Electrical and Computer Systems Engineering Department

Computer Design Laboratory

ENCS411

Project Report
SMART HOME AUTOMATION PROJECT

Prepared by

Anas Nimer_1180180

Instructor:

Dr. Abdallatif Abuissa

Teacher assistant:

Eng. Motasem Diab

BIRZEIT

August – 2020

1. Abstract

This project talks about how to build smart house using Arduino and other chips, we learn how to deal with a number of chips such as LEDS, LDR, LCDS, servo motor, DC motor, Photocell, IR remote control and its sensor. we will show each chip distinctly in this report and how to write code that manage each of them using tinkercad application.

Table of Contents

Table of figure:

2. Components Background

2.1. Arduino Board:

Contains analog & digital pins wit ground and power used as basic unit that all chips connected with it.

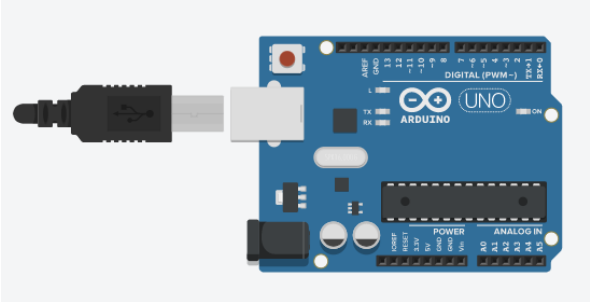

Figure 1:Arduino Board

2.2. Breadboard Small :

Used to put chips and wires on it contains 2 special rows for power and ground in each half.

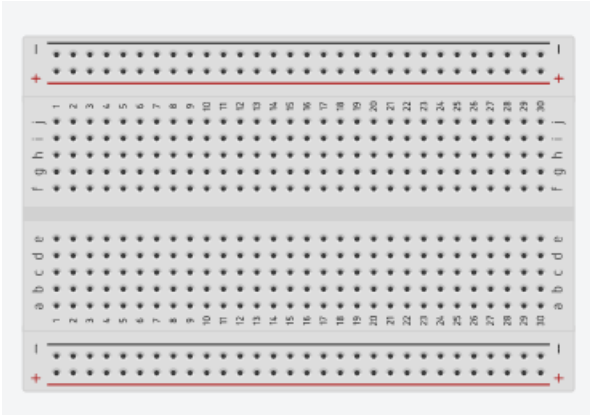

Figure 2:Breadboard Small

2.3. Servo Motor:

Used as door in this project it's a motor connect with power, ground and signal can controlled by enter degree to it.

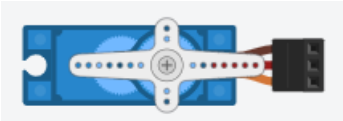

Figure 3:Servo Motor

2.4. Ultrasonic Distance Sensor:

Used to calculate distance for body close to it on range nearly equal to 310 cm its work by connect it with ground, power and signal from Arduino chip.

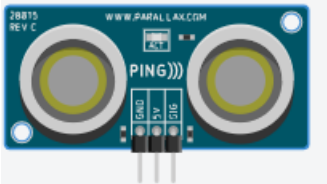

Figure 4:Ultrasonic Distance Sensor

2.5. PIR Sensor:

Used to check if there's a movement on its range or not need ground, power and signal.

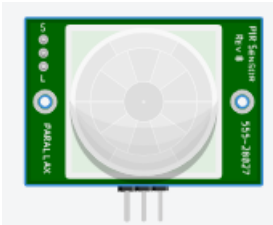

Figure 5:PIR Sensor

2.6. Light Bulb:

Include two terminals.

Figure 6:Light Bulb

2.7. Photocell (LDR):

Used to calculate Light Intensity.

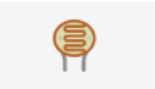

Figure 7:Photocell LDR

2.8. Resistor:

Connect with LED and photocell to protect it.

Figure 8:Resistor

2.9. Temperature Sensor:

Used to calculate temperature need to connect with power, ground and analog signal pin.

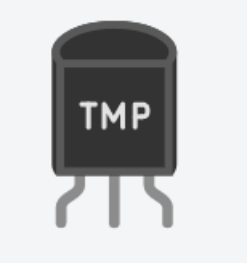

Figure 9:Temperature Sensor

2.10. DC motor:

Work as fan when temperature is greater than 30 Celsius.

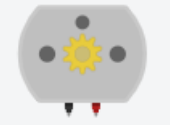

Figure 10:DC Motor

2.11. IR remote:

Used to control some chips after pressed on its buttons it will send Invisible rays to IR sensor where each button has special code value these values used to connect between the remote order and the chip as link or connect.

Figure 11:IR Remote

2.12. IR Sensor:

Used to receive remote control ray if it sends any ray.

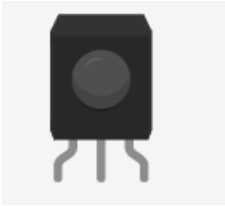

Figure 12:IR Sensor

2.13. LCD 16*2:

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display.

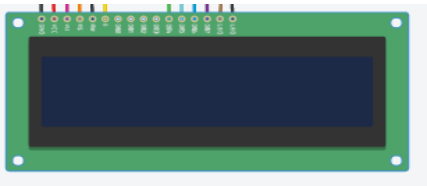

Figure 13:LCD 16*2

2.14. Potentiometer:

device that is used to measure the voltage or electric potential. It provides a variable resistance when the shaft of the device is turned.

Figure 14:Potentiometer

2.15. H-bridge Moter Driver:

is a 16-pin Motor Driver IC which can control a set of two DC motors simultaneously in any direction.

Figure 15:H-bridge Moter Driver

3. Design & Implementation:

3.1. Arduino #2 (A.1)

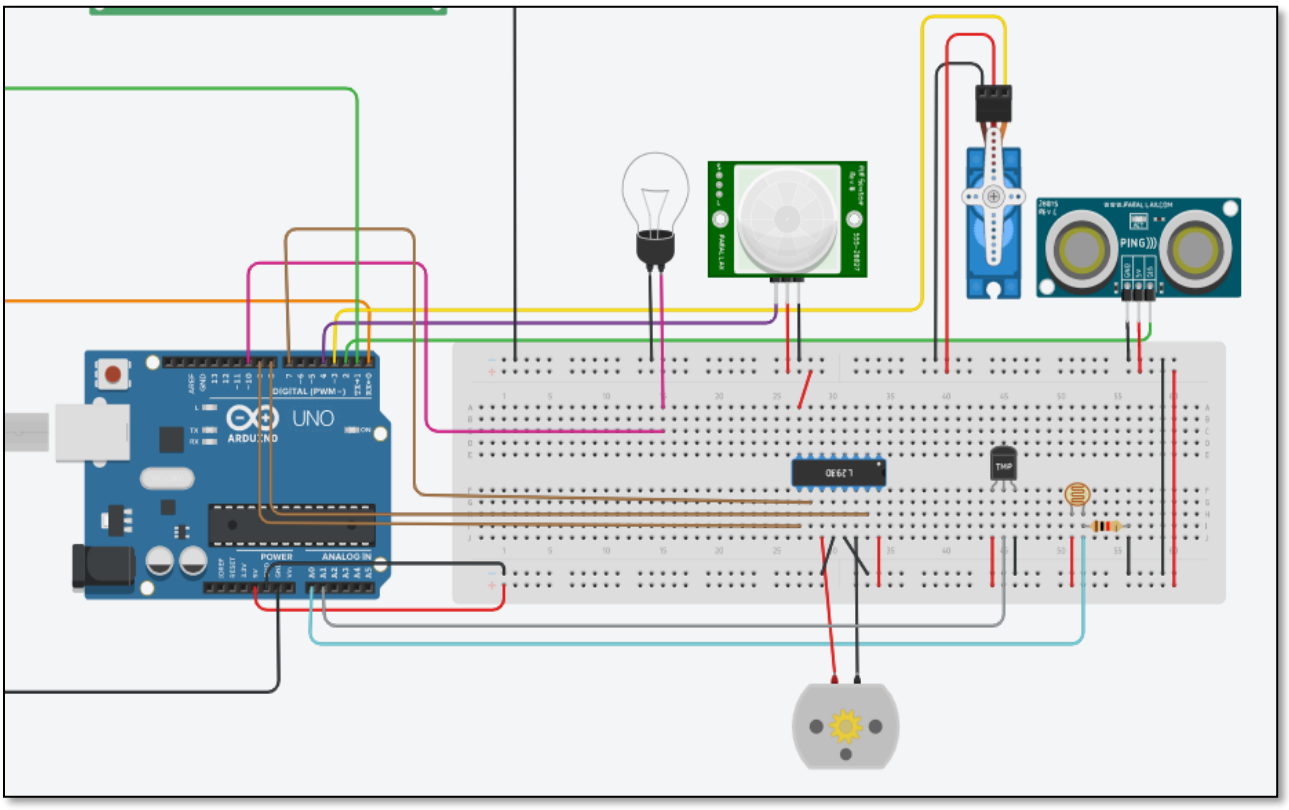

Figure 16:Arduino #1 Design

This Arduino include a lot of chips we mention them on the background, it's contained a door, light bulb and fan for the smart home we will discuss each of them with their codes below.

3.1.1. Smart Home Door

3.1.1.1. Automatic Mode:

Which is servo motor with ultrasonic distance sensor when the sensor recognizes a body in range of 1m it well send signal to servo motor which is open in 90 degree as a door.

```
// Reading Sensors
distance = readUltrasonicDistance(pingPin);
LDR = analogRead(LDRPin);
PIR = digitalRead(PIRPin);
tmp = readTmp(temperaturePin);
fanSpeed = map(tmp, -40, 125, 0, 255);

// Automatic mode
if(mode == 0){
  // Servo
  if(!doorOpen || millis() - doorDelay == doorPeriod * 1000){
    if(distance <= 100){/* if distance less or equal 1m then open
                           the door to 90 degree ,
                           else close the door to 0 degree */
      door.write(90);
      doorDelay = millis();
      doorOpen = true;
    }else{
      door.write(0);
      doorOpen = false;
    }
  }
```

Figure 17:Ultrasonic & Servo Motor Code (Automatic Mode)

In this code we have read all sensors, and ultrasonic used to calculate distance using reed UltrasonicDistance function and it's saved at distance variable. After that we check if the door closed or the difference between millis and door delay equal door period mull 1000. Then we check if the distance between the body and sensor (if it's less than or equal to 1m) then servo motor open to 90 degrees for some delay (we can save it in door delay variable).

Else (if distance greater than 1m then servo motor directly closes to 0 degree).

- **Note: (here 90 degrees used because with 180 it's difficult to recognize it in the screenshots).**

3.1.1.2. Manual Mode:

```
// Open the door if closed, and close if opened
if(button == 4 && !doorOpen){
  door.write(90);
  doorOpen = true;
  button = -1;
}else if(button == 4 && doorOpen){
  door.write(0);
  doorOpen = false;
  button = -1;
}
```

Figure 18:Ultrasonic & Servo Motor Code (Manual Mode)

In manual mode servo doesn't depend on the distance to open or close it will wait an order from the IR remote control for open and close.

In this code we have check if button #4 is pressed and door close then the door will open. Else if button #4 is pressed and door open then the door will close.

