Faculty of Engineering and Technology
Department of Electrical and Computer Engineering

ENCS 4110
COMPUTER DESIGN LABORATORY
Term Project

"SMART HOME AUTOMATION PROJECT"

Prepared by:

Nsreen Tawafsha – 1182319

Supervised by:

Dr. Ahmad Afaneh

Teacher Assistant:

Eng. Heba Qadah

Section: 4

Date: 21 May 2021

1. Abstract

The aim of this project is to create a smart home automation in which, if someone is close to the door (servo motor), the door will be opened, and if the surrounding temperature is greater than 30 degrees, the fan will be turned on. In addition, if there is motion but no light, the bulb will be turned on.

All pervious processes done either manually by the human or automatically by the system. So that to comprehend how to use and control these components.

Table of Content

2. Brief background about the components and their usage.

2.1 Ultrasonic Sensor:-

Ultrasonic (US) sensor in **Figure 2.1.1** is a 4 pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The sensor works with the simple high school formula that Distance = Speed × Time [1].

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor [1].

HC-SR04 distance sensor is commonly used with both microcontroller and microprocessor platforms like Arduino, ARM, PIC, Raspberry Pie etc. The following guide is universally since it has to be followed irrespective of the type of computational device used [1].

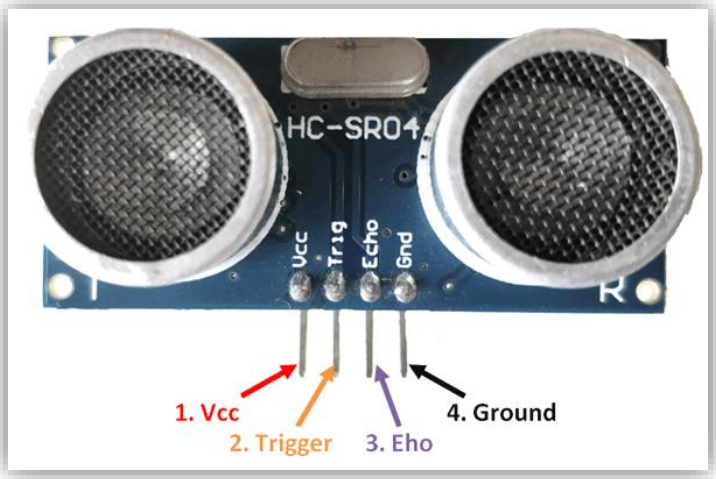


Figure 2.1.1: Ultrasonic Sensor HC SR04 Pin Diagram

✓ **Ultrasonic Sensor Pin Configuration:**

Pin Number	Pin Name	Description
1	Vcc	The Vcc pin powers the sensor, typically with +5V
2	Trigger	Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave.
3	Echo	Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor.
4	Ground	This pin is connected to the Ground of the system.

2.2 Servo meter:-

A servomotor in **Figure 2.2.1** is a linear actuator or rotary actuator that allows for precise control of linear or angular position, acceleration, and velocity. It consists of a motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors [2].

The main reason behind using a servo is that it provides angular precision, i.e. it will only rotate as much we want and then stop and wait for the next signal to take further action. The servo motor is unlike a standard electric motor which starts turning as when we apply power to it, and the rotation continues until we switch off the power. We cannot control the rotational progress of electrical motor, but we can only control the speed of rotation and can turn it ON and OFF. Small servo motors are included many beginner Arduino starter kits, as they are easy to operate as part of a small electronics projects [2].

Servo motors have three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino board. The ground wire is typically black or brown and should be connected to a ground pin on the Arduino board. The signal pin is typically yellow, orange or white and should be connected to a digital pin on the Arduino board. Note that servos draw considerable power, so if you need to drive more than one or two, you'll probably need to power them from a separate supply (i.e. not the 5V pin on your Arduino). Be sure to connect the grounds of the Arduino and external power supply together [3].

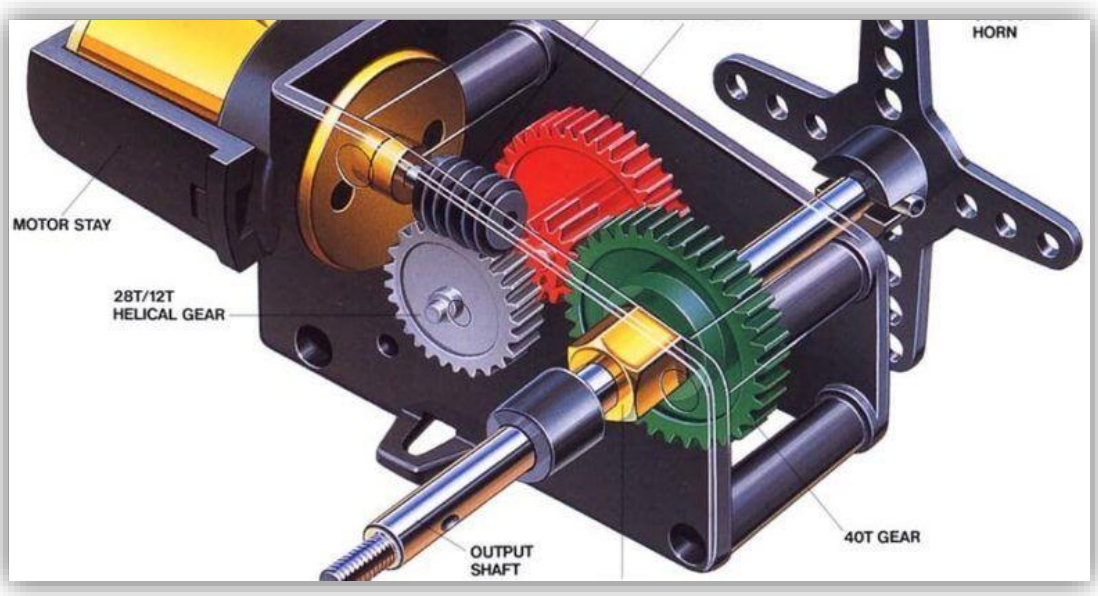


Figure 2.1.2: Servo Meter

2.3 Light Dependent Resistor (LDR):-

Light dependent resistors, LDRs or photoresistors as shown in **Figure 2.2.3** are often used in electronic circuit designs where it is necessary to detect the presence or the level of light [4].

A photoresistor or light dependent resistor is an electronic component that is sensitive to light. When light falls upon it, then the resistance changes. Values of the resistance of the LDR may change over many orders of magnitude the value of the resistance falling as the level of light increases [4].

These electronic components can be described by a variety of names from light dependent resistor, LDR, photoresistor, or even photo cell, photocell or photoconductor [4].

Although other electronic components such as photodiodes or photo-transistor can also be used, LDRs or photo-resistors are a particularly convenient to use in many electronic circuit designs. They provide large change in resistance for changes in light level [4].

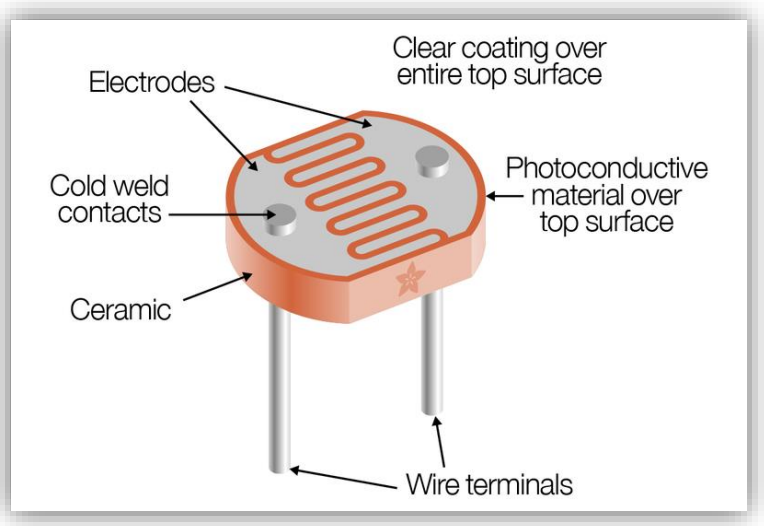


Figure 2.3.1: Light Dependent Resistor

✓ Types of photoresistor:

Light dependent resistors, LDRs or photoresistors fall into one of two types or categories:

- **Intrinsic photoresistors:** Intrinsic photoresistors use un-doped semiconductor materials including silicon or germanium. Photons fall on the LDR excite electrons moving them from the valence band to the conduction band. As a result, these electrons are free to conduct electricity. The more light that falls on the device, the more electrons are liberated and the greater the level of conductivity, and this results in a lower level of resistance.
- **Extrinsic photoresistors:** Extrinsic photoresistors are manufactured from semiconductor of materials doped with impurities. These impurities or dopants create a new energy band above the existing valence band. As a result, electrons need less energy to transfer to the conduction band because of the smaller energy gap.

2.4 Passive Infrared Sensor (PIR):-

PIR sensors allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensors range. They are small, inexpensive, low-power, easy to use and don't wear out. For that reason they are commonly found in appliances and gadgets used in homes or businesses. They are often referred to as PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors [5].

