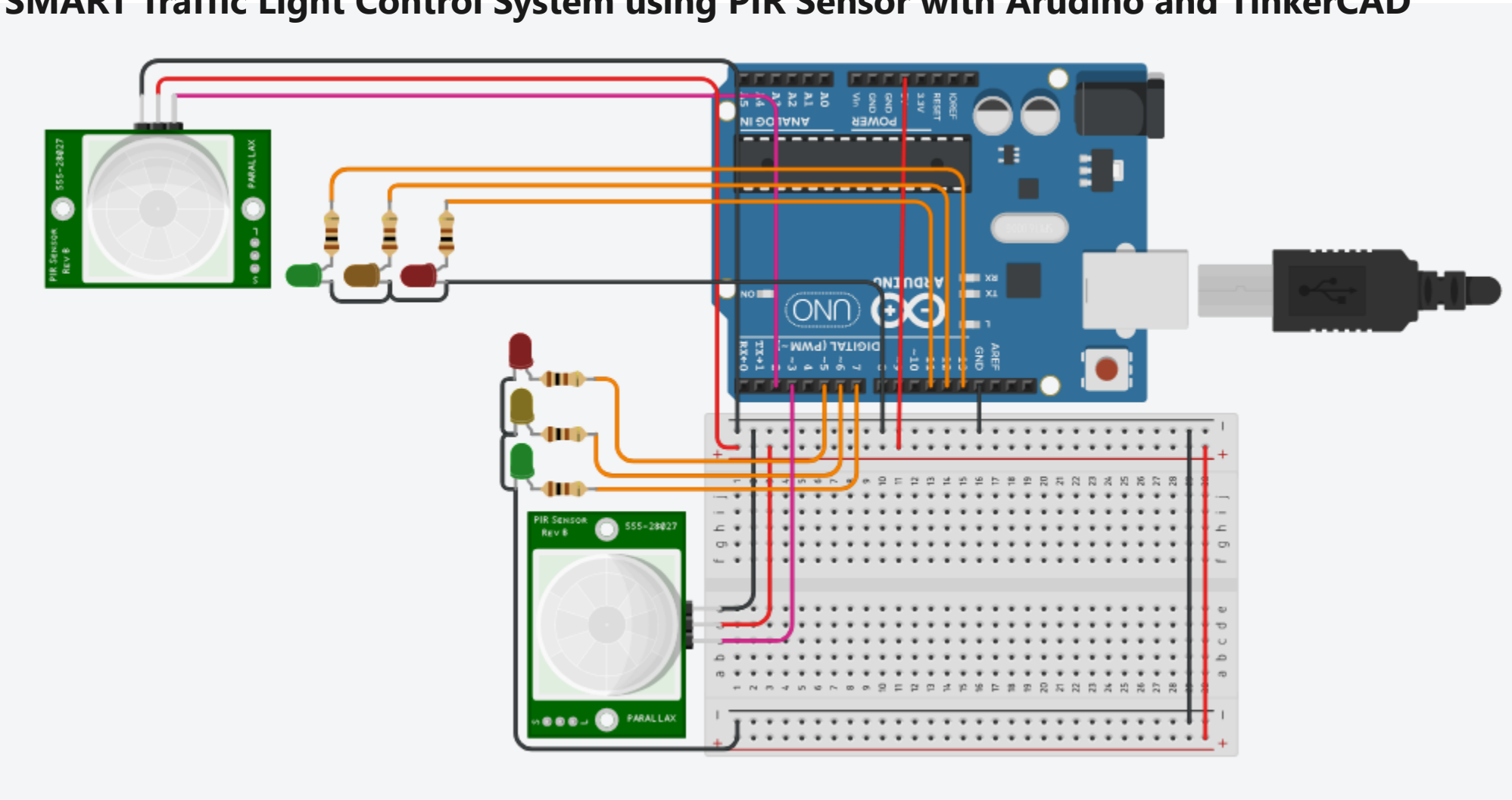
# TINKER CAD PROGRAM 2 Contd,,

```
 distance1=calculatedistance(pingPin , echoPin);
  distance2=calculatedistance(pingPin2 , echoPin2);
digitalWrite(red1,LOW);
digitalWrite(green2,LOW);
digitalWrite(red2,HIGH);
digitalWrite(green1,HIGH);
digitalWrite(yellow1,LOW);
digitalWrite(yellow2,LOW);
  }


}

}
```

```
long microsecondsToCentimeters(long
microseconds)
{
   return microseconds / 29 / 2;
}

int calculatedistance(int pingPin , int echoPin){
    long duration, inches, cm,meter;
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(pingPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  cm = microsecondsToCentimeters(duration);
  meter = cm/100;
  return meter;
}
```