3.1.2. Smart Home Light:

3.1.2.1. (Automatic Mode):

In this part a light bulb used when there's a movement and the light is low then the light bulb will be turned on.

```
PIR = digitalRead(PIRPin); // PIR sensor read
```

Figure 19:Reed PIR Sensor

In figure 22 the values of PIR sensor read and stored in variables to use it later.

```
// Set the controlability of light bulb
/* 1: light depends on PIR only
*  2: light depends on LDR only
*  3: light depends on LDT and PIR together
*/
if(button > 0 && button < 4){
  lightControl = button;
}
```

Figure 20:Controllability of light bulb

In figure 23 the values of light Control read and stored in variables to use it later.

```
// Light Bulb
if(lightControl == 1){
  if(PIR){
    analogWrite(lightPin, 255);
    lightOn = true;
  }else{
    analogWrite(lightPin, 0);
    lightOn = false;
  }
}else if(lightControl == 2){
  if(LDR < 500){
    analogWrite(lightPin, 255);
    lightOn = true;
  }else{
    analogWrite(lightPin, 0);
    lightOn = false;
  }
}else if(lightControl == 3){
  if(LDR < 500 && PIR){
    analogWrite(lightPin, 255);
    lightOn = true;
  }else{
    analogWrite(lightPin, 0);
    lightOn = false;
  }
}
```

Figure 21:Light Bulb Automatic Code

In this code we have check the value of light control to know what kind of (PIR, LDR or LDR and PIR) the light will depend on. So, if light control equal 1 -(the light depends on PIR only)- we check if PIR not null then we write in light pin 255 and turn on the light. Else if light control equal 2 -(the light depends on LDR only)- we check if LDR less than 500 then we write in light pin 255 and turn on the light. Finally, if light control equal 3 -(the light depends on PIR and LDR)- we check if LDR less than 500 and PIR not null we write in light pin 255 and turn on the light.

3.1.2.2. Manual Mode:

```
// Turn the light on if off, and off if on
if(button == 5 && !lightOn){
  analogWrite(lightPin, 255);
  lightOn = true;
  button = -1;
}else if(button == 5 && lightOn){
  analogWrite(lightPin, 0);
  lightOn = false;
  button = -1;
}
```

Figure 22:Light Bulb Manual Code

Changes occurs depending on IR remote only, so in this code we have check if button #5 is pressed and light off then write in light pin 255 and the light will be turn on. Else if button #5 is pressed and light on then write in light pin 0 and the light will be turn off.

3.1.3. Smart Home Fan:

3.1.3.1. Automatic Mode:

In this part a DC motor work as fan depending on the temperature from the sensor .

```
fanSpeed = map(tmp, -40, 125, 0, 255); // calculate celsius from TMP
```

Figure 23:Calculate Temperature In Celsius

Temperature calculated in Celsius and saved in Celsius variable to use it in the code below.

```
// Fan
if(tmp > 35){
  digitalWrite(motorPin1, HIGH);
  digitalWrite(motorPin2, LOW);
  analogWrite(motorPin3, fanSpeed);
  fanOn = true;
}else{
  digitalWrite(motorPin1, HIGH);
  digitalWrite(motorPin2, HIGH);
  analogWrite(motorPin3, 0);
  fanOn = false;
}
```

Figure 24:Temperature & DC Motor Automatic Code

After calculate temperature in Celsius if it's greater than temperature limit which is 35 C then DC Motor worked as fan and its speed depending on the temperature when it's increased motor will be faster and if the temperature is less than the limit DC motor will turned off.

3.1.3.2. Manual Mode:

```
    // Turn the fan on if off, and off if on
    if(button == 6 && !fanOn){
      digitalWrite(motorPin1, HIGH);
      digitalWrite(motorPin2, LOW);
      analogWrite(motorPin3, fanSpeed);
      fanOn = true;
    }else if(button == 6 && fanOn){
      digitalWrite(motorPin1, HIGH);
      digitalWrite(motorPin2, HIGH);
      analogWrite(motorPin3, 0);
      fanOn = false;
    }
  }
```

Figure 25:Temperature & DC Motor Manual Code

The same thing as in automatic mode but DC activate depending on IR remote signal then if button #6 is pressed and the fan off then write in motor pin1 High, write in motor pin2 Low and write in motorpin3 speed of fan then turn on the fan. Else if button #6 is pressed and the fan on then write in motor pin1 High, write in motor pin2 High and write in motorpin3 zero then turn off the fan.

After that we define states for each chip by give each state binary number of 6 bit each bit has a different meaning than the other bit, in order to be sent these number to Arduino #3 to be recognized it and used it to print some statement on screen

```
    if(distance <= 100){
      state |= 0b00000001;
    }else{
      state &= 0b11111110;
    }
    if(PIR){
      state |= 0b00000010;
    }else{
      state &= 0b11111101;
    }
    if(LDR < 500){
      state &= 0b11111011;
    }else{
      state |= 0b00000100;
    }
    if(tmp < 35){
      state &= 0b11110111;
    }else{
      state |= 0b00001000;
    }
    if(!doorOpen){
      state &= 0b11101111;
    }else{
      state |= 0b00010000;
    }
```

```
  }
  if(!lightOn){
    state &= 0b11011111;
  }else{
    state |= 0b00100000;
  }
  if(!fanOn){
    state &= 0b10111111;
  }else{
    state |= 0b01000000;
  }

  if(sent == 15){
    Serial.write(state);
    sent = 0;
  }
  sent++;
}

double readTmp(int pin){
  int pinReading = analogRead(temperaturePin);
  double voltage = (pinReading * 5.0)/1024;
  double milliVolt = voltage * 1000;
  return (milliVolt-500)/10;
}

double readUltrasonicDistance(int pin){
  // set pin to output mode
  pinMode(pin, OUTPUT);

  // send a signal
  digitalWrite(pin, LOW);
  delayMicroseconds(2);
  digitalWrite(pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(pin, LOW);

  // set pin to input mode
  pinMode(pin, INPUT);

  // get duration in microseconds
  int duration = pulseIn(pin, HIGH);
  return duration / 29 / 2;
}
```

Figure 26: statues number of chips

The setting state type for each chip they are defined in a way as shown in the figure below

```
/* Setting state byte
----------------------
bit    value   meaning
0      0       Distance: > 1m
0      1       Distance: < 1m

1      0       No Movement
1      1       Movement

2      0       No Enough Light
2      1       Enough Light

3      0       Temperature: < 35
3      1       Temperature: > 35

4      0       Door Closed
4      1       Door Open

5      0       Light Off
5      1       Light On

6      0       Fan Off
6      1       Fan On
*/
```

Figure 27:setting state type

3.2. Arduino #1 (A.2):

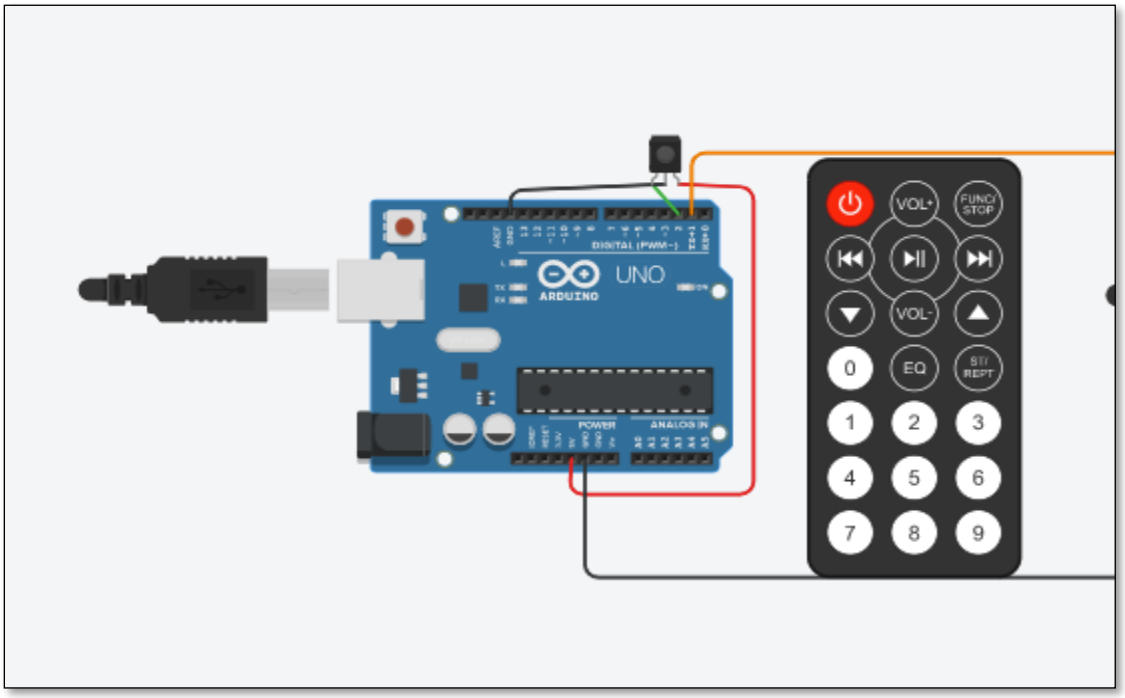

Figure 28:Arduino #1

The point of these board is to check if the user wants to work at manual or automatic mode and use IR remote control in the manual mode where in this mode if it's more accurate to do something.

3.2.1. Manual Mode:

```
/* IR Remote */
#include <IRremote.h>
IRrecv irrecv(2);
decode_results results;

/* Remote Buttons Mapping */
#define power 0xFD00FF
#define one   0xFD08F7
#define two   0xFD8877
#define three 0xFD48B7
#define four  0xFD28D7
#define five  0xFDA857
#define six   0xFD6897
void setup(){

  /* Start Remote */
  irrecv.enableIRIn();
  Serial.begin(9600);
}

void loop(){

  /* Read remote button */
  if(irrecv.decode(&results)){
    int button;
    switch(results.value){
        case power:
            button = 0;  /* map with power button to change
                            between two mode (Aoutmatic,Manual) */

        break;
        case one:
            button = 1; /* map with button 1 to make light bulb
                           depends on PIR only */
        break;
        case two:
            button = 2; /* map with button 2 to make light bulb
                           depends on LDR only */
        break;
        case three:
            button = 3; /* map with button 3 to make light bulb
                           depends on PIR and LDR */
        break;
        case four:
            button = 4; /* map with button 4 to open or close the door*/
        break;
```

```
    break;
    case five:
        button = 5; /* map with button 5 to on or off the light*/
    break;
  case six:
        button = 6; /* map with button 6 to on or off the fan(DC_motor)*/
    break;
}
```

Figure 29:Manual & IR Remote Code

In this code if user want to work in manual mode, it will give its orders using IR remote by press its buttons where each of them has a specified value that can accessed if the IR sensor recognize remote ray, the remote transmit invisible ray for each pressed button and the sensor receive them and give unique value for each of them then using these values we can run any chip by connect its condition with these pushed buttons. (The comments declare the other details)