PIRs are basically made of a pyroelectric sensor (which you can see in **Figure 2.4.1** below as the round metal can with a rectangular crystal in the center), which can detect levels of infrared radiation. Everything emits some low level radiation, and the hotter something is, the more radiation is emitted. The sensor in a motion detector is actually split in two halves. The reason for that is that we are looking to detect motion (change) not average IR levels. The two halves are wired up so that they cancel each other out. If one half sees more or less IR radiation than the other, the output will swing high or low [5].

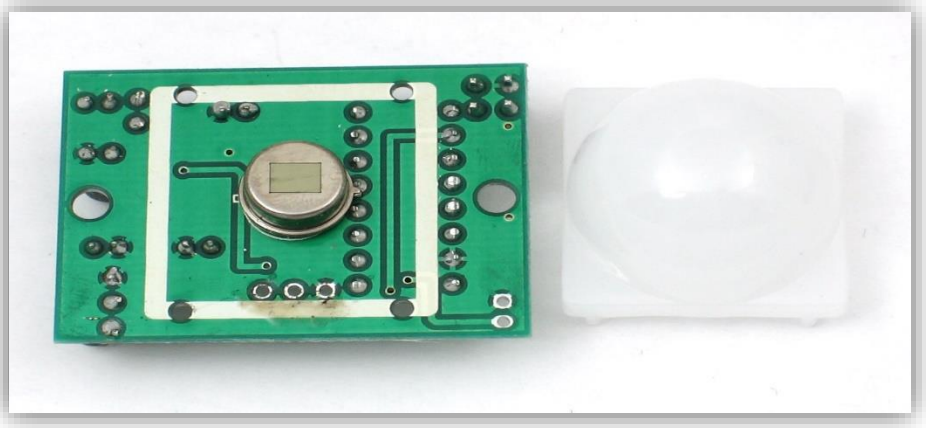


Figure 2.4.1: PIR

Our new PIRs have more adjustable settings and have a header installed in the 3-pin ground/out/power pads as seen in **Figure 2.4.2**

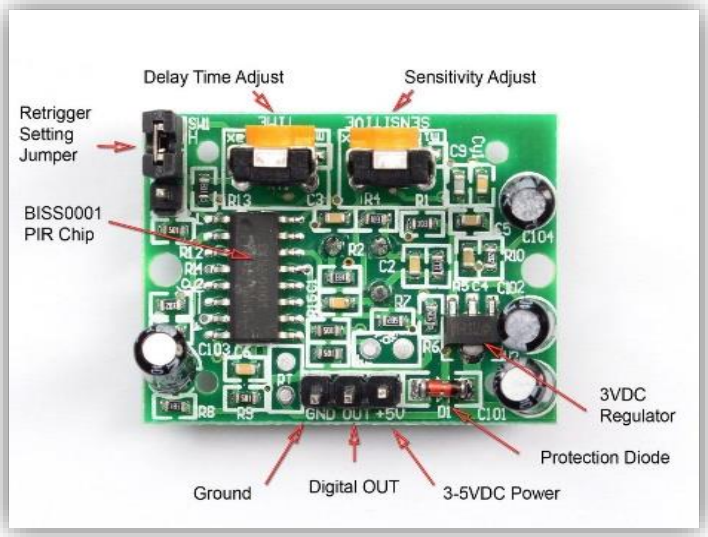


Figure 2.2.4: PIR Layout

2.5 Bulb:-

An incandescent light bulb, incandescent lamp or incandescent light globe as seen in is **Figure 2.5.1** an electric light with a wire filament heated until it glows. The filament is enclosed in a glass bulb with a vacuum or inert gas to protect the filament from oxidation. Current is supplied to the filament by terminals or wires embedded in the glass. A bulb socket provides mechanical support and electrical connections [6].

Most light bulbs have either clear or coated glass. Coated glass bulbs have kaolin clay blown in and electrostatically deposited on the interior of the bulb. The powder layer diffuses the light from the filament. Pigments may be added to the clay to adjust the color of the light emitted. Kaolin diffused bulbs are used extensively in interior lighting because of their comparatively gentle light. Other kinds of colored bulbs are also made, including the various colors used for "party bulbs", Christmas tree lights and other decorative lighting. These are created by coloring the glass with a dopant; which is often a metal like cobalt (blue) or chromium (green). Neodymium-containing glass is sometimes used to provide a more natural-appearing light [6].

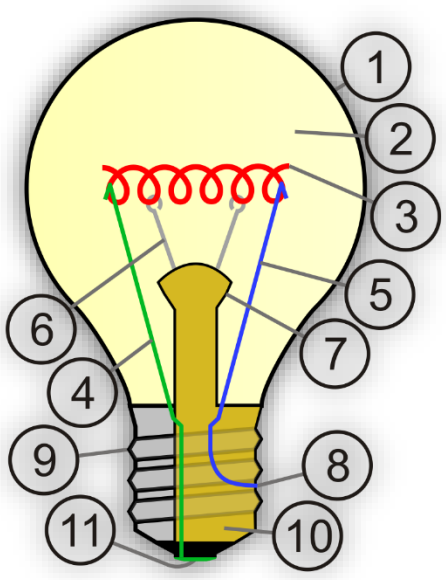


Figure 2.5.1: Bulb

1) Outline of Glass bulb.
2) Low pressure inert gas (argon, nitrogen, krypton, xenon)
3) Tungsten filament
4) Contact wire (goes out of stem)
5) Contact wire (goes into stem)
6) Support wires (one end embedded in stem; conduct no current)
7) Stem (glass mount)
8) Contact wire (goes out of stem)
9) Cap (sleeve)
10) Insulation (vitrite)
11) Electrical contact

2.6 Temperature sensor:-

A temperature sensor is an electronic device that measures the temperature of its environment and converts the input data into electronic data to record, monitor, or signal temperature changes. There are many different types of temperature sensors. Some temperature sensors require direct contact with the physical object that is being monitored (contact temperature sensors), while others indirectly measure the temperature of an object (non-contact temperature sensors). **Figure 2.6.1** shows the pins of the temperature sensor [7].

- Contact Temperature Sensor Types :

  These types of temperature sensor are required to be in physical contact with the object being sensed and use conduction to monitor changes in temperature. They can be used to detect solids, liquids or gases over a wide range of temperatures.
- Non-contact Temperature Sensor Types:

  These types of temperature sensor use convection and radiation to monitor changes in temperature. They can be used to detect liquids and gases that emit radiant energy as heat rises and cold settles to the bottom in convection currents or detect the radiant energy being transmitted from an object in the form of infra-red radiation (the sun). Non-contact temperature sensors are usually infrared (IR) sensors. They remotely detect the IR energy emitted by an object and send a signal to a calibrated electronic circuit that determines the object's temperature.

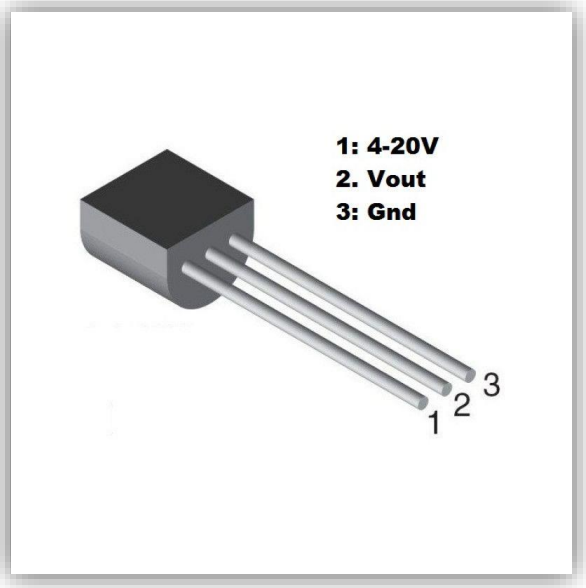


Figure 2.6.1: Temperature sensor

2.7 DC Motor:-

Electrical DC Motors are continuous actuators that convert electrical energy into mechanical energy. The DC motor achieves this by producing a continuous angular rotation that can be used to rotate pumps, fans, compressors, wheels, etc [8].

The DC Motor or Direct Current Motor to give it its full title, is the most commonly used actuator for producing continuous movement and whose speed of rotation can easily be controlled, making them ideal for use in applications were speed control, servo type control, and/or positioning is required. A DC motor consists of two parts, a "Stator" which is the stationary part and a "Rotor" which is the rotating part. The result is that there are basically three types of DC Motor available [8].

- Brushed Motor (as seen in **Figure 2.7.1**)
- Brushless Motor
- Servo Motor

Normal DC motors have almost linear characteristics with their speed of rotation being determined by the applied DC voltage and their output torque being determined by the current flowing through the motor windings. The speed of rotation of any DC motor can be varied from a few revolutions per minute (rpm) to many thousands of revolutions per minute making them suitable for electronic, automotive or robotic applications. By connecting them to gearboxes or gear-trains their output speed can be decreased while at the same time increasing the torque output of the motor at a high speed [8].