# Schematic 3- TINKERCAD

## SMART Traffic Light Control System using PIR Sensor with Arudino and TinkerCAD




Simulation Component Required:
LED 6, Resistors: 6 Nos 220ohms, PIR 2Nos sensor

# TINKER CAD PROGRAM 3

```
// Pin sensor Pir
int sh = 2;
int sv = 3;

// Valores de los sensores
int valh = 0;
int valv = 0;

// Pin semaforo vertical
int rv = 5;
int av = 6;
int vv = 7;

// Pin semaforo horizontal
int rh = 11;
int ah = 12;
int vh = 13;

void setup()
{
  for (int i = 5; i <= 13; i++) {
            pinMode(i, OUTPUT);
  }
  pinMode(sh, INPUT);
  pinMode(sv, INPUT);
  Serial.begin(9600);
}

void loop()
{
  valh = digitalRead(sh);
  valv = digitalRead(sv);

  Serial.print("Horizontal: ");
  Serial.println(valh);
  Serial.print("Vertical: ");
  Serial.println(valv);
  Serial.println();

  int th1 = 3000;
  int th2 = 400;

  int tv1 = 3000;
  int tv2 = 400;

  if (valh == HIGH && valv == LOW) {
            Serial.println("Aumentar
Horizontal");
    th1 = th1 * 2;
    th2 = th2 * 2;
  } else if (valh == LOW && valv ==
HIGH) {
    Serial.println("Aumentar Vertical");
    tv1 = tv1 * 2;
            tv2 = tv2 * 2;
  }

  semaforoHorizontal(th1, th2);
  semaforoVertical(tv1, tv2);
}
```

# TINKER CAD PROGRAM 3 Contd,,

```
void semaforoHorizontal(int t1, int t2) {
  digitalWrite(rv, HIGH);
  digitalWrite(vh, HIGH);
  delay(t1);
  digitalWrite(vh, LOW);
  digitalWrite(ah, HIGH);
  delay(t2);
  digitalWrite(rv, LOW);
  digitalWrite(ah, LOW);
}
```

```
void semaforoVertical(int t1, int t2) {
  digitalWrite(rh, HIGH);
  digitalWrite(vv, HIGH);
  delay(t1);
  digitalWrite(vv, LOW);
  digitalWrite(av, HIGH);
  delay(t2);
  digitalWrite(rh, LOW);
  digitalWrite(av, LOW);
}
```

# BECE351E – INTERNET OF THINGS

**EXPT NO: 7**

**DATE: 21 June 2023**

**NAME: Syed Urooj Ul Hasan**

**REG. NO.: 21BRS1244**

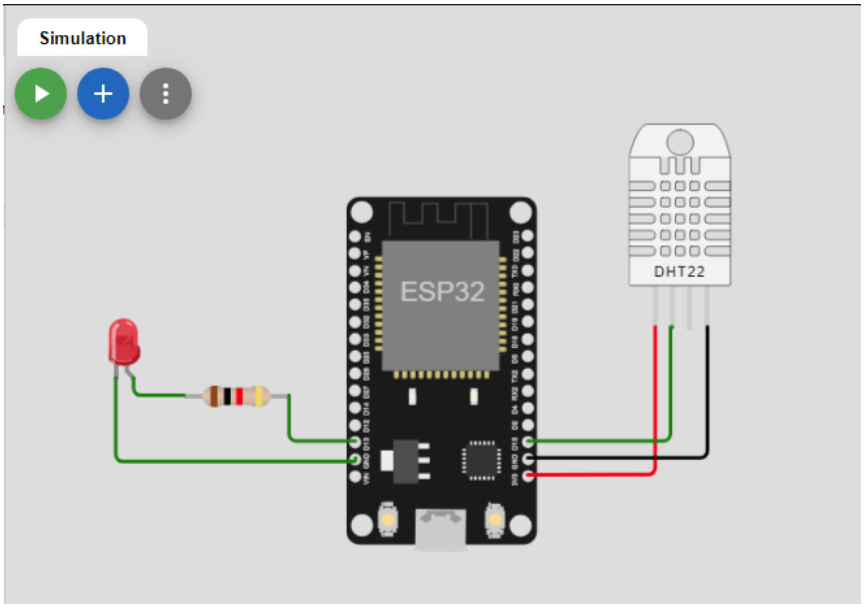
**AIM:** Real-Time Environmental monitoring using Arduino Uno WiFi Rev 2 with DHT11 and ThingSpeak Cloud Computing