3.2.2. Automatic Mode:

```
irrecv.resume();   /* to use Automatic mode*/
Serial.write(button);
```

Figure 30:Automatic & IR Remote Code

3.3. Arduino #3 (A.3):

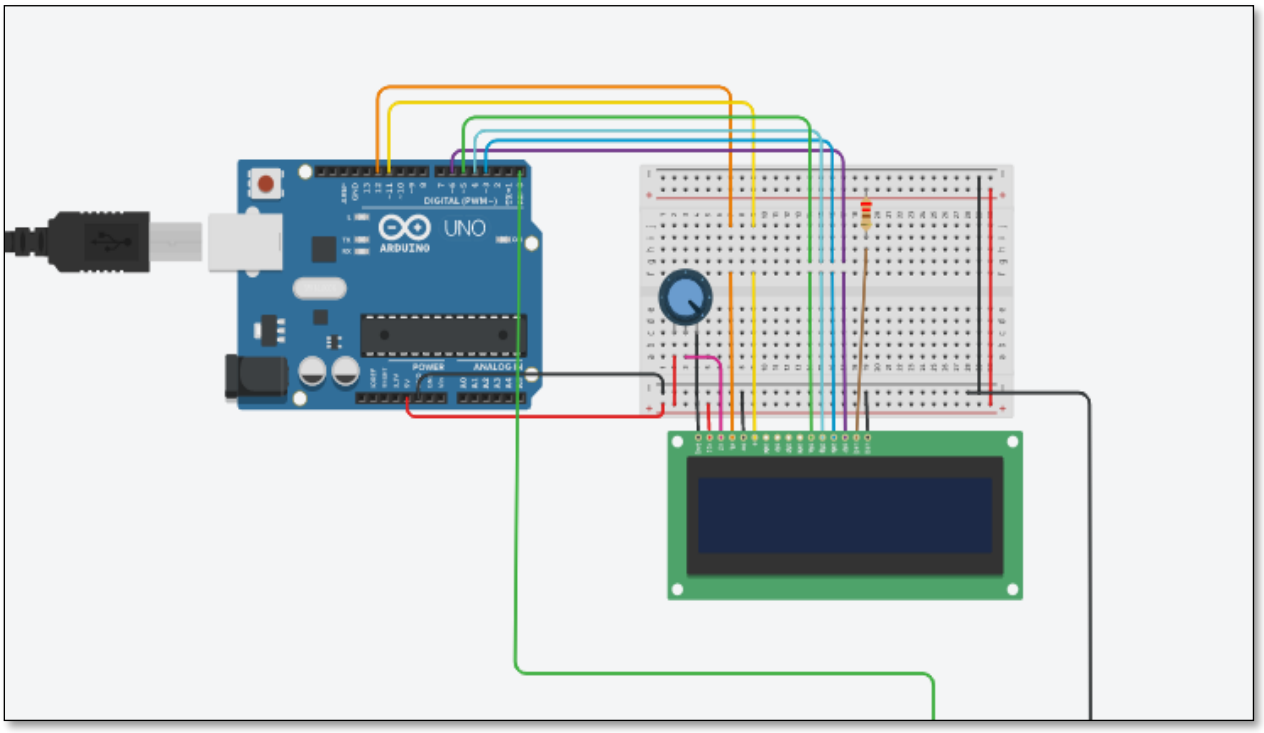

Figure 31:Arduino #3

By using the numbers that were defined in the Arduino #2, we will use them in this Arduino (Arduino #3) in order to print some identification sentences on the screen.

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 6);
String states[4][2] = {
    {"Distance: > 1m","Door Closed"},
    {"No Movement", "Light Off"},
    {"No Enough Light", "Light Off"},
    {"Temperature:< 35", "Fan Off"}
    };
int state;
int bits[7] = {0, 0, 0, 0, 0, 0, 0};
int i = 0;
int turnCount = 15;

void setup() {
  Serial.begin(9600);
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
}

void loop(){

  if(Serial.available()){
    state = Serial.read();
    Serial.println(state, BIN);
    bits[0] = getBit(state, 0);
    bits[1] = getBit(state, 1);
    bits[2] = getBit(state, 2);
    bits[3] = getBit(state, 3);
    bits[4] = getBit(state, 4);
    bits[5] = getBit(state, 5);

    if(bits[0] == 0)
      states[0][0] = "Distance: > 1m";
    else
      states[0][0] = "Distance: < 1m";

    if(bits[1] == 0)
      states[1][0] = "No Movement";
    else
      states[1][0] = "Movement";

    if(bits[2] == 0)
      states[2][0] = "No Enough Light";
```

```
    if(bits[3] == 0)
      states[3][0] = "Temperature:< 35";
    else
      states[3][0] = "Temperature:> 35";

    if(bits[4] == 0)
      states[0][1] = "Door Closed";
    else
      states[0][1] = "Door Open";

    if(bits[5] == 0){
      states[1][1] = "Light Off";
      states[2][1] = "Light Off";
    }else{
      states[1][1] = "Light On";
      states[2][1] = "Light On";
    }

    if(bits[6] == 0)
      states[3][1] = "Fan Off";
    else
      states[3][1] = "Fan On";
  }
```

Figure 32:mapping between number and chips states

In this code, we made a mapping between the number of bits that we obtained from the binary numbers that we defined in the Arduino#2 and some of the sentences that we want to print on the screen. In each case, we check the states for each chip through the binary number followed by it, and through that we determine whether the chip is working or stopped.

```
  // Printing the current state on the lcd
  lcd.setCursor(0, 0);
  lcd.print(states[i][0]);
  lcd.setCursor(0, 1);
  lcd.print(states[i][1]);