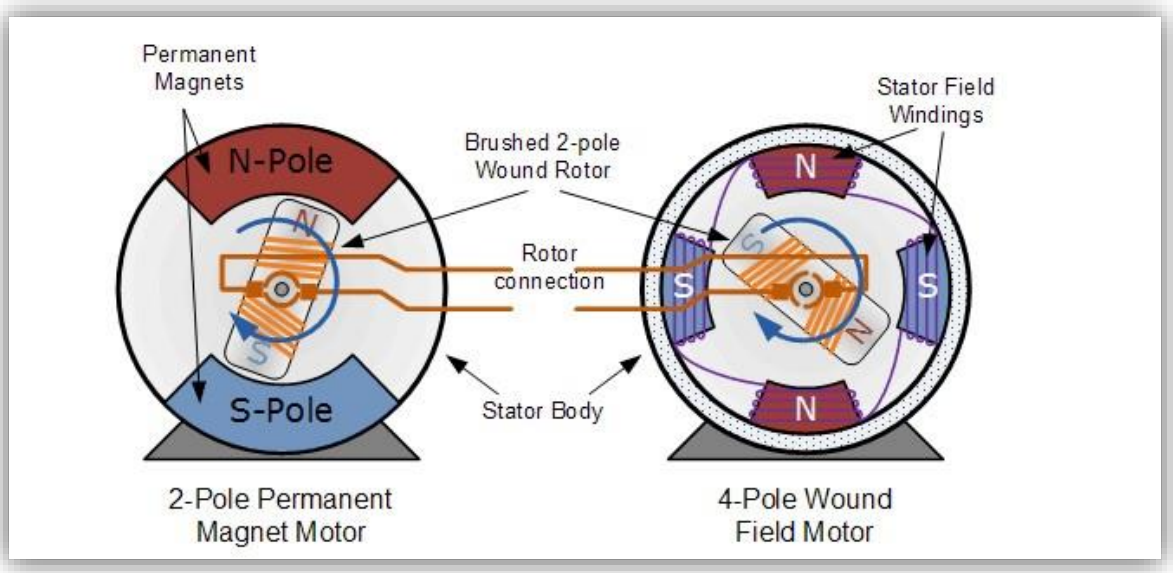


Figure 2.7.1: Conventional (Brushed) DC Motor

2.8 Arduino Switch:-

Switches are used to turn ON/OFF devices and to connect different parts of a circuit. The slide-switch in Arduino moves the slider of the switch from the open position (ON) to the closed position (OFF) [9].

It allows the flow of current in the circuit without the need for splice wire. The slide switches are widely used in small circuits applications [9].

✓ Types of Switches :

There are major four types of switches in Arduino, which are listed below:

- **SPST (Single Pole Single Throw) Switch:**
  It is a switch that has one input and one output. The circuit is ON when the switch is closed and vice versa as seen in **Figure 2.8.1**

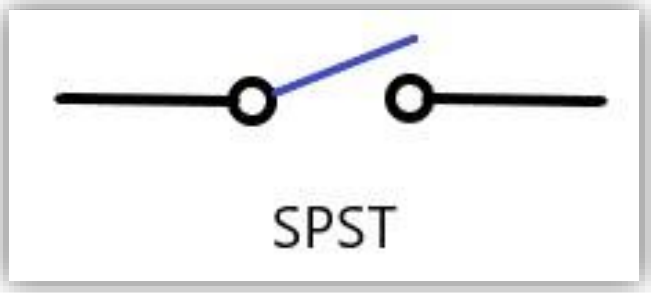


Figure 2.8.1: SPST

- **SPDT (Single Pole Double Throw) Switch:**
  It is a three-terminal switch. It has a single input, which can switch between two outputs as seen in **Figure 2.8.2**

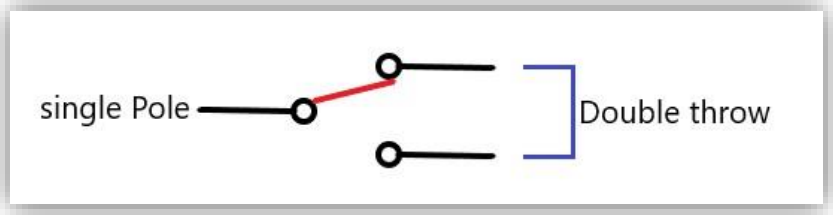


Figure 2.8.1: SPDT

- **SP3T (Single Pole Three Throw) Switch:**
  It is a switch with one input and three outputs, where each input corresponds to any of the output in a circuit.as seen in **Figure 2.8.3**

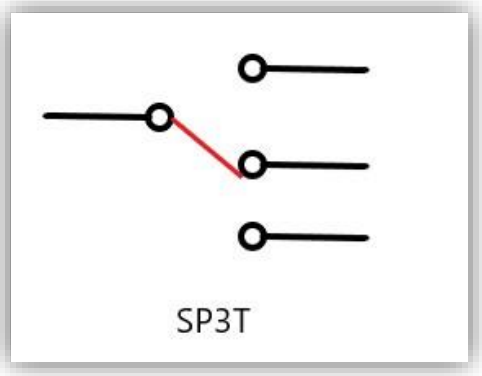


Figure 2.8.1: SP3T

- **DPDT (Double Pole Double Throw) Switch:**
  It is a switch with two inputs and four outputs. Each input of a switch in Arduino can be connected to either of the two outputs as seen in **Figure 2.8.4**

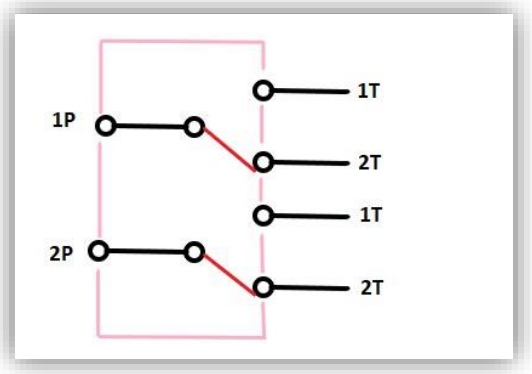


Figure 2.8.4: DPDT

Where,

- Pole: It signifies the number of circuits the switch can control.
- Throw: It signifies the number of positions in which each pole of the switch can connect to it.

2.9 Remote Control:-

Infrared (IR) communication is a widely used and easy to implement wireless technology that has many useful applications. The most prominent examples in day to day life are TV/video remote controls, motion sensors, and infrared thermometers [10].

There are plenty of interesting Arduino projects that use IR communication too. With a simple IR transmitter and receiver, you can make remote controlled robots, distance sensors, heart rate monitors, DSLR camera remote controls, TV remote controls, and lots more [10].

A typical infrared communication system requires an IR transmitter and an IR receiver. The transmitter looks just like a standard LED, except it produces light in the IR spectrum instead of the visible spectrum [10].

Each time you press a button on the remote control, a unique hexadecimal code is generated. This is the information that is modulated and sent over IR to the receiver. In order to decipher which key is pressed, the receiving microcontroller needs to know which code corresponds to each key on the remote [10]. Remote Control seen in **Figure 2.9.1**

Figure 2.9.1: IR REMOTE AND RECEIVER

2.10 LED:-

Light-emitting diodes, or LEDs, are widely used as a standard source of light in electrical equipment. It has a wide array of applications ranging from your mobile phone to large advertising billboards. They find applications in devices for showing what the time is and for displaying different types of data. In this post, the main focus would be on learning a lot about LEDs, such as its operations and functions [11].

A light releasing diode is an electric component that emits light when the electric current flows through it. It is a light source based on semiconductors. When current passes through the LED, the electrons recombine with holes emitting light in the process. It is a specific type of diode having similar characteristics as the p-n junction diode. This means that an LED allows the flow of current in its forward direction while it blocks the flow in the reverse direction. Light-emitting diodes are built using a weak layer of heavily doped semiconductor material. Based on the semiconductor material used and the amount of doping, an LED will emit a colored light at a particular spectral wavelength when forward biased [11].

✓ LED Symbol:

The symbol is similar to that of the p-n junction diode. The difference between these two symbols is that the two arrows indicate that the diode is emitting the light as seen in **Figure 2.10.1**

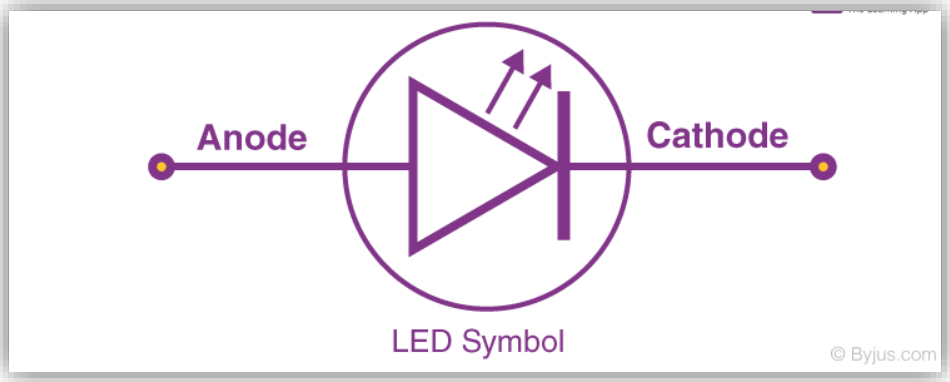


Figure 2.10.1: LED symbol

✓ Uses of LED:

LEDs find applications in various fields, including optical communication, alarm and security systems, remote-controlled operations, robotics, etc. It finds usage in many such areas because of its long-lasting capability, low power requirements, swift response time, and fast switching capabilities. Below are a few standards LED uses:

- Used for TV back-lighting
- Uses in displays
- Used in automotive
- LEDs used in the dimming of lights

2.11 Buzzers:-

An arduino buzzer is also called a piezo buzzer. It is basically a tiny speaker that you can connect directly to an Arduino. You can make it sound a tone at a frequency you set. The buzzer produces sound based on reverse of the piezoelectric effect [12].