## ESP32 Peripherals and I/O

48 GPIO pins in total, only 25 of them are broken out to the pin headers on both sides of the development board. These pins can be assigned a variety of peripheral duties, including:

| | |
|---|---|
| **15 ADC channels** | ***15 channels of 12-bit SAR ADC with selectable ranges of 0-1V, 0-1.4V, 0-2V, or 0-4V*** |
| **2 UART interfaces** | ***2 UART interfaces with flow control and IrDA support*** |
| **25 PWM outputs** | ***25 PWM pins to control things like motor speed or LED brightness*** |
| **2 DAC channels** | ***Two 8-bit DACs to generate true analog voltages*** |
| **SPI, I2C and I2S interface** | ***Three SPI and one I2C interfaces for connecting various sensors and peripherals, as well as two I2S interfaces for adding sound to your project*** |
| **9 Touch Pads** | ***9 GPIOs with capacitive touch sensing*** |

## Circuit:

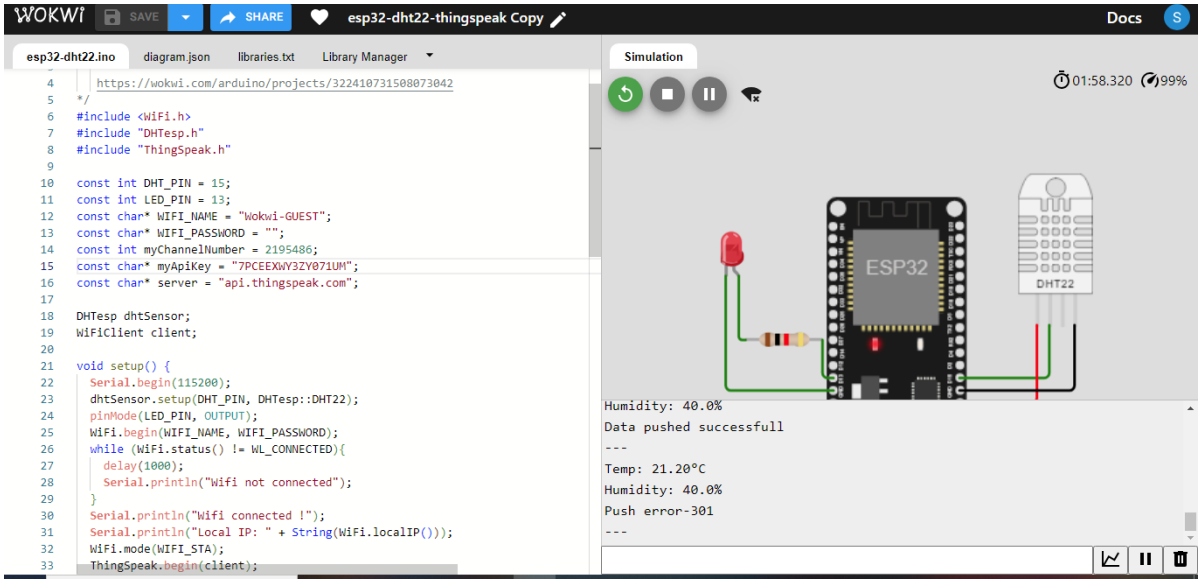



## Code:

```
#include <WiFi.h>
#include "DHTesp.h"
#include "ThingSpeak.h"

const int DHT_PIN = 15;
const int LED_PIN = 13;
const char* WIFI_NAME = "Wokwi-GUEST";
const char* WIFI_PASSWORD = "";
const int myChannelNumber = 2195488;
const char* myApiKey = "7BSUF9FY08KPS5OG";
const char* server = "api.thingspeak.com";

DHTesp dhtSensor;
WiFiClient client;

void setup() {
  Serial.begin(115200);
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  pinMode(LED_PIN, OUTPUT);
  WiFi.begin(WIFI_NAME, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED){
    delay(1000);
    Serial.println("Wifi not connected");
  }
  Serial.println("Wifi connected !");
```

```
  Serial.println("Local IP: " + String(WiFi.localIP()));
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);
}

void loop() {
  TempAndHumidity  data = dhtSensor.getTempAndHumidity();
  ThingSpeak.setField(1,data.temperature);
  ThingSpeak.setField(2,data.humidity);
  if (data.temperature > 35 || data.temperature < 12 || data.humidity > 70 ||
data.humidity < 40) {
    digitalWrite(LED_PIN, HIGH);
  }else{
    digitalWrite(LED_PIN, LOW);
  }

  int x = ThingSpeak.writeFields(myChannelNumber,myApiKey);

  Serial.println("Temp: " + String(data.temperature, 2) + "°C");
  Serial.println("Humidity: " + String(data.humidity, 1) + "%");

  if(x == 200){
    Serial.println("Data pushed successfull");
  }else{
    Serial.println("Push error" + String(x));
  }
  Serial.println("---");

  delay(10000);
}
```

## Output:





**RESULT:** The outputs were successfully obtained.

# BECE351E – INTERNET OF THINGS

**EXPT NO: 8**

**DATE: 27/06/2023**

**NAME: Syed Urooj Ul Hasan**

**REG. NO.: 21BRS1244**

## Real Time Environmental monitoring using Arduino Uno WiFi Rev 2 with DHT11 and ThinkgSpeak Cloud Computing

## AIM

To perform and analysis Real-Time Environmental monitoring using Arduino Uno WiFi Rev 2 with DHT11 and ThingSpeak Cloud Computing in Arduino.