  // To alternate the shown state
  if(turnCount == 0){
    lcd.clear();
    i = (i + 1) % 4;
    turnCount = 15;
  }
  turnCount--;
}
int getBit(int b, int bitIndex){
    return (b & (1 << bitIndex)) != 0;
}
```

Figure 33:print chip state chips on screen

this code used to print the statement (status of each chip) on screen by using some delay between statements.

4. Testing

4.1. Automatic Mode

4.1.1. Smart Home Door:

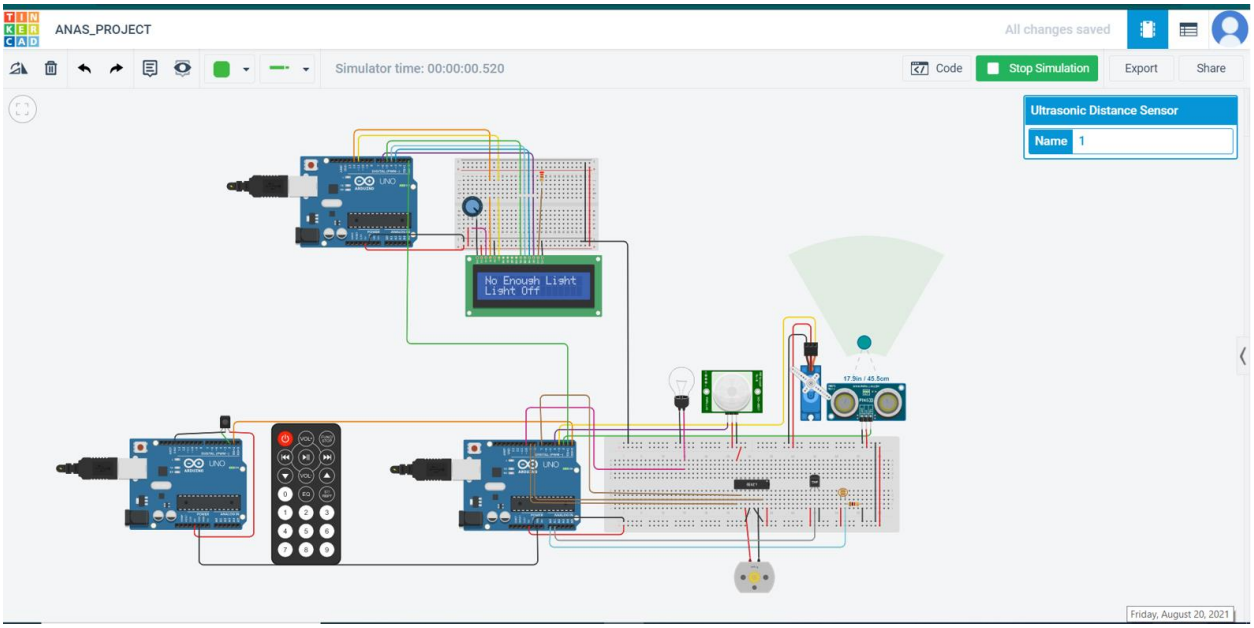

Figure 34:open the door

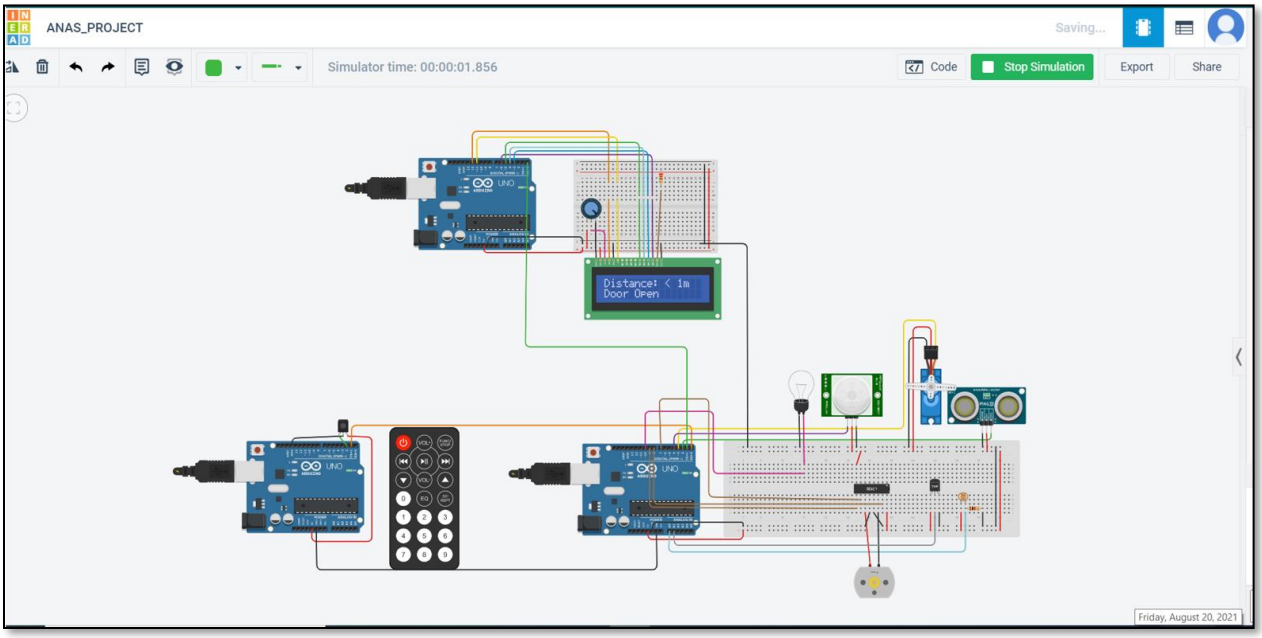

Figure 35:The Door Still Opened

- The door still opened to the next 3 seconds because there's a body in the range.

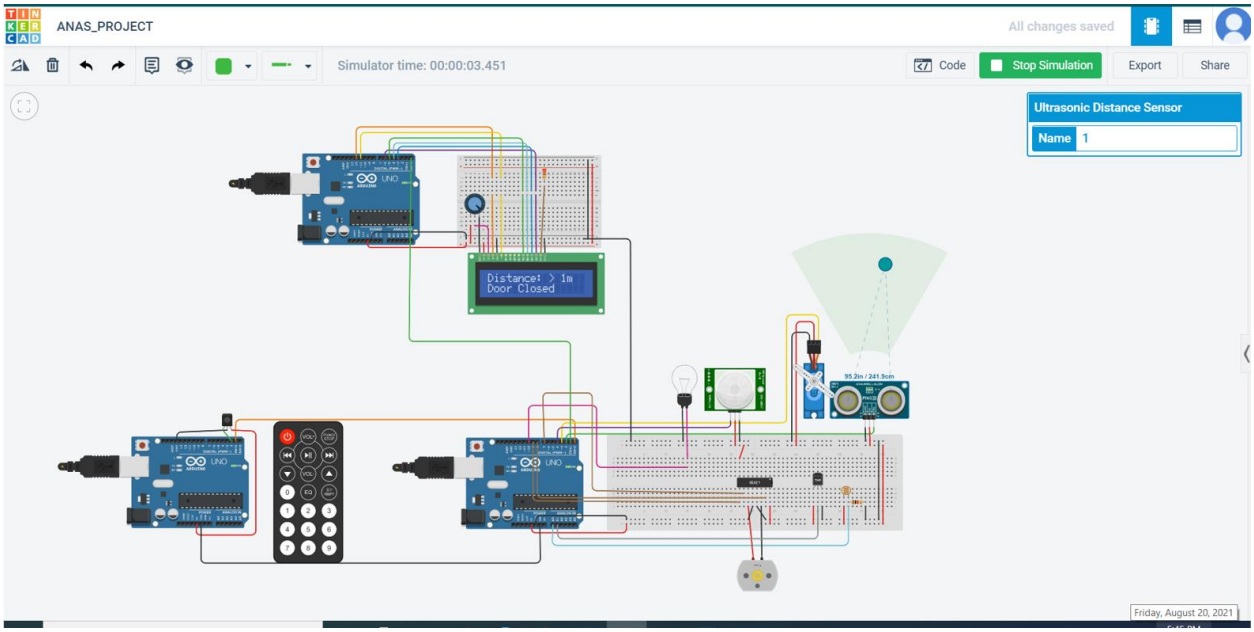

Figure 36:The Door When It's Close

4.1.2. Smart Home Light Bulb:

4.1.2.1. Light Bulb by PIR only:

- Start simulation and press on button 1 to make bulb light depends on PIR only

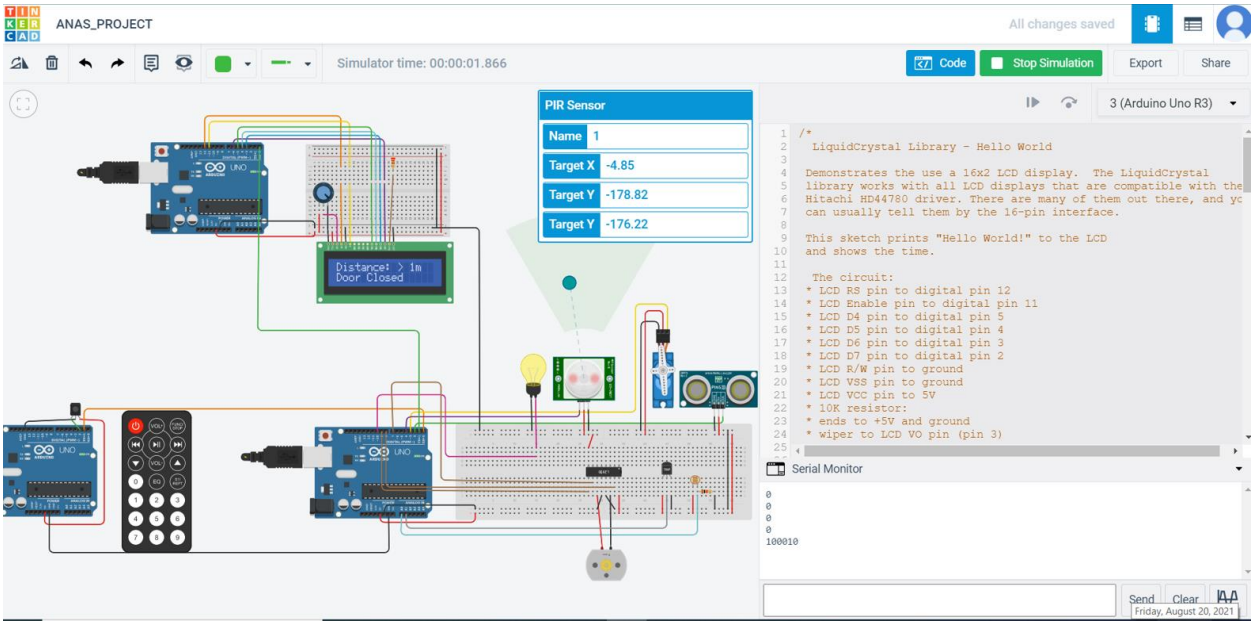

Figure 37:Turn on the Light by using PIR only(when there is movement)

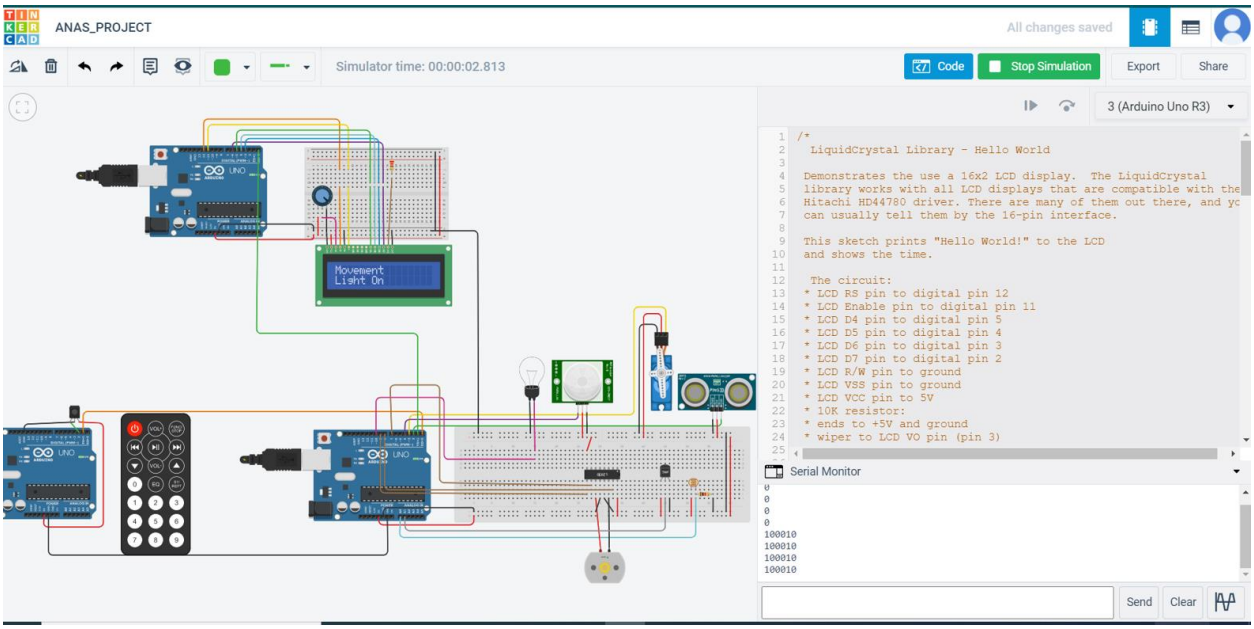

Figure 38:Turn off the Light after few second (when there is no movement)