The buzzer produces the same noisy sound irrespective of the voltage variation applied to it. It consists of piezo crystals between two conductors. When a potential is applied across these crystals, they push on one conductor and pull on the other. This, push and pull action, results in a sound wave. Most buzzers produce sound in the range of 2 to 4 kHz [12].

Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke. Buzzer seen in **Figure 2.11.1**

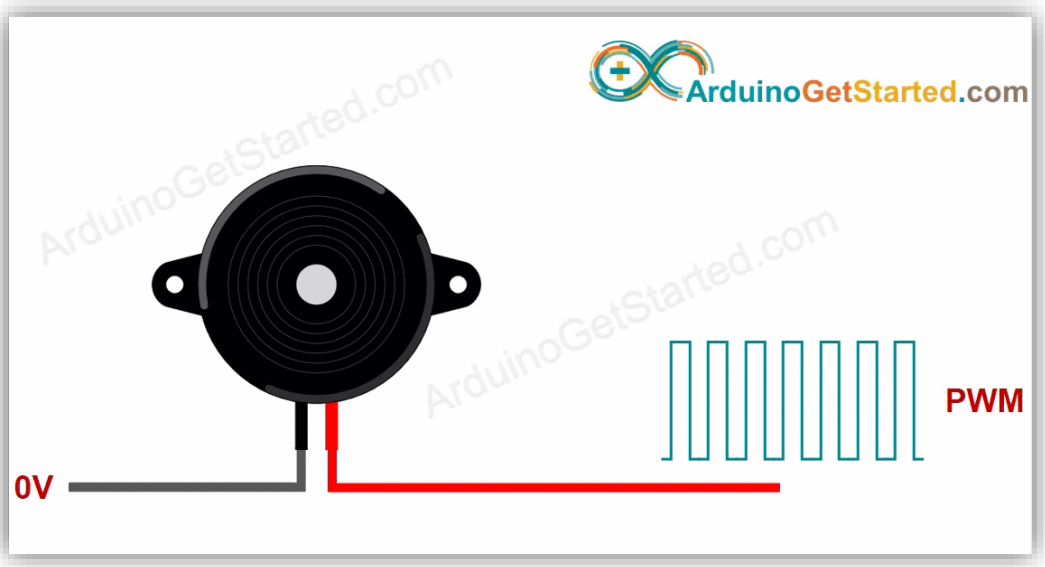


Figure 2.11.1: Buzzer

3. Design and implementation

Figure 3.3.1 shows the implementation of the whole system.

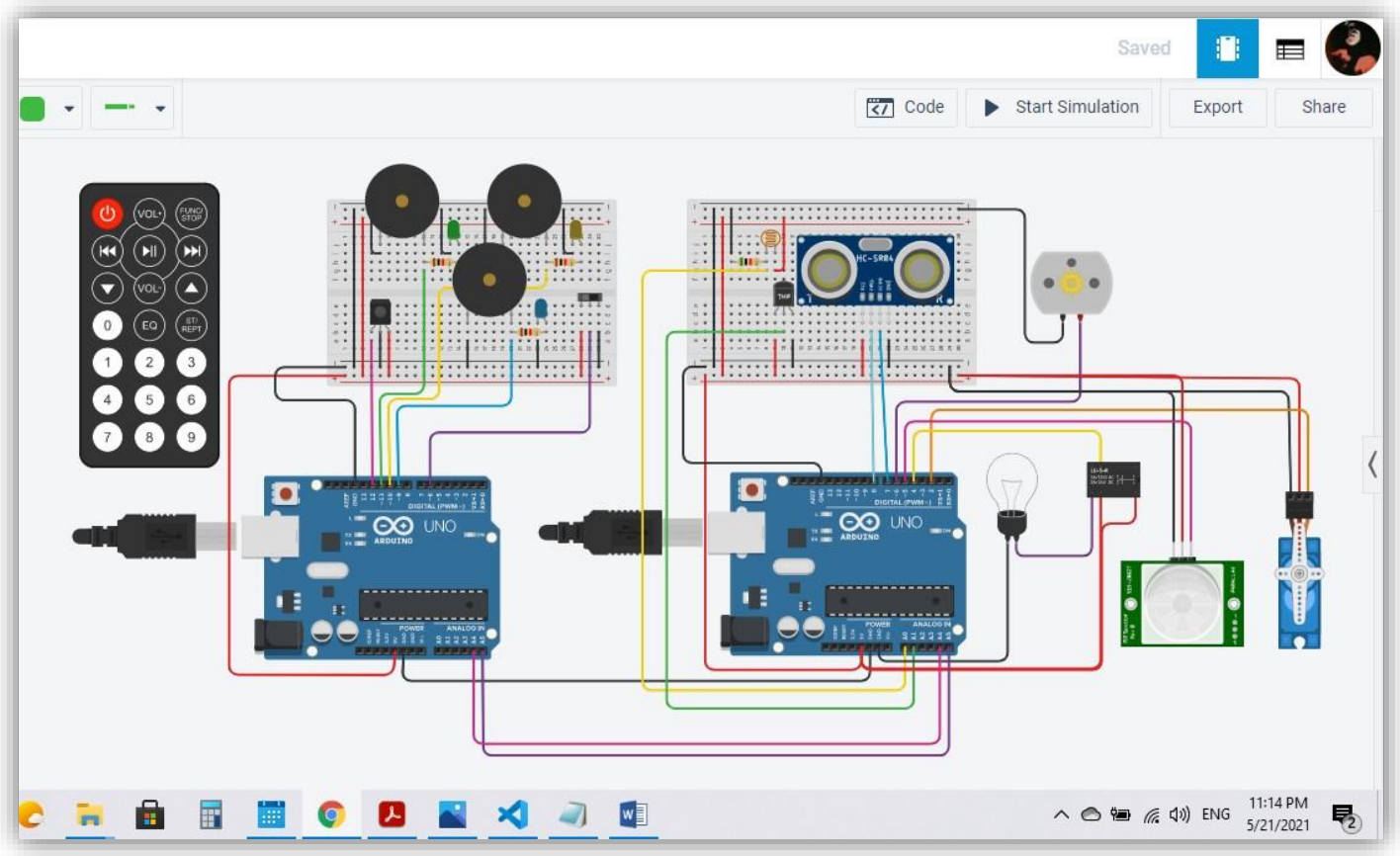


Figure 3.3.1: SMART HOME AUTOMATION PROJECT

Here we receive different alarms when system works in manual mode, the green led and its buzzer to represent the situation about the DC motor (FAN) of Arduino #1, the blue led and its buzzer to represent the situation about the servo meter (Door) of Arduino #1, and the yellow led and its buzzer to represent the situation about the Bulb of Arduino #1, we control all previous components using the Remote control connected with Arduino #2.

When pressing 1 at the remote control, the DC motor will turn on, whereas when pressing 2, it will turn off. The same process applied at the door, when pressing 3 at the remote control it will open, while when pressing 4 it will close. In addition when pressing 5 on the remote control, the bulb will turn on for 1 minute, whereas when pressing on 6, the bulb will turn off.

However, if we are in the automatic mode, then all processes applied to the DC motor, the door, and the bulb will be automatically done by Arduino #1, so the door will be opened if there is someone in distance < 50 and it will checked every 3 seconds by the system, the same as in the DC motor, if the temperature > 30 then the DC motor will be turned on and will go faster as the temperature increases, all done by the system itself. Finally, the system will turned the bulb on for 1 minute if there is a motion and there is no enough light.

The code of this complete system written in Appendix A for the Arduino #1 board, and in Appendix B for the Arduino #2, both codes included an explanation of each step.