## BLOCKS/COMPONENTS REQUIRED

Tinker CAD Block Modules

1. DHT Sensor
2. Wifinina
3. ThingSpeak

Components:

1. DHT11
2. Aurdino uno

## ALGORITHM

Steps to perform real-time environmental monitoring using Arduino Uno WiFi

Rev 2, DHT11 sensor, and ThingSpeak in a summarized form:

1. Set up the Arduino Uno WiFi Rev 2: Connect the Arduino Uno WiFi Rev 2 to your computer using a USB cable and install the necessary drivers and software (Arduino IDE).

2. Connect the DHT11 sensor: Connect the DHT11 sensor to the Arduino Uno WiFi Rev 2 using three wires: VCC to 5V, GND to GND, and DATA to a digital pin (e.g., Pin 2).

3. Install required libraries: In the Arduino IDE, go to "Sketch" -> "Include Library" -> "Manage Libraries" and search for "DHT sensor library" and "WiFiNINA." Install these libraries.

4. Set up the ThingSpeak account: Create an account on ThingSpeak (https://thingspeak.com) and create a new channel to store the sensor data.

5. Connect to Wi-Fi: In your Arduino code, include the WiFiNINA library and configure the Wi-Fi credentials (SSID and password). Use the `WiFi.begin()` function to connect to your Wi-Fi network.

6. Read sensor data: Use the DHT sensor library to read temperature and humidity values from the DHT11 sensor. Store these values in variables.

7. Create a ThingSpeak update: Use the `ThingSpeak.writeField()` function to update the ThingSpeak channel with the sensor data. Pass in the channel API key and the field number to update along with the sensor values.

8. Repeat the process: Use a delay to set the interval at which you want to update the sensor data. You can use the `delay()` function to wait for a specific amount of time before repeating the process.

9. Upload the code: Compile the code and upload it to the Arduino Uno WiFi Rev 2 board.

10. Monitor the data: In your ThingSpeak account, navigate to the channel you created. You should be able to see real-time updates of the temperature and humidity data sent by your Arduino.

**SMART CAR PARKING SYSTEM**

1. Sensors: Smart parking systems rely on sensors to detect the availability of parking spaces. Different types of sensors can be used, including ultrasonic sensors, infrared sensors, or magnetic sensors. These sensors are installed in individual parking spots to monitor occupancy.

2. Data Collection: The sensor data is collected and processed to determine the availability of parking spaces in real-time. The data is usually transmitted to a central server or cloud platform for analysis and storage.

3. Communication: Smart parking systems often utilize wireless communication protocols such as Wi-Fi, cellular networks, or Internet of Things (IoT) technologies to transmit data between the sensors, central server, and user applications.