- Light turned off after few seconds (there's small delay because of Arduino 1 & 2 delays).

4.1.2.2. Light Bulb by LDR only:

- Start simulation and press on button 2 to make bulb light depends on LDR only.

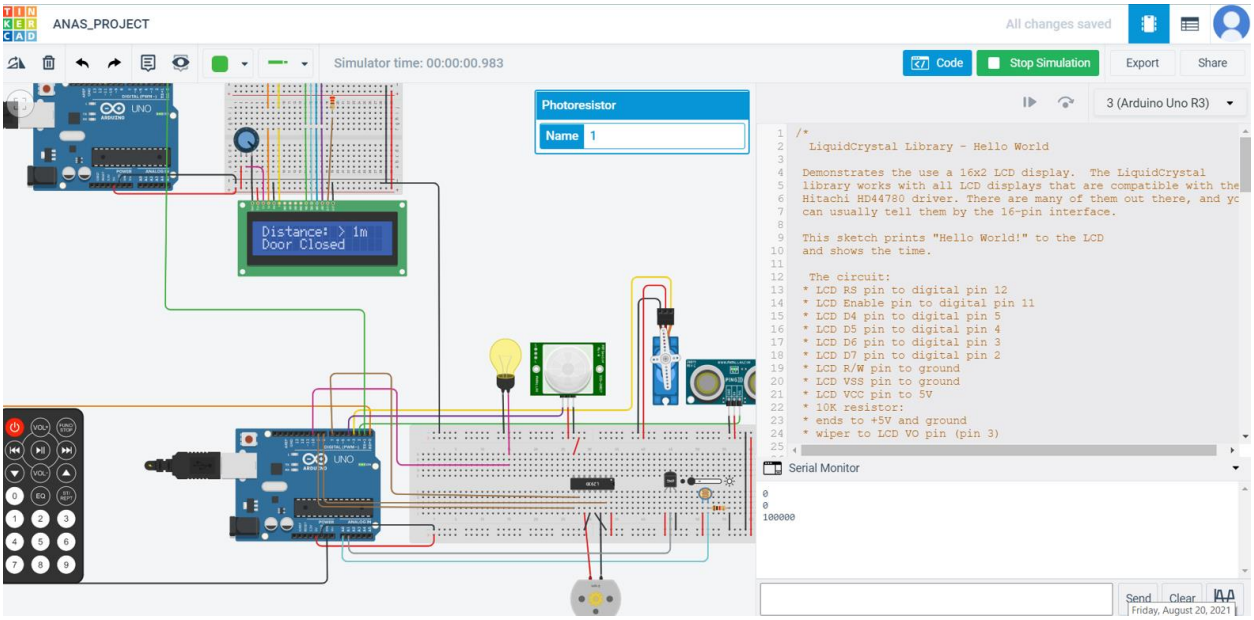

Figure 39:Turn on the Light by using LDR only (LDR<500)

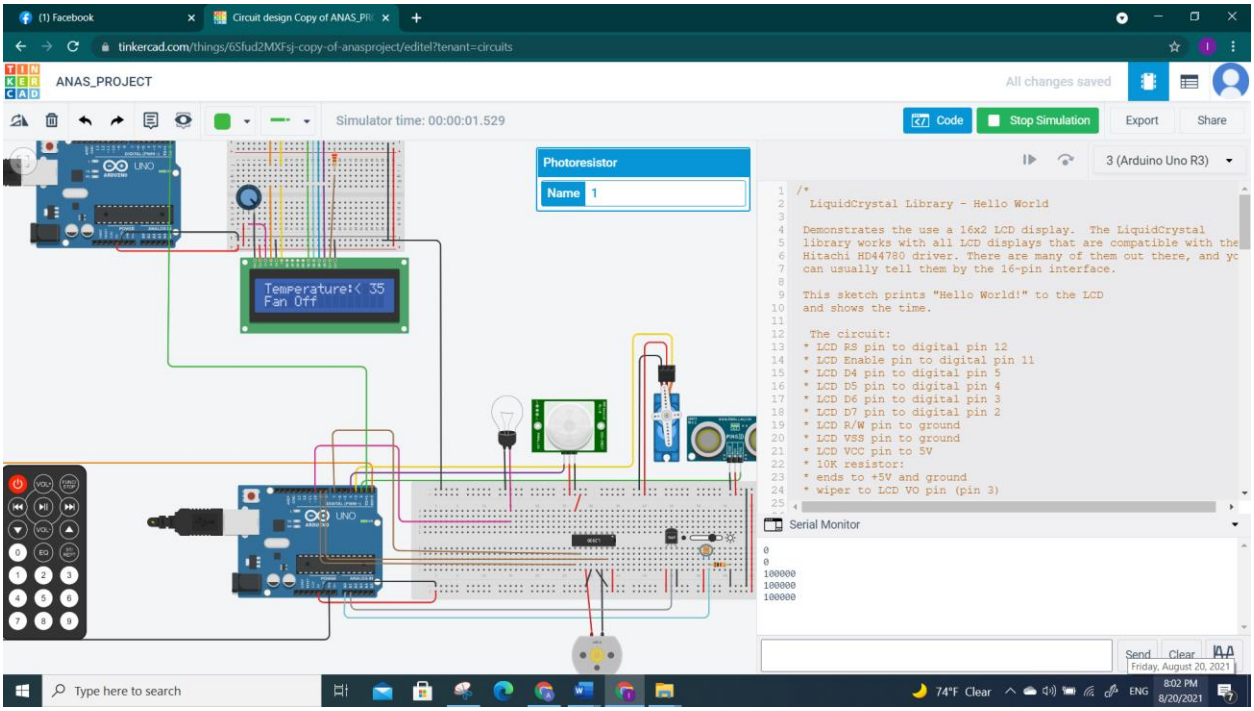

Figure 40:Turn on the Light by using LDR only (LDR>500)

4.1.2.3. Light Bulb by PIR & LDR:

- Start simulation and press on button 3 to make bulb light depends on PIR and LDR.

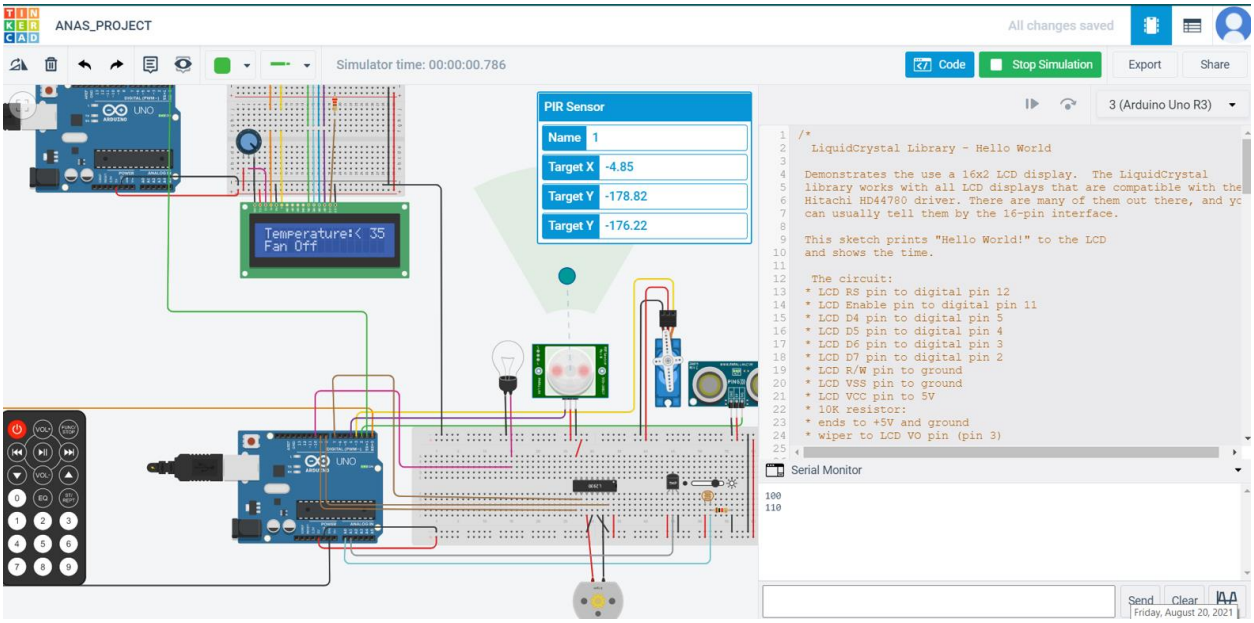

Figure 41:Turn off the Light

- The Light Bulb off because the LDR>500, Although there is movement in PIR.

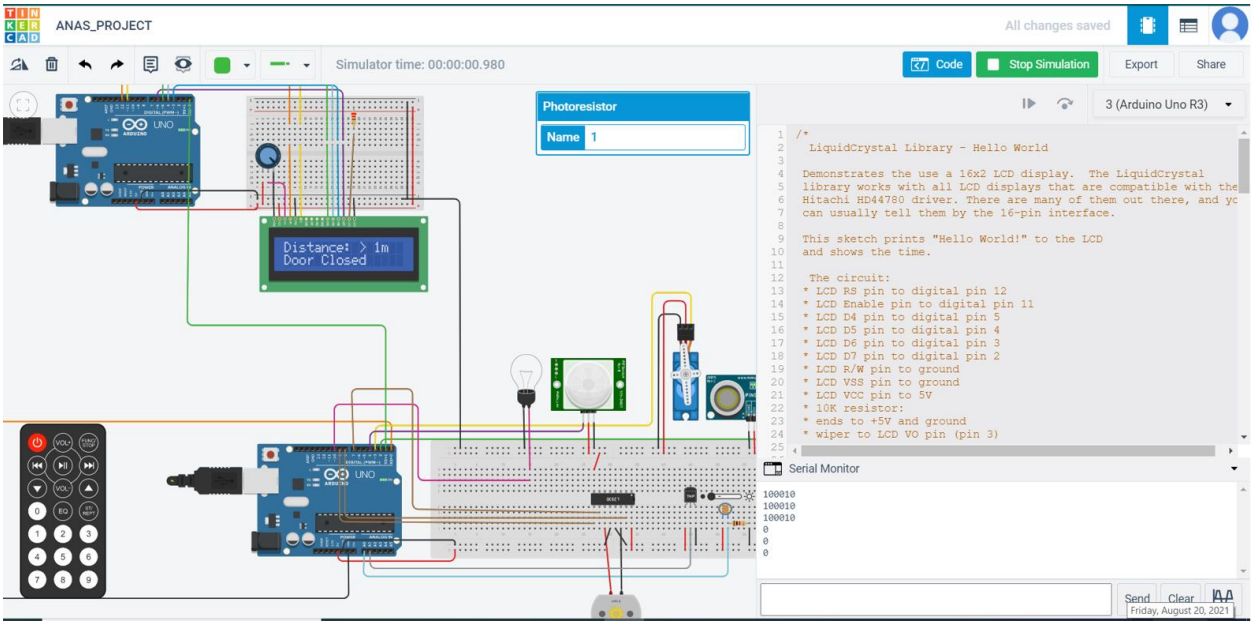

Figure 42:Turn off the Light

- The Light Bulb off because there is no movement in PIR, Although LDR<500.

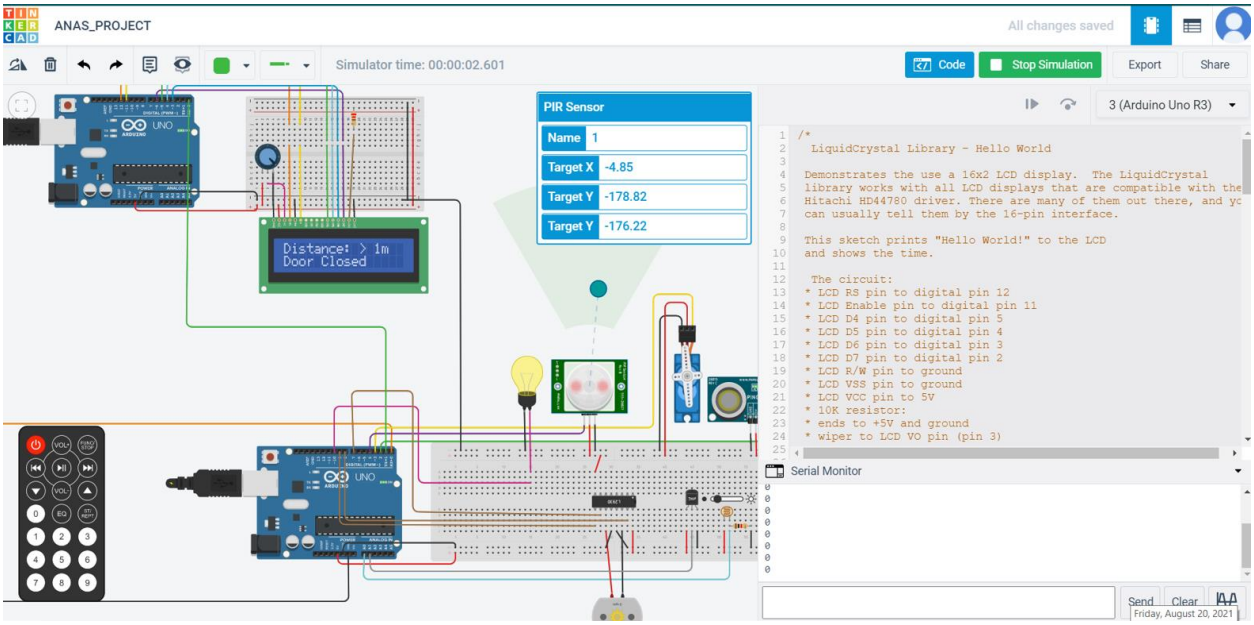

Figure 43:Turn on the Light

- The Light Bulb on because there is movement in PIR and LDR<500.

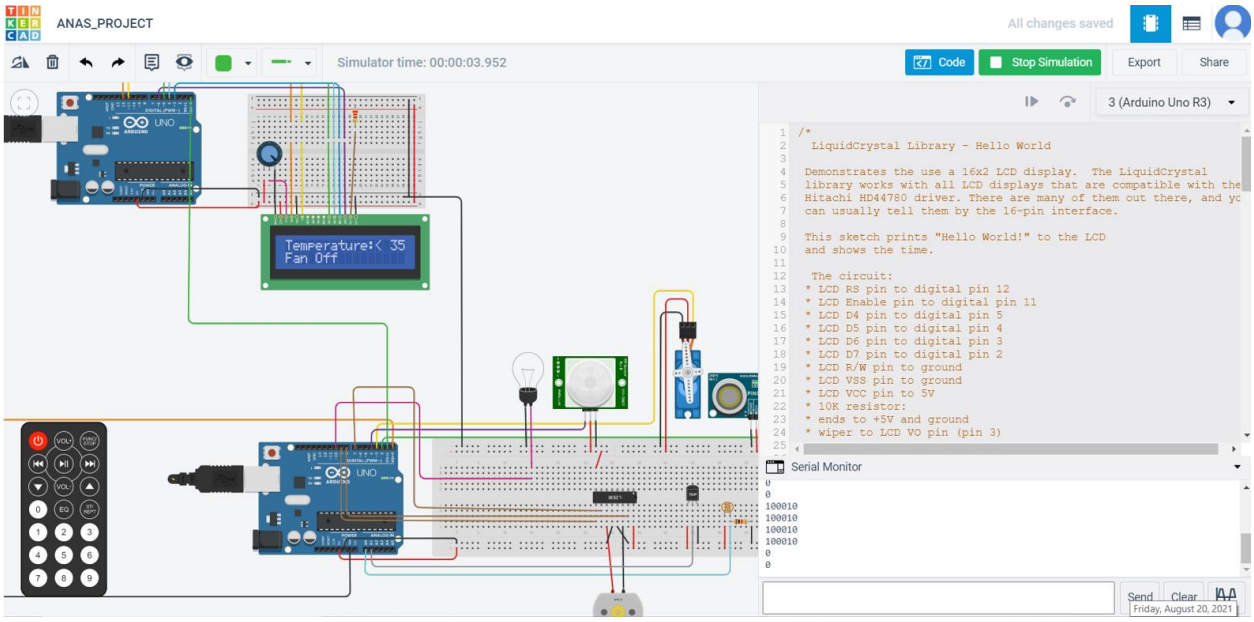

Figure 44:Turn off the Light after few second

- The Light Bulb off after few second.

4.1.3. Smart Home Fan:

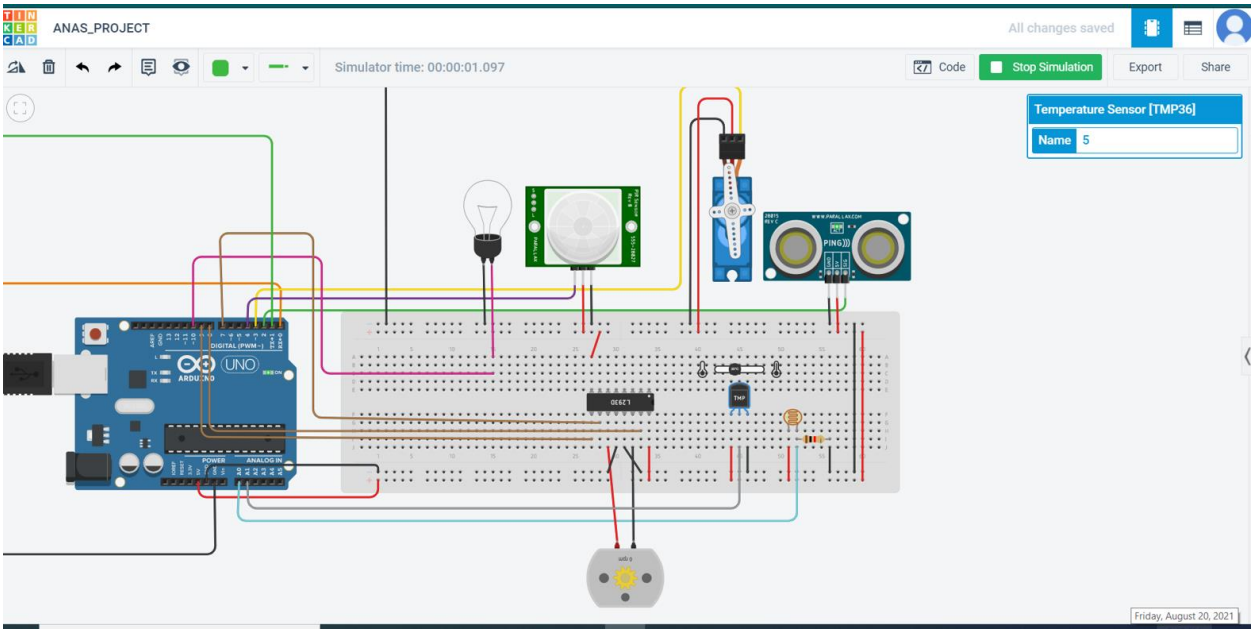