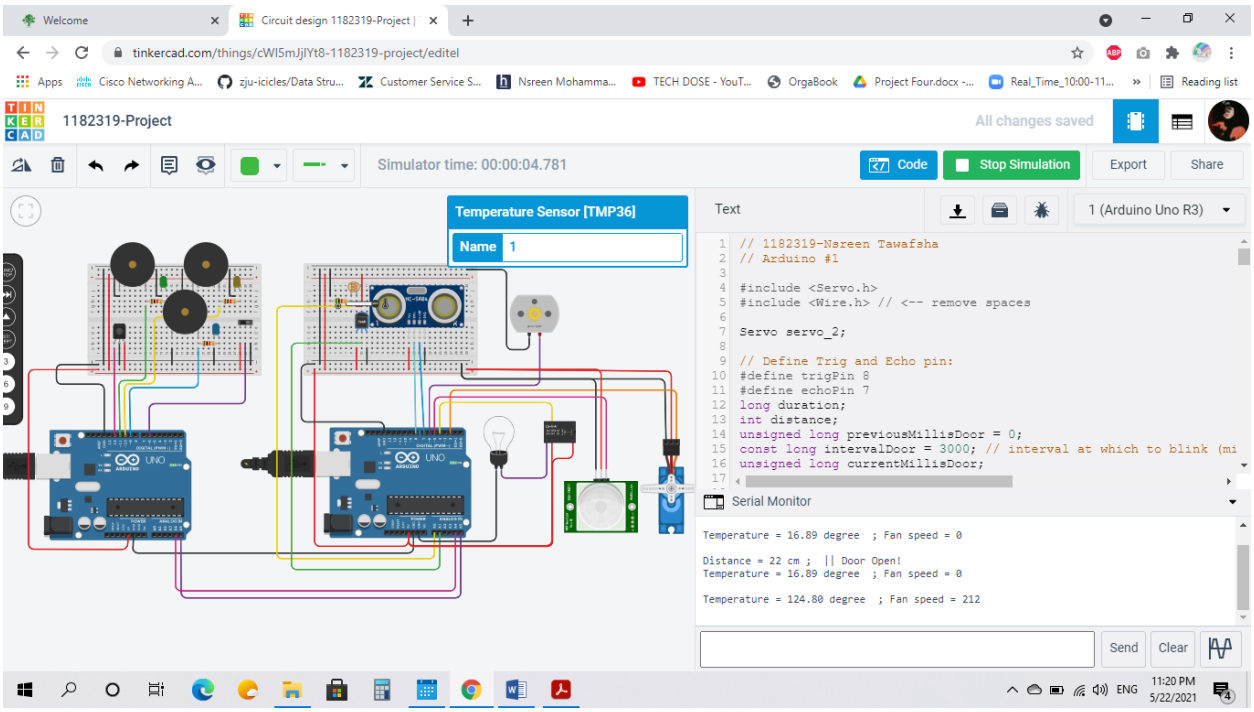
➔ The link of the tinkercad of the whole project :

https://www.tinkercad.com/things/cWl5mJjlYt8-1182319-project/editel?sharecode=dXaFD55nDqubBRgTd8wXuaiID0mERwkEix6jY3MGX3c

4. Testing

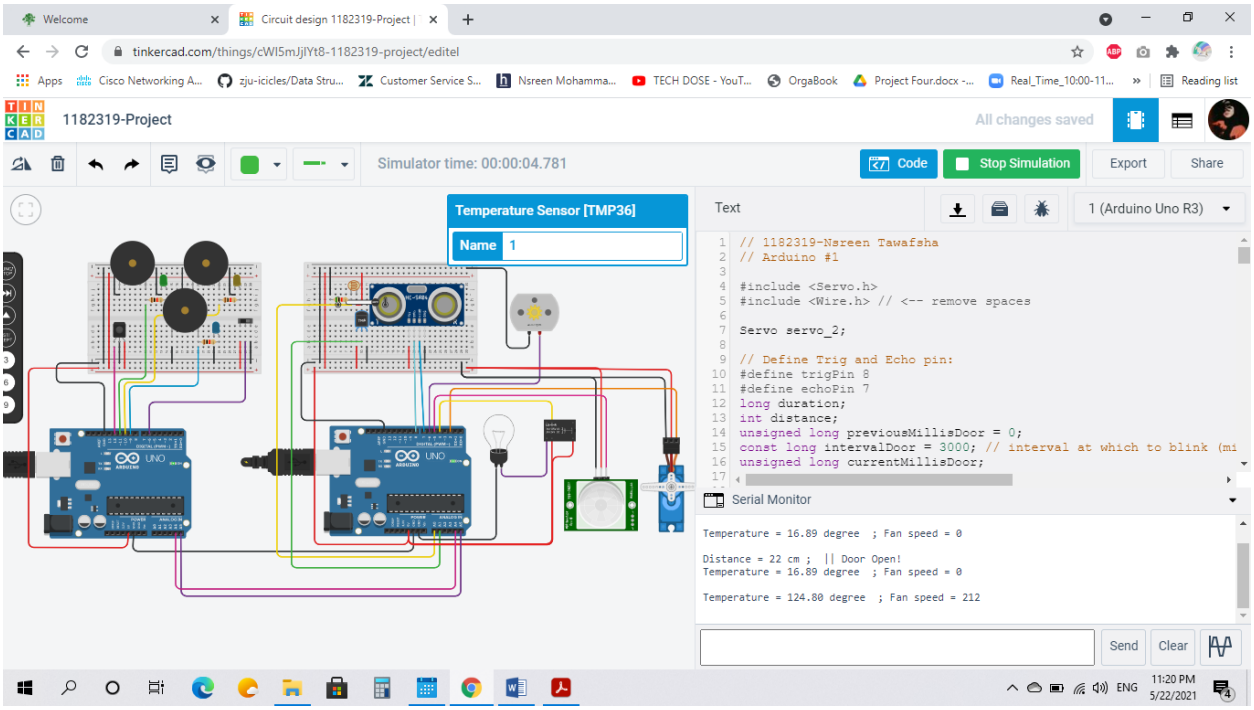
4.1 Automatic mode:

- Temperature sensor and DC motor(Fan):



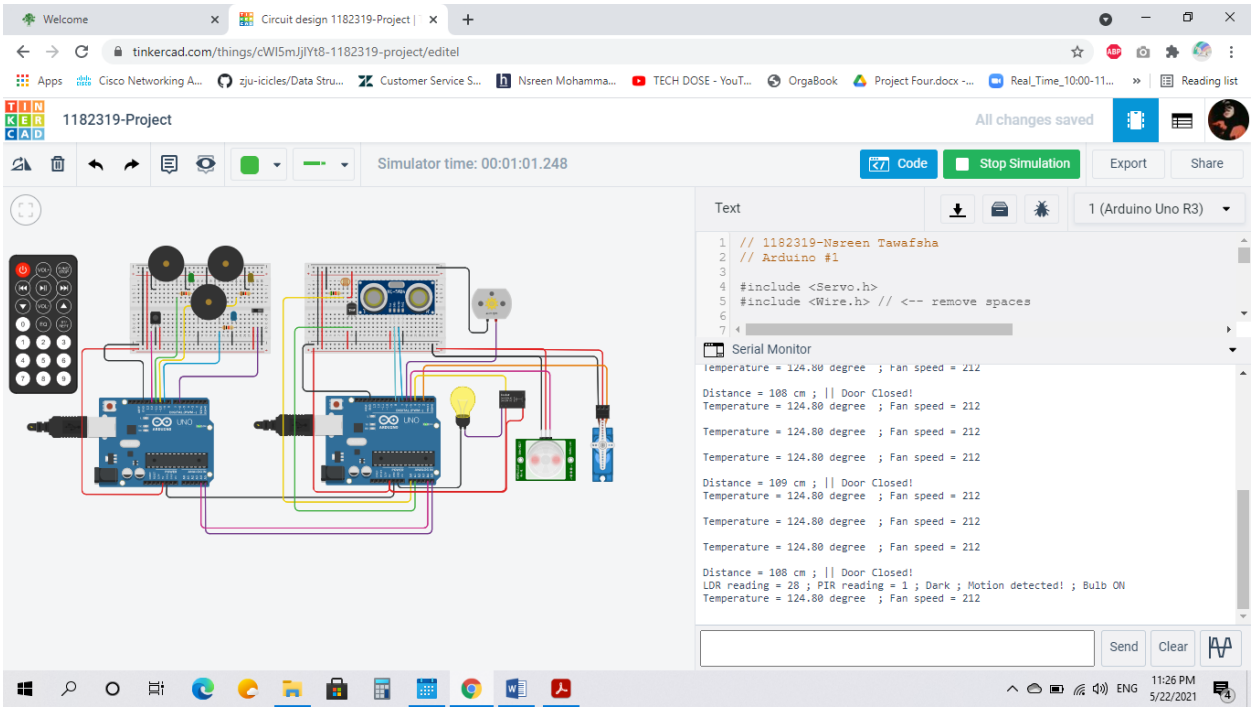
If you can see here at the above screenshot that the switch in automation mode, so if the temperature is > 30 then the fan will turned on (as seen in rmp reading on the DC motor)and its speed go faster as the temperature increases it will be (temperature * 1.7) since the maximum allow speed = 255 degree.