4. Mobile Applications: Users can access smart parking systems through dedicated mobile applications. These applications provide real-time information about parking space availability, allow users to reserve parking spots, and offer navigation guidance to the nearest available spot.

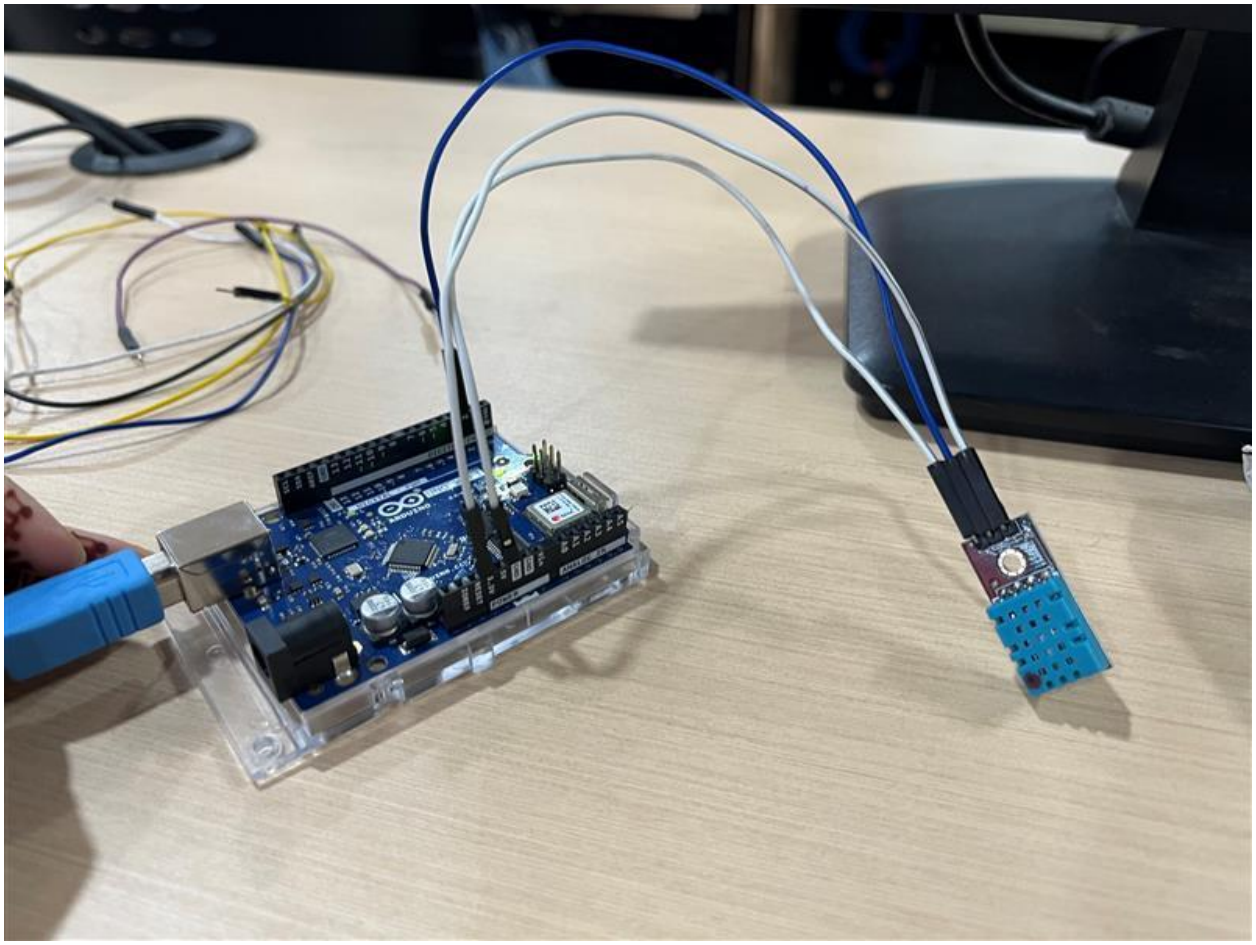
5. Reservation and Payment: Smart parking systems can include features for reserving parking spaces in advance. Users can use mobile applications or other interfaces to reserve a spot and make payments electronically. This helps ensure parking availability upon arrival.

6. Guidance Systems: To assist drivers in finding available parking spaces, smart parking systems often provide guidance through digital signage, LED indicators, or mobile applications. This guidance directs drivers to the nearest available parking spot, reducing the time spent searching for parking.