Figure 45:Fan Off at Temperature Lower Than 35 C

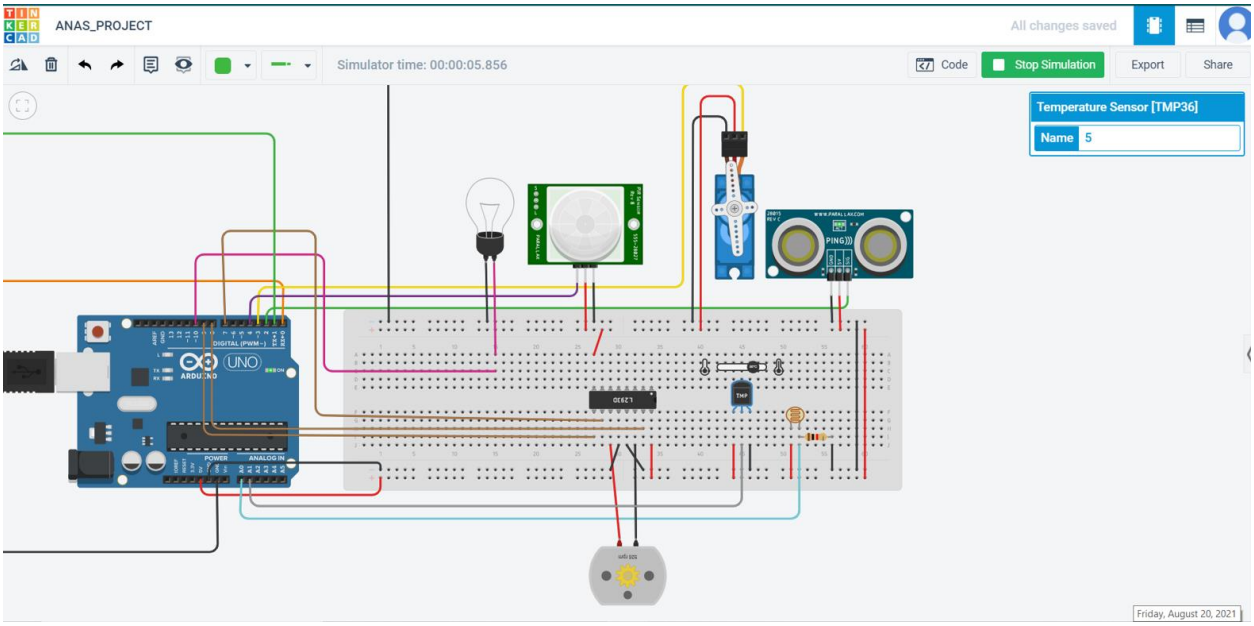

Figure 46:Fan on at Temperature Greater Than 35 C

4.2. Manual Mode

4.2.1. Smart Home Door (Default Value => Close):

- Start simulation and press on Power button then press on button 4 to make door open or close **(in this project we make button 4 if we press on it at first time the door will open and if press on it at second time the door will close).**

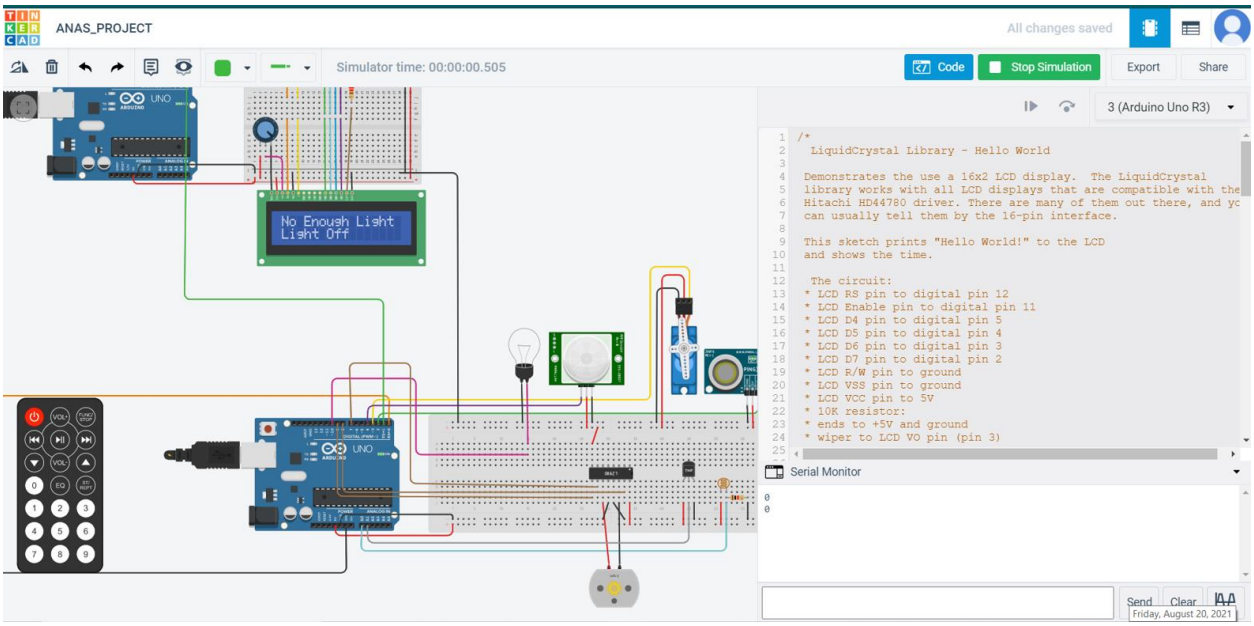

Figure 47:Door Default Value

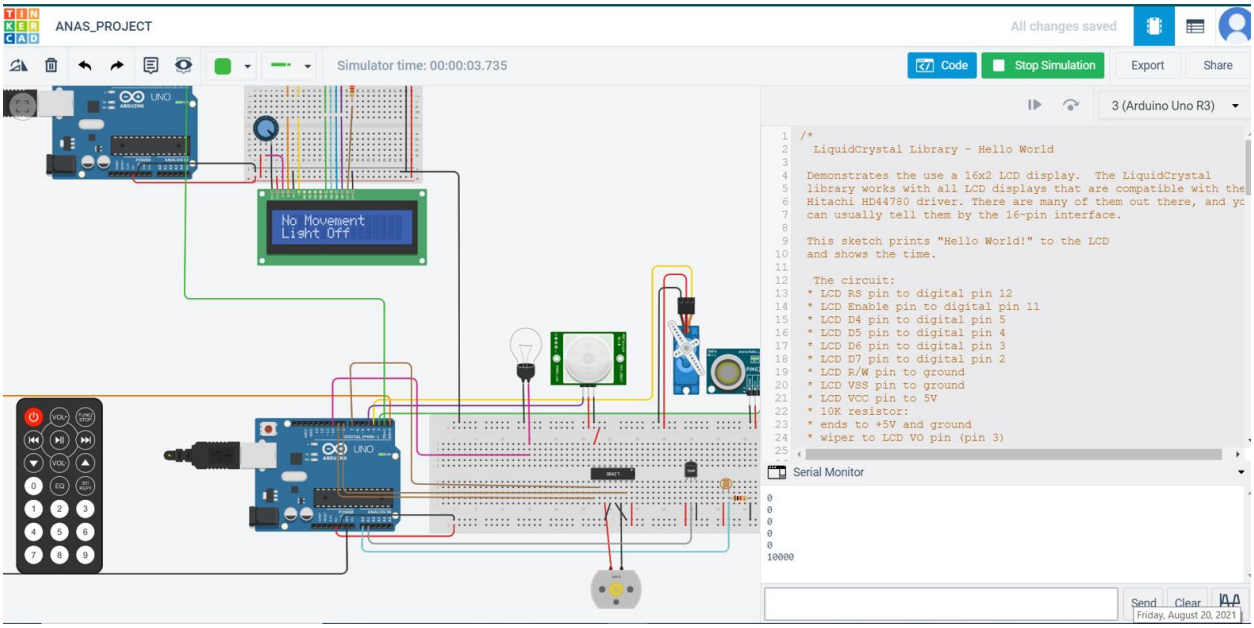

Figure 48:Open the door After push on 4

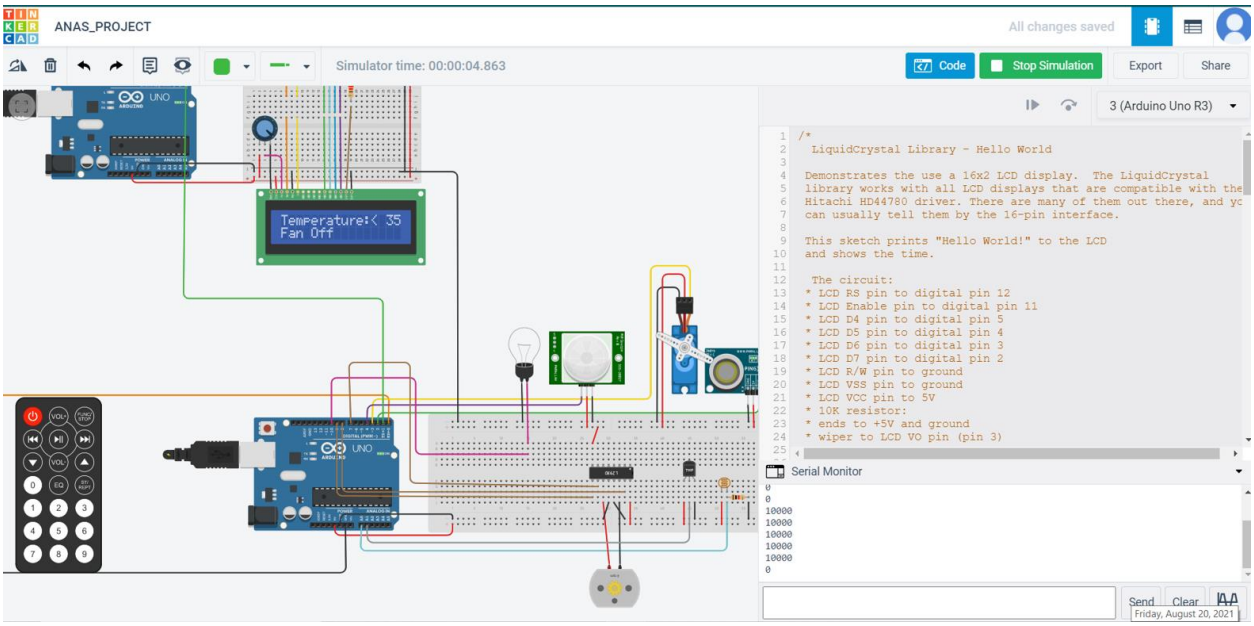

Figure 49:Close the door After push 4 for the second time

4.2.2. Smart Home Light (Default Value => Off):

- Start simulation and press on Power button then press on button 5 to make light bulb turn on or turn off **(in this project we make button 5 if we press on it at first time the light bulb will turn on and if press on it at second time the light bulb will turn off).**

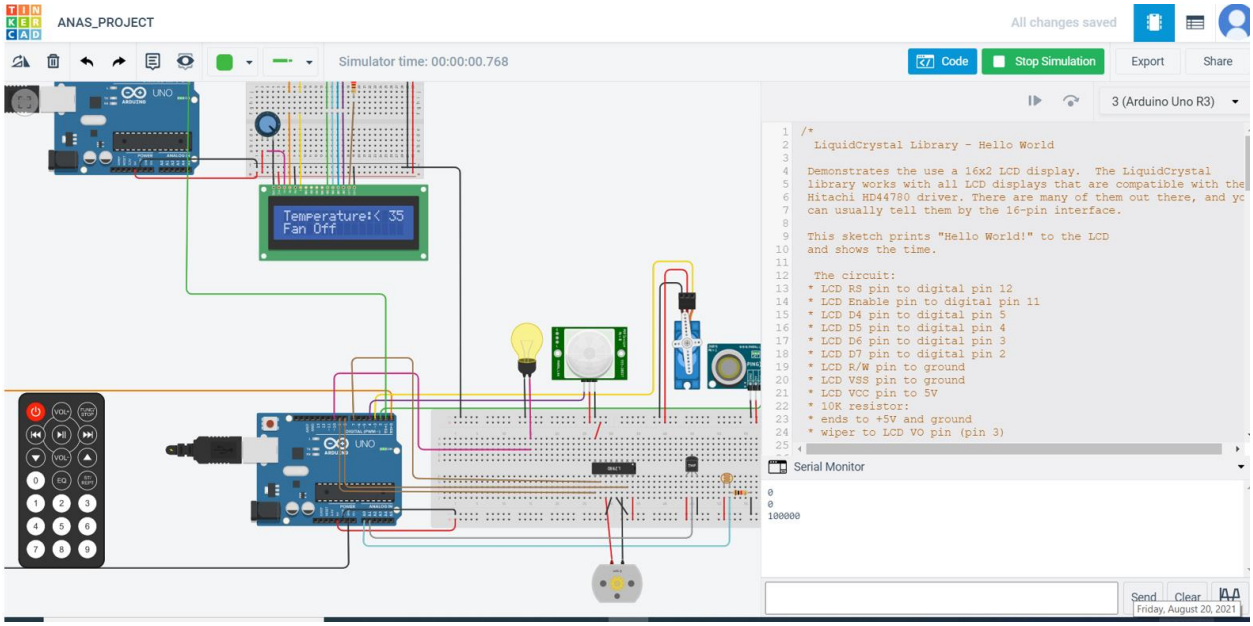

Figure 50:Turn on the Light After push 5

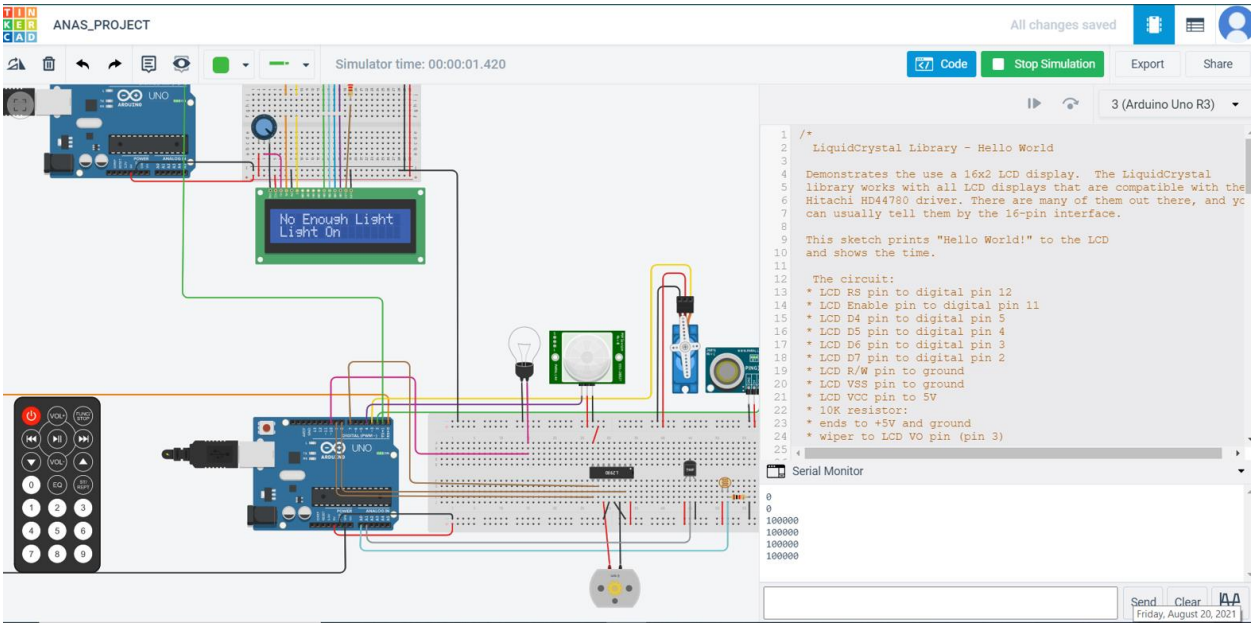