- Ultrasound sensor and servo motor (Door):

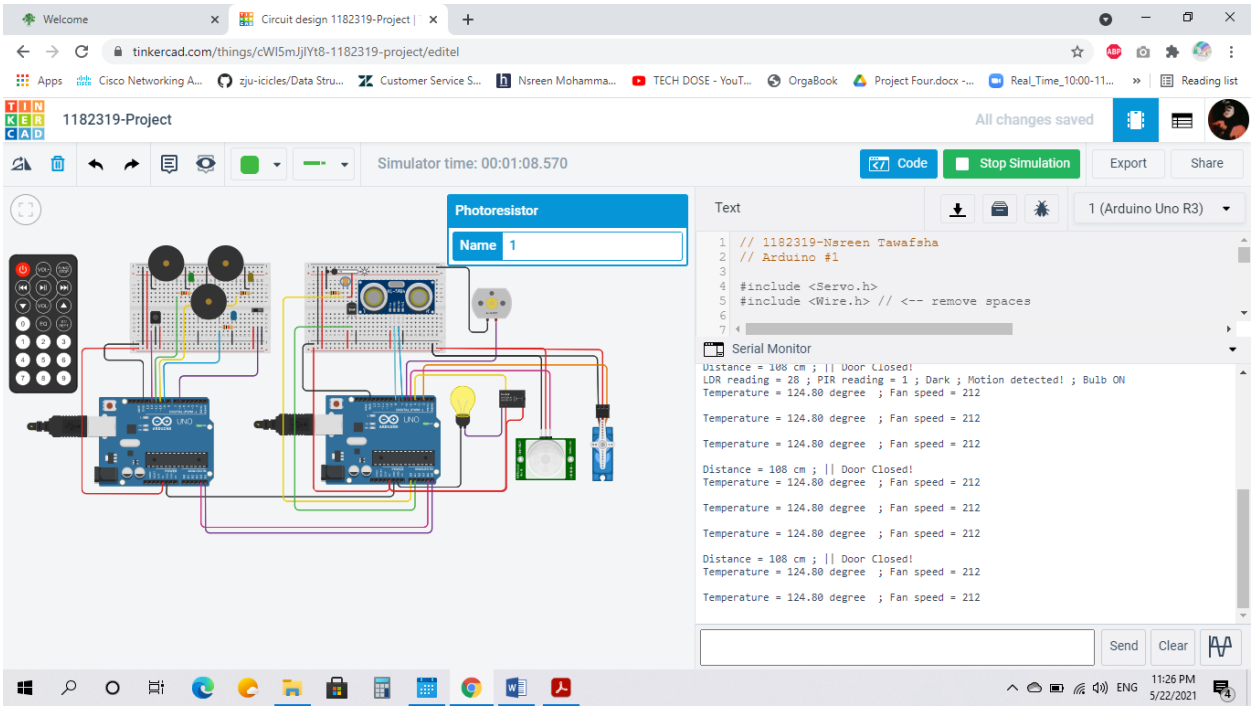


If you can see here at the above screenshot that the switch in automation mode, so if the distance is < 50 then the ultrasound sensor will know that so the door will open automatically (as seen in case of the servo motor) and the system will automatically check if there is someone in distance < 50 or not, to close the door or open it as needed.

- LDR, PIR and the Bulb:

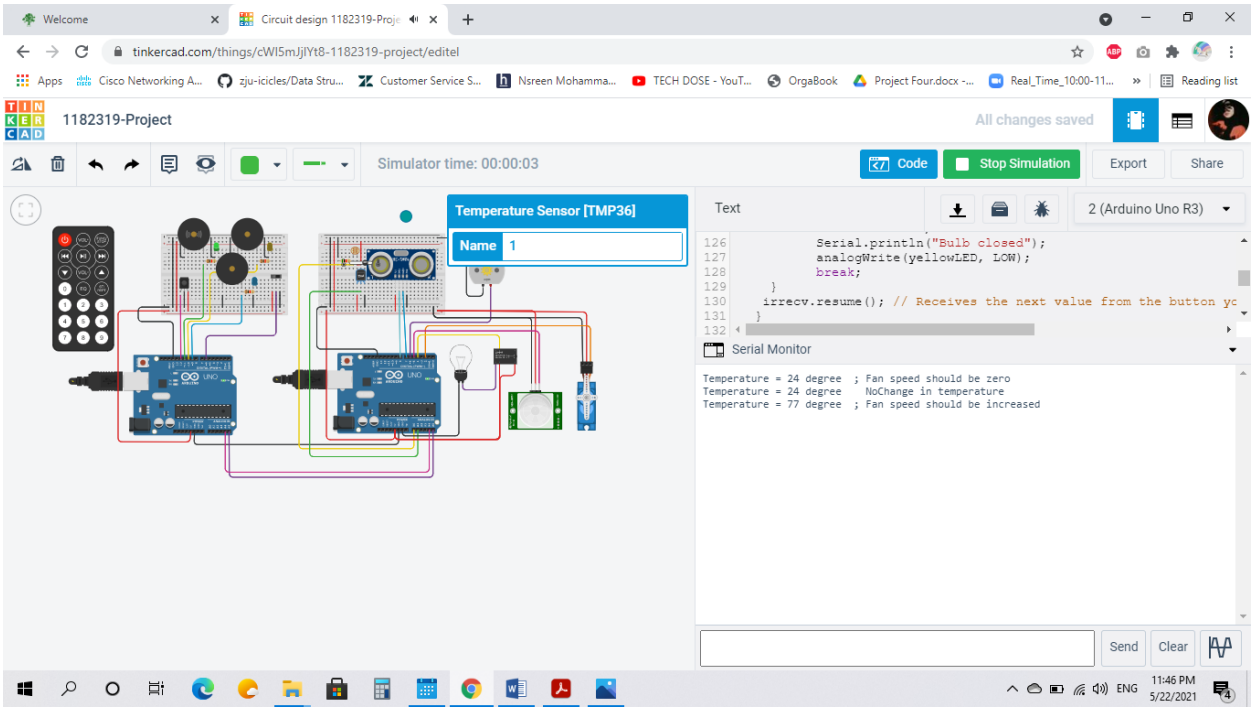


If you can see here at the above screenshot that the switch in automation mode, so if there is no enough light (the reading of LDR = 0 that's mean it's smaller than 500) and there is a motion (the reading of the PIR = 1) so the bulb will turn on for 1 Minuit as seen in below picture aslo after 1 minutes and 8 seconds.



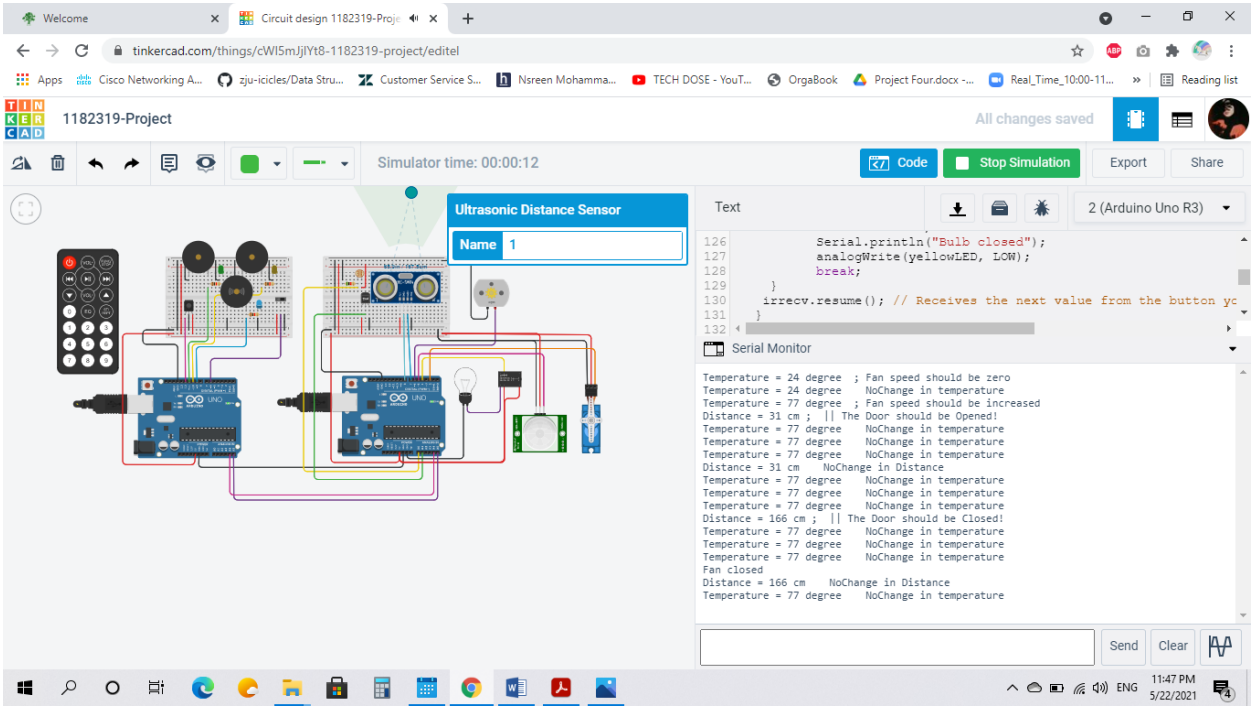
4.1 Automatic mode:

- Temperature sensor and DC motor(Fan) and there alarm (green led):