7. Integration with Smart Cities: Smart car parking systems can be integrated into larger smart city initiatives. By connecting parking data with other urban systems, such as traffic management or public transportation, cities can optimize traffic flow and reduce congestion.

8. Data Analytics: The collected parking data can be analyzed to derive insights and optimize parking operations. It can help identify parking patterns, occupancy trends, and enable predictive analytics to optimize parking space allocation and resource management.

## CIRCUIT



## CODE

(i)

```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {
     Serial.begin(9600);
     dht.begin();
}

void loop() {
```

```
     float h = dht.readHumidity();
     float t = dht.readTemperature();
     float f = dht.readTemperature(true);

     if (isnan(h) || isnan(t) || isnan(f)) {
          Serial.println( "Sensor Error" );
          return;
     }

     float hif = dht.computeHeatIndex(f, h);
     float hic = dht.computeHeatIndex(t, h, false);

     Serial.println( " Humidity: " + String( h ) );
     Serial.println( " Temperature (C): " + String( t ) );
     Serial.println( " Temperature (F): " + String( f ) );
     Serial.println( " Heat index (C): " + String( hic ) );
     Serial.println( " Heat index (F): " + String( hif ) );

     delay(60000);
}
```

(ii)

```
#include <WiFiNINA.h>
#include <DHT.h>
#include "secrets.h"
#include "ThingSpeak.h"
#define DHTPIN 8
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

char ssid[] = "Amma2"; // your network SSID (name)
char pass[] = "8939708340"; // your network password
int keyIndex = 0; // your network key Index number

WiFiClient client;

uint8_t temperature, humidity;
unsigned long myChannelNumber = 2195330;
const char * myWriteAPIKey = "CZOV2GGSW0LP6RH4";

void setup() {
     Serial.begin(115200);
     dht.begin();

     while (!Serial) {
```

```
      ;
      }

      Serial.println();
      Serial.println();
      Serial.print("Connecting to ");
      Serial.println(ssid);

      if (WiFi.status() == WL_NO_MODULE) {
           Serial.println("Communication with WiFi module
      failed!");
      while (true);
      }

      ThingSpeak.begin(client);
}

void loop() {
      if(WiFi.status() != WL_CONNECTED){
           Serial.print("Attempting to connect to SSID: ");

           Serial.println(ssid);

           while(WiFi.status() != WL_CONNECTED){

                WiFi.begin(ssid, pass); // Connect to WPA/WPA2
                network. Change this line if
                using open or WEP network

                Serial.print(".");

                delay(5000);

           }
           Serial.println("\nConnected.");

      }
      temperature = dht.readTemperature();
      humidity = dht.readHumidity();
      Serial.print("Temperature Value is :");
      Serial.print(temperature);
      Serial.println("C");
      Serial.print("Humidity Value is :");
      Serial.print(humidity);
      Serial.println("%");
```

```
    ThingSpeak.writeField(myChannelNumber, 1, temperature,
    myWriteAPIKey);
    ThingSpeak.writeField(myChannelNumber, 2, humidity,
    myWriteAPIKey);

    delay(1000);
}
```

## OUTPUT

```
Connecting to POCO X3 Pro
Attempting to connect to SSID: POCO X3 Pro
.
Connected.
Temperature Value is :23C
Humidity Value is :72%
Temperature Value is :23C
Humidity Value is :71%
Temperature Value is :29C
Humidity Value is :98%
Temperature Value is :31C
Humidity Value is :98%
Temperature Value is :29C
Humidity Value is :98%
Temperature Value is :28C
Temperature Value is :24C
Humidity Value is :96%
Temperature Value is :24C
Humidity Value is :95%
Temperature Value is :24C
Humidity Value is :94%
Temperature Value is :24C
Humidity Value is :93%
Temperature Value is :24C
Humidity Value is :91%
Temperature Value is :23C
Humidity Value is :89%
Temperature Value is :23C
Humidity Value is :88%
```

**INFERENCE**

In this Interface, you need to replace “YourWiFiSSID” and “YourWiFiPassword” with the credentials of your WiFi network. Also, replace “YourThingSpeakAPIKey” with your actual ThingSpeak API key, and make sure to set “YOUR_CHANNEL_ID” to the ID of your specific ThingSpeak channel.

**RESULT**

Hence, we performed the experiment using Arduino Hardware.