Figure 51: Turn Off the Light After push 5 for another time

4.2.3. Smart Home Fan (Default Value => Off):

- Start simulation and press on Power button then press on button 6 to make fan turn on or turn off **(in this project we make button 6 if we press on it at first time the fan will turn on and if press on it at second time the fan will turn off).**

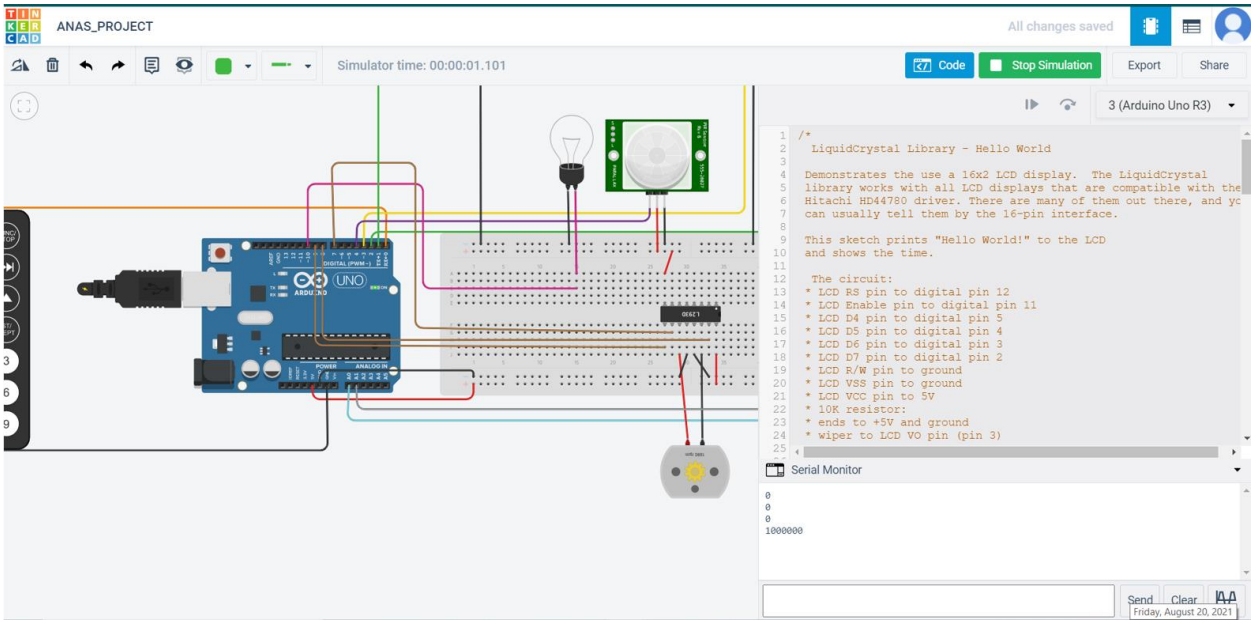

Figure 52 (Turn On the Fan After push 6)

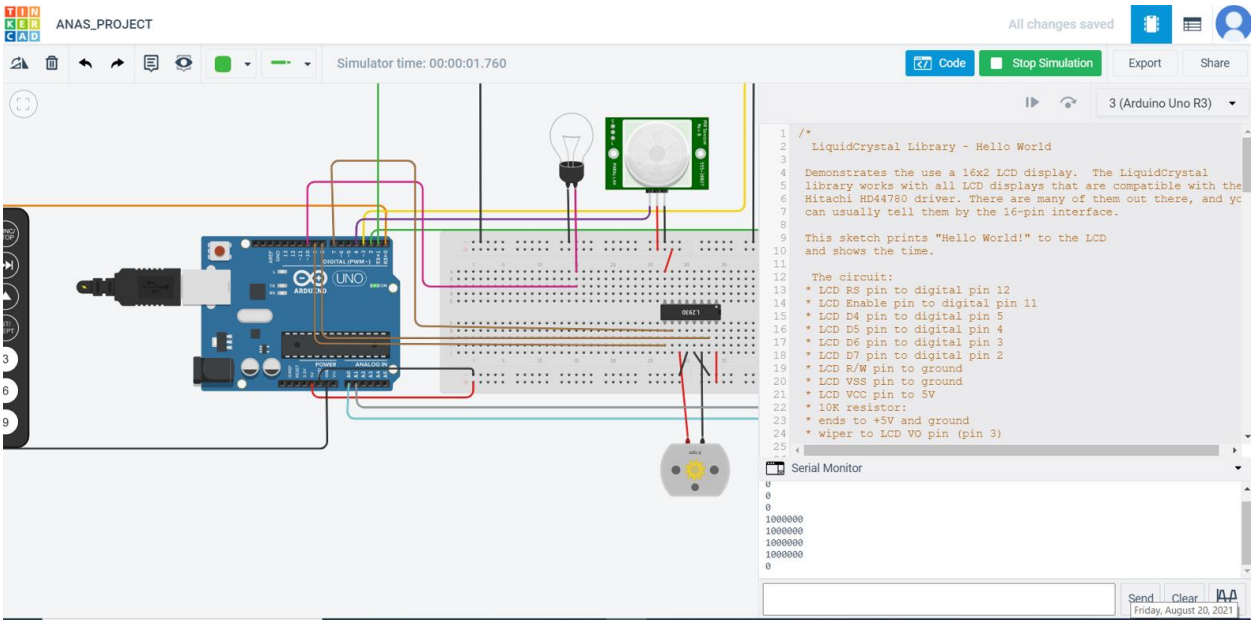

Figure 53 (Turn Off the Fan After push 6 for another time)

5. Conclusion:

In this project we learn how to make creative thing using Arduino also we learn a new chips and libraries used to make the goal, also how to connect between many of Arduinos by using different of ways.

6. References

[1] https://www.youtube.com/channel/UCdTV5phGMx4LbNrohWBc98g [Accessed 5:15 Aug 2021].

[2] https://youtu.be/VwPMm-v3970 [Accessed 6:00 Aug 2021].

[3] https://www.tinkercad.com/things/iynCadPAY7O-anas1180180project/editel?sharecode=0bVyyFu-fG8t-hODPvqKkRHHta4DAw-eStP_JmhVvKI [my project share in tinkercade]

7. Appendix

7.1. A.1:

```
#include <Servo.h>


/* Variables */
bool mode     = false; /* Manual disabled initially */
int button    = -1;
int doorPeriod = 3;
bool doorOpen  = false;
int doorDelay;
int distance;
int LDR;
int PIR;
int lightControl = 3;
double tmp;
int fanSpeed;
bool lightOn = false;
bool fanOn = false;


/* Pins */
const int pingPin       = 2;
const int servoPin      = 3;
const int LDRPin        = 0;
const int PIRPin        = 4;
const int lightPin      = 10;
const int temperaturePin = A1;
const int motorPin1     = 7;
const int motorPin2     = 8;
const int motorPin3     = 9;
```

```
/* States */
int state = 0;
int sent = 0;

// Servo motor
Servo door;

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



  pinMode(servoPin, OUTPUT);
  pinMode(LDRPin, INPUT);
  pinMode(PIRPin, INPUT);
  pinMode(lightPin, OUTPUT);
  pinMode(temperaturePin, INPUT);
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(motorPin3, OUTPUT);

  // Servo Motor
  door.write(0);
  door.attach(servoPin);
}

void loop(){

  if(Serial.available()){
```

```
  button = Serial.read();
  if(button == 0){
    mode = !mode;
          }
}
// Set the controlability of light bulb
/* 1: light depends on PIR only
*  2: light depends on LDR only
*  3: light depends on LDT and PIR together
*/
if(button > 0 && button < 4){
          lightControl = button;
}

// Reading Sensors
distance = readUltrasonicDistance(pingPin);
LDR = analogRead(LDRPin);
PIR = digitalRead(PIRPin);
tmp = readTmp(temperaturePin);
fanSpeed = map(tmp, -40, 125, 0, 255);

// Automatic mode
if(mode == 0){

  // Servo
  if(!doorOpen || millis() - doorDelay >= doorPeriod * 1000){
    if(distance <= 100){
      door.write(90);
      doorDelay = millis();
```

```
    doorOpen = true;
  }else{
        door.write(0);
    doorOpen = false;
  }
}

// Light Bulb
if(lightControl == 1){
  if(PIR){
    analogWrite(lightPin, 255);
    lightOn = true;
  }else{
    analogWrite(lightPin, 0);
    lightOn = false;
  }
}else if(lightControl == 2){
  if(LDR < 500){
    analogWrite(lightPin, 255);
    lightOn = true;
  }else{
    analogWrite(lightPin, 0);
    lightOn = false;
  }
}else if(lightControl == 3){
  if(LDR < 500 && PIR){
    analogWrite(lightPin, 255);
    lightOn = true;
  }else{
```

```
      analogWrite(lightPin, 0);
      lightOn = false;
    }
  }

  // Fan
  if(tmp > 35){
    digitalWrite(motorPin1, HIGH);
    digitalWrite(motorPin2, LOW);
    analogWrite(motorPin3, fanSpeed);
    fanOn = true;
  }else{
    digitalWrite(motorPin1, HIGH);
    digitalWrite(motorPin2, HIGH);
    analogWrite(motorPin3, 0);
    fanOn = false;
  }

}else{
  // Open the door if closed, and close if opened
  if(button == 4 && !doorOpen){
    door.write(90);
    doorOpen = true;
    button = -1;
  }else if(button == 4 && doorOpen){
    door.write(0);
    doorOpen = false;
    button = -1;
  }
```

```
  // Turn the light on if off, and off if on
  if(button == 5 && !lightOn){
    analogWrite(lightPin, 255);
    lightOn = true;
    button = -1;
  }else if(button == 5 && lightOn){
    analogWrite(lightPin, 0);
    lightOn = false;
    button = -1;
  }
  // Turn the fan on if off, and off if on
  if(button == 6 && !fanOn){
    digitalWrite(motorPin1, HIGH);
    digitalWrite(motorPin2, LOW);
    analogWrite(motorPin3, fanSpeed);
    fanOn = true;
    button = -1;
  }else if(button == 6 && fanOn){
    digitalWrite(motorPin1, HIGH);
    digitalWrite(motorPin2, HIGH);
    analogWrite(motorPin3, 0);
    fanOn = false;
    button = -1;
  }
}

 /* Setting state byte 00000000
 ----------------------
```

```
bit   value  meaning
0     0      Distance: > 1m
0     1      Distance: < 1m

1     0      No Movement
1     1      Movement

2     0      No Enough Light
2     1      Enough Light

3     0      Temperature: < 35
3     1      Temperature: > 35

4     0      Door Closed
4     1      Door Open

5     0      Light Off
5     1      Light On

6     0      Fan Off
6     1      Fan On
*/
if(distance <= 100){
  state |= 0b00000001;
}else{
  state &= 0b11111110;
}
if(PIR){
  state |= 0b00000010;
```

```
}else{
  state &= 0b11111101;
}
if(LDR < 500){
  state &= 0b11111011;
}else{
  state |= 0b00000100;
}
if(tmp < 35){
  state &= 0b11110111;
}else{
  state |= 0b00001000;
}
if(!doorOpen){
  state &= 0b11101111;
}else{
  state |= 0b00010000;
}
if(!lightOn){
  state &= 0b11011111;
}else{
  state |= 0b00100000;
}
if(!fanOn){
  state &= 0b10111111;
}else{
  state |= 0b01000000;
}
```

```
  if(sent == 15){
            Serial.write(state);
    sent = 0;
  }
  sent++;
}

double readTmp(int pin){
  int pinReading = analogRead(temperaturePin);
  double voltage = (pinReading * 5.0)/1024;
  double milliVolt = voltage * 1000;
  return (milliVolt-500)/10;
}

double readUltrasonicDistance(int pin){
  // set pin to output mode
  pinMode(pin, OUTPUT);

  // send a signal
  digitalWrite(pin, LOW);
  delayMicroseconds(2);
  digitalWrite(pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(pin, LOW);

  // set pin to input mode
  pinMode(pin, INPUT);

  // get duration in microseconds
```

```
  int duration = pulseIn(pin, HIGH);
  return duration / 29 / 2;
}
```

7.2. A.2:

```
/* IR Remote */
#include <IRremote.h>
IRrecv irrecv(2);
decode_results results;

/* Remote Buttons Mapping */
#define power 0xFD00FF
#define one   0xFD08F7
#define two   0xFD8877
#define three 0xFD48B7
#define four  0xFD28D7
#define five  0xFDA857
#define six   0xFD6897
void setup(){

  /* Start Remote */
  irrecv.enableIRIn();
  Serial.begin(9600);
}

void loop(){

  /* Read remote button */
```

```
if(irrecv.decode(&results)){
  int button;
  switch(results.value){
        case power:
                        button = 0;  /* map with power button to change
                between two mode (Aoutmatic,Manual) */

        break;
        case one:
                 button = 1; /* map with button 1 to make light bulb
                depends on PIR only */
        break;
        case two:
                 button = 2; /* map with button 2 to make light bulb
                depends on LDR only */
        break;
        case three:
                 button = 3; /* map with button 3 to make light bulb
                depends on PIR and LDR */
        break;
        case four:
                 button = 4; /* map with button 4 to open or close the door*/
        break;
        case five:
                 button = 5; /* map with button 5 to on or off the light*/
        break;
    case six:
                 button = 6; /* map with button 6 to on or off the fan(DC_motor)*/
        break;
```

```
    }
    irrecv.resume();  /* to use Automatic mode*/
    Serial.write(button);
  }
}
```

7.3. A.3:

```
/*
 LiquidCrystal Library - Hello World

 Demonstrates the use a 16x2 LCD display.  The LiquidCrystal
 library works with all LCD displays that are compatible with the
 Hitachi HD44780 driver. There are many of them out there, and you
 can usually tell them by the 16-pin interface.

 This sketch prints "Hello World!" to the LCD
 and shows the time.

  The circuit:
 * LCD RS pin to digital pin 12
 * LCD Enable pin to digital pin 11
 * LCD D4 pin to digital pin 5
 * LCD D5 pin to digital pin 4
 * LCD D6 pin to digital pin 3
 * LCD D7 pin to digital pin 2
 * LCD R/W pin to ground
 * LCD VSS pin to ground
 * LCD VCC pin to 5V
```

```
* 10K resistor:
* ends to +5V and ground
* wiper to LCD VO pin (pin 3)

Library originally added 18 Apr 2008
by David A. Mellis
library modified 5 Jul 2009
by Limor Fried (http://www.ladyada.net)
example added 9 Jul 2009
by Tom Igoe
modified 22 Nov 2010
by Tom Igoe

This example code is in the public domain.

http://www.arduino.cc/en/Tutorial/LiquidCrystal
*/

// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 6);
String states[4][2] = {
   {"Distance: > 1m","Door Closed"},
   {"No Movement", "Light Off"},
   {"No Enough Light", "Light Off"},
   {"Temperature:< 35", "Fan Off"}
   };
```

```
int state;
int bits[7] = {0, 0, 0, 0, 0, 0, 0};
int i = 0;
int turnCount = 15;

void setup() {
  Serial.begin(9600);
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
}

void loop(){

  if(Serial.available()){
    state = Serial.read();
    Serial.println(state, BIN);
    bits[0] = getBit(state, 0);
    bits[1] = getBit(state, 1);
    bits[2] = getBit(state, 2);
    bits[3] = getBit(state, 3);
    bits[4] = getBit(state, 4);
    bits[5] = getBit(state, 5);
    bits[6] = getBit(state, 6);

    if(bits[0] == 0)
      states[0][0] = "Distance: > 1m";
    else
      states[0][0] = "Distance: < 1m";
```

```
if(bits[1] == 0)
  states[1][0] = "No Movement";
else
  states[1][0] = "Movement";

if(bits[2] == 0)
  states[2][0] = "No Enough Light";
else
  states[2][0] = "Enough Light";

if(bits[3] == 0)
  states[3][0] = "Temperature:< 35";
else
  states[3][0] = "Temperature:> 35";

if(bits[4] == 0)
  states[0][1] = "Door Closed";
else
  states[0][1] = "Door Open";

if(bits[5] == 0){
  states[1][1] = "Light Off";
  states[2][1] = "Light Off";
}else{
  states[1][1] = "Light On";
  states[2][1] = "Light On";
}
```

```
  if(bits[6] == 0)
    states[3][1] = "Fan Off";
  else
    states[3][1] = "Fan On";
 }

 // Printing the current state on the lcd
 lcd.setCursor(0, 0);
 lcd.print(states[i][0]);
 lcd.setCursor(0, 1);
 lcd.print(states[i][1]);

 // To alternate the shown state
 if(turnCount == 0){
  lcd.clear();
        i = (i + 1) % 4;
  turnCount = 15;
 }
 turnCount--;
}
int getBit(int b, int bitIndex){
        return (b & (1 << bitIndex)) != 0;
}
```