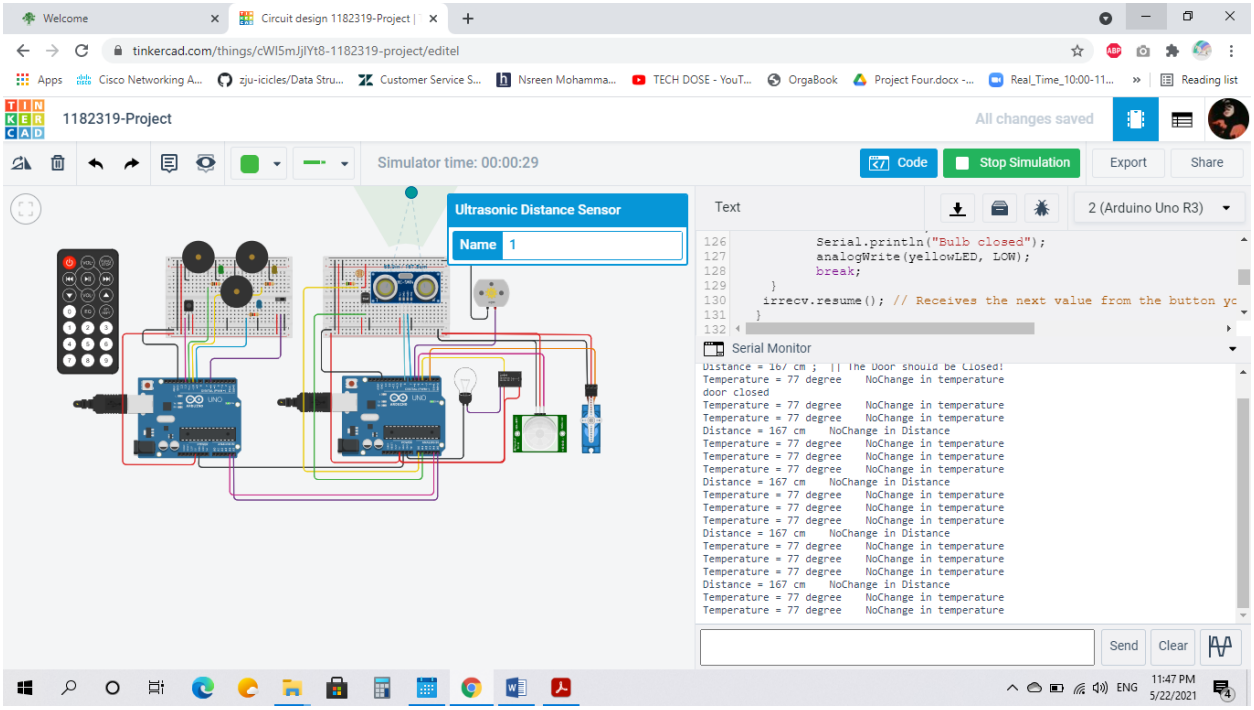
If you can see here at the above screenshot that the switch in manual mode, so if the temperature is > 30 then the green led and its buzzer will turn on as seen above, so if pressing in 2 then the led and the buzzer will turn off and the fan will be turned on in speed = 1.7 * temperature value that is if the temperature is > 30, but if not, the fan then will turned off.

- Ultrasound sensor and servo motor (Door):



If you can see here at the above screenshot that the switch in manual mode, so if the distance is < 50 then the blue led and its buzzer will turn on as seen above, so if pressing in 4 then the led and the buzzer will turn off and the door will open or close as needed.

- No change in any case:



If there is no change in any case, so no change on the state of the components then no led and its buzzer will turned on.

5. Conclusion

In this experiment, we learned how to build a complete smart home automation, and control many components such as the DC motor, servo motor and bulb using the Arduino board, also we learned how to communicate between two Arduino boards in intelligent way.

6. References

[1] https://components101.com/sensors/ultrasonic-sensor-working-pinout-datasheet

[2] https://www.electrical4u.com/what-is-servo-motor/

[3] https://www.arduino.cc/reference/en/libraries/servo/

[4] https://www.electronics-notes.com/articles/electronic_components/resistors/light-dependent-resistor-ldr.php

[5] https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/overview

[6] https://en.wikipedia.org/wiki/Incandescent_light_bulb

[7] https://www.fierceelectronics.com/sensors/what-a-temperature-sensor

[8] https://www.electronics-tutorials.ws/io/io_7.html

[9] https://www.javatpoint.com/arduino-switch

[10] https://www.circuitbasics.com/arduino-ir-remote-receiver-tutorial/

[11] https://byjus.com/physics/light-emitting-diode/#Uses

[12] https://ozeki.hu/p_2977-how-to-use-a-buzzer-in-arduino.html

7. Appendix

7.1 Appendix A:

```
// 1182319-Nsreen Tawafsha
// Arduino #1

#include <Servo.h>
#include <Wire.h> // <-- remove spaces

Servo servo_2;

// Define Trig and Echo pin:
#define trigPin 8
#define echoPin 7
long duration;
int distance;
unsigned long previousMillisDoor = 0;
const long intervalDoor = 3000; // interval at which to blink (milliseconds)
unsigned long currentMillisDoor;

#define LDR 0 // the cell and 10K pulldown are connected to a0
#define PIR 5
#define bulb 4
int LDR_Reading; // the analog reading from the sensor divider
int PIR_Reading; // variable for reading the pin status of PIR
unsigned long previousMillisBulb = 0;
const long intervalBulb = 60000; // interval at which to blink (milliseconds)
unsigned long currentMillisBulb;

#define FAN 6
#define TEMP 1
int tempMin = 30;
int fanSpeed;
double temperature; // variable for storing the analog readong from the Temperature sensor
int tempSend = 0;

int automaticMode = 1; //variable to make it possible to read the state, if we gonna be in the automatic mode or not
int fanStateHere = 0;      //Blue led will turn on to alarm about the situation of the servo motor(Door)
int doorStateHere = 0;         //Yellow led will turn on to alarm about the situation of the Bulb
int bulbStateHere = 0;            //Green led will turn on to alarm about the situation of the DC motor(FAN)

int fanState = 0;
int doorState = 0;
int bulbState = 0;
//************************************************************
void setup()
{
   // Define inputs and outputs:
   pinMode(trigPin, OUTPUT);
   pinMode(echoPin, INPUT);
   servo_2.attach(2, 500, 2500); //servo motor
```

```
  pinMode(LDR, INPUT);
  pinMode(PIR, INPUT);
  pinMode(bulb, OUTPUT);

  pinMode(TEMP, INPUT);
  pinMode(FAN, OUTPUT);

  servo_2.write(0);
  digitalWrite(bulb, LOW);

  /// For sending
  Wire.begin(8);                // join i2c bus with address #2
  Wire.onRequest(requestEvent); // register event

  //Begin Serial communication at a baudrate of 9600:
  Serial.begin(9600);
}
//************************************************************
void requestEvent()
{
  Wire.write(doorStateHere);
  Wire.write(bulbStateHere);
  Wire.write(fanStateHere);
  Wire.write(distance);
  Wire.write(tempSend);
  Wire.write(LDR_Reading);
  Wire.write(PIR_Reading);
}
//************************************************************
void loop()
{
  //---------------------------------------------------------------------
  ////---------- readUltrasonicDistance ----------////
  //---------------------------------------------------------------------

  currentMillisDoor = millis();
  if (currentMillisDoor - previousMillisDoor >= intervalDoor)
  {

    // save the last of the blink
    previousMillisDoor = currentMillisDoor;

    // Clear the trigPin by setting it LOW:
    digitalWrite(trigPin, LOW);
    // Trigger the sensor by setting the trigPin high for 3 seconds:
    digitalWrite(trigPin, HIGH);
    digitalWrite(trigPin, LOW);
    // Read the echoPin, pulseIn() returns the duration (length of the pulse) in microseconds:
    duration = pulseIn(echoPin, HIGH);
    // Calculate the distance:
    distance = duration * 0.01723;
    Serial.print("Distance = ");
```

```
    Serial.print(distance);
    Serial.print(" cm ");

    if (automaticMode == 0)
    {   /// Auto mode
       // Print the distance on the Serial Monitor (Ctrl+Shift+M):
       if (distance <= 50)
       {
          servo_2.write(90);
          Serial.println("  || Door Open!");
          doorStateHere = 90 ;
       }
       else
       {
          servo_2.write(0);
          Serial.println("; || Door Closed!");
          doorStateHere = 0 ;
       }
    }
    else
    { ///// send alarm to arduino #2 about the door
       servo_2.write(doorState);
    }
}

//---------------------------------------------------------------------------
////---------- LDR (to detect Light) and PIR (to detect movement) ----------////
//---------------------------------------------------------------------------

currentMillisBulb = millis();
if (currentMillisBulb - previousMillisBulb >= intervalBulb)
{
   // save the last of the blink
   previousMillisBulb = currentMillisBulb;

   LDR_Reading = analogRead(LDR);
   Serial.print("LDR reading = ");
   Serial.print(LDR_Reading); // the raw analog reading

   PIR_Reading = digitalRead(PIR);
   Serial.print(" ; PIR reading = ");
   Serial.print(PIR_Reading); // the raw digital reading

   if (automaticMode == 0)
   { /// Auto mode
      if (LDR_Reading < 500 && PIR_Reading == HIGH)
      {
         digitalWrite(bulb, HIGH);
         Serial.print(" ; Dark ; Motion detected!");
         Serial.println(" ; Bulb ON");
         bulbStateHere = HIGH ;
      }
      else
```

```
        {
          digitalWrite(bulb, LOW);
          Serial.print(" ; No need for Light");
          Serial.println(" ; Bulb OFF");
          bulbStateHere = LOW ;
        }
    }
    else
    { //sending alarm
      digitalWrite(bulb, bulbState);
    }
}

//---------------------------------------------------------------------------
////---------- Temperature Sensor and fan (DC Motor) ----------////
//---------------------------------------------------------------------------

temperature = analogRead(TEMP);
temperature = (temperature * 5) / 1024;
temperature = (temperature - 0.5) * 100;
tempSend = temperature;
Serial.print("Temperature = ");
Serial.print(temperature);
Serial.print(" degree ");

if (automaticMode == 0)
{ /// Auto mode
   if (temperature > tempMin)
   {
      // the actual speed of fan//map(temp, tempMin, tempMax, 32, 255);
      fanSpeed = 1.7 * temperature;
      analogWrite(FAN, fanSpeed);
      Serial.print(" ; Fan speed = ");
      Serial.println(fanSpeed);
      Serial.print("\n");
      fanStateHere = fanSpeed ;
   }
   else
   {
      fanSpeed = 0;
      analogWrite(FAN, fanSpeed);
      Serial.print(" ; Fan speed = ");
      Serial.println(fanSpeed);
      Serial.print("\n");
      fanStateHere = fanSpeed ;
   }
}
else
{ //sending alarm
   analogWrite(FAN, fanState);
}
// receive the values
Wire.requestFrom(9, 4);
```

```
  automaticMode = Wire.read(); // read one character from the I2C
  fanState = Wire.read();
  doorState = Wire.read();
  bulbState = Wire.read();

  delay(1000); // Delay a little bit to improve simulation performance
}
```

7.2 Appendix B:

```
// 1182319-Nsreen Tawafsha
// Arduino #2

#include <Wire.h> // <-- remove spaces
#include <IRremote.h>

int IR_Recv = 12;   //IR Receiver Pin 12
IRrecv irrecv(IR_Recv);
decode_results results;

#define blueLED 9
#define yellowLED 10
#define greenLED 11

int arduino_1_Address = 9;
int switchValue = 1;
int doorOldState = 0;
int bulbOldState = 0;
int fanOldState = 0;

unsigned long previousMillisDoor = 0;
const long intervalDoor = 3000;  // interval at which to blink (milliseconds)
unsigned long currentMillisDoor;

unsigned long previousMillisBulb = 0;
const long intervalBulb = 60000;  // interval at which to blink (milliseconds)
unsigned long currentMillisBulb;

int fanState = 0;
int doorState = 0;
int bulbState = 0;

int arduino_2Pin = 6;
int tempMin = 30 ;
int distance ;
int temperature ;
int LDR_Reading ;
int PIR_Reading;

int distanceOld = 0;
int temperatureOld = 0;
int LDR_ReadingOld = 0;
int PIR_ReadingOld = 0;

//************************************************************
void setup()
{
  pinMode(arduino_2Pin, INPUT);
  pinMode(blueLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
```

```
  Wire.begin(9); // join i2c bus
  Wire.onRequest(requestEvent); // register event

  irrecv.enableIRIn(); // Starts the receiver

  //Begin Serial communication at a baudrate of 9600:
  Serial.begin(9600);
}
//*****************************************************************
void requestEvent() {
  Wire.write(switchValue);
  Wire.write(fanState);
  Wire.write(doorState);
  Wire.write(bulbState);
}
//*****************************************************************
void loop()
{
  //-------------------------------------------------------------------------
    ////---------- Sending and Receving values ----------////
  //-------------------------------------------------------------------------

  // read the switch value and send it:
  switchValue = digitalRead(arduino_2Pin);
  Wire.beginTransmission( arduino_1_Address ); // transmit to device #9
  Wire.write(switchValue); // sends switch value
  Wire.endTransmission(); // stop transmitting

  // receive the values
  Wire.requestFrom(8, 7);
  doorOldState = Wire.read();// read one character from the I2C
  bulbOldState = Wire.read();
  fanOldState = Wire.read();
  distance = Wire.read();
  temperature = Wire.read();
  LDR_Reading = Wire.read();
  PIR_Reading = Wire.read();

  //********** Arduino #2 **********//
  // check the state of the switch. If it is HIGH, we are in Manual Mode:
  if (switchValue == HIGH ){ // Go to Arduino #2
    //decodes the infrared input
    if (irrecv.decode(&results)){
      long int decCode = results.value;
      //switch case to use the selected remote control button
      switch (results.value){
      case 16582903: //when you press the 1 button ( turn on the Fan )
        fanState = temperature * 1.7;
        Serial.println("Fan opened");
        analogWrite(greenLED, LOW);
        break;
      case 16615543: //when you press the 2 button ( turn off the fan)
```

```
      fanState = 0;
      Serial.println("Fan closed");
      analogWrite(greenLED, LOW);
      break;
    case 16599223: //when you press the 3 button (door open)
       doorState = 90 ;
       Serial.println("door opened");
       analogWrite(blueLED, LOW);
       break;
    case 16591063: //when you press the 4 button
       doorState = 0 ;
       Serial.println("door closed");
       analogWrite(blueLED, LOW);
       break;
    case 16623703: //when you press the 5 button (tuen the bulb on)
       bulbState = 1 ;
       Serial.println("Bulb opened");
       analogWrite(yellowLED, LOW);
       break;
    case 16607383: //when you press the 6 button
       bulbState = 0 ;
       Serial.println("Bulb closed");
       analogWrite(yellowLED, LOW);
       break;
  }
 irrecv.resume(); // Receives the next value from the button you press
}

 //-------------------------------------------------------------------------
  ////---------- Arduino #2 Door ----------////
 //-------------------------------------------------------------------------
 currentMillisDoor = millis();
 if (currentMillisDoor - previousMillisDoor >= intervalDoor) {
  // save the last of the blink
  previousMillisDoor = currentMillisDoor;

  Serial.print("Distance = ");
  Serial.print(distance);
  Serial.print(" cm ");

  if(doorState != doorOldState || distance != distanceOld){
   //analogWrite(blueLED , 350);
   digitalWrite(blueLED, HIGH);
   if(distance <= 50){
    Serial.println(";  || The Door should be Opened!");
   }
   else{
    Serial.println(";  || The Door should be Closed!");
   }
   doorOldState = doorState ;
   distanceOld = distance ;
  }
  else{
```

```
    //analogWrite(blueLED , LOW);
    Serial.println("   NoChange in Distance");
   }

  }
  //-------------------------------------------------------------------------
   ////---------- Arduino #2 Bulb ----------////
  //-------------------------------------------------------------------------
  currentMillisBulb = millis();
  if (currentMillisBulb - previousMillisBulb >= intervalBulb) {
   // save the last of the blink
   previousMillisBulb = currentMillisBulb ;

   Serial.print("LDR reading = ");
   Serial.print(LDR_Reading);
   Serial.print(" ; PIR reading = ");
   Serial.print(PIR_Reading);

   if(bulbState != bulbOldState || ((LDR_Reading != LDR_ReadingOld)||(PIR_Reading != PIR_ReadingOld) )){
      //analogWrite(yellowLED, 500);
    digitalWrite(yellowLED, HIGH);
     if (LDR_Reading < 500 && PIR_Reading == HIGH) {
      Serial.print(" ; Dark ; Motion detected!");
      Serial.println(" ; Bulb should be turned ON");
     }
     else          {
      Serial.print(" ; No need for Light");
      Serial.println(" ; Bulb should be turned OFF");
     }
     bulbOldState = bulbState ;
     LDR_ReadingOld = LDR_Reading ;
     PIR_ReadingOld = PIR_Reading ;
    }
    else{
     //analogWrite(yellowLED, LOW);
     Serial.println("   NoChange in Motion and light");
    }


  }
  //-------------------------------------------------------------------------
   ////---------- Arduino #2 FAN ----------////
  //-------------------------------------------------------------------------
  Serial.print("Temperature = ");
  Serial.print(temperature);
  Serial.print(" degree ");

  if(fanState!= fanOldState || temperature != temperatureOld){
   //analogWrite(greenLED, 200);
   digitalWrite(greenLED, HIGH);
   if(temperature > tempMin){
    Serial.print(" ; Fan speed should be increased\n");
   }
```

```
    else{
      Serial.print(" ; Fan speed should be zero\n");
    }
    fanOldState = fanState ;
    temperatureOld = temperature ;
  }
  else{
    //digitalWrite(greenLED, 0);
    Serial.println("   NoChange in temperature");
  }

 }
 //--------------------------------------------------------------------------
  ////---------- Arduino #1 ----------////
 //--------------------------------------------------------------------------
 else{  // Go to Arduino #1
  digitalWrite(blueLED, LOW);
  digitalWrite(yellowLED, LOW);
  digitalWrite(greenLED, LOW);
  Serial.println("\n\n-----------------------------------------");
  Serial.println("Arduino #1 ;  Automatic Mode");
  Serial.println("-----------------------------------------\n\n");
 }
 delay(1000); // Delay a little bit to improve simulation performance
}
```