**PIR-sensor based Electronic device control with**

**Voltage regulation**

|  |  |
| --- | --- |
|  | **PROJECT REPORT** |

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| S.No | TITLE | PAGE No. |
| 1 | Abstract | 4 |
| 2 | Block diagram | 5 |
| 3 | Microcontroller | 5 |
| 4 | Passive infrared sensor(PIR sensor) | 6 |
| 5 | Light dependent resistor(LDR) | 7 |
| 6 | Ambient Temperature sensor (LM35) | 8 |
| 7 | Sensors Testing | 9 |
| 8 | Electrical Device On/Off Control | 15 |
| 9 | Voltage regulation Unit | 25 |
| 10 | Coding | 29 |
| 11 | Outputs | 38 |
| 12 | Conclusion and references | 42 |

ABSTRACT

*In this project we present a way to control the voltage regulation of a PIR-sensor-based lighting device. Generally, although a PIR-sensor-based lighting device will turn on when motion is detected and turn off when the motion is no longer present, our circuit supplies the lamp with power when motion is detected; when the motion disappears it turns the lamp off, and the electric power is shut off to reduce the standby power. We use an MCU receiving signals from a PIR sensor which detects any individual approaching thedevice. The MCU controls the SSR On/Off when used as a lightswitch for shutting off the standby power. The MCU monitoringprogram provides automatic detection of any individual bymeans of the PIR sensor. The MCU has internal modules tosimplify the hardware circuit design1.*

*In voltage regulation part we use zero cross detection to control the voltage. The voltage control is in accordance with the ambience.*

Block diagram:

Light/Fan

Voltage regulation Unit

LDR Sensor

LM35 – Temperature Sensor

PIR Sensor

Arduino

Microcontroller used:

Arduino Uno R3

**Arduino** is an open source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world.Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (*shields*) and other circuits. The boards feature serial communications interfaces.

Sensors used:

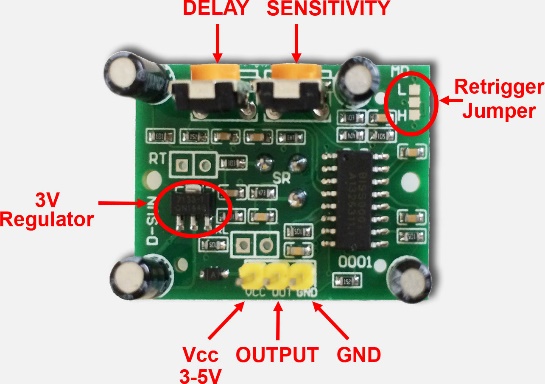
**Passive infrared sensor (PIR sensor):**

A **passive infrared sensor** (**PIR sensor**) is an electronic sensor that measure infrared (IR) light radiating from objects in its field of view. All objects with a temperature above [absolutezero](https://en.wikipedia.org/wiki/Absolute_zero) emit [heat](https://en.wikipedia.org/wiki/Heat) energy in the form of radiation. Usually this radiation isn't visible to the [humaneye](https://en.wikipedia.org/wiki/Human_eye) because it radiates at infrared wavelengths, but it can be detected by electronic devices designed for such a purpose.

The term *passive* in this instance refers to the fact that PIR devices do not generate or radiate energy for detection purposes. They work entirely by detecting infrared radiation emitted by or reflected from objects.

Specs:

* **Power supply:** 5V-12V input voltage for most modules (they have a 3.3V regulator), but 5V is ideal in case the regulator has different specs
* **Output:**Digital pulse high when triggered (motion detected) digital low when idle (no motion detected). Pulse lengths are determined by resistors and capacitors on the PCB and differ from sensor to sensor.
* **Sensitivity range:** up to 20 feet (6 meters) 110° x 70° detection range
* Delay Time and Sensitivity adjustable.

**Ambient light sensor & Temperature sensor**:

**Ambient light/temperature sensors** are used to detect the environment around the room. For example, in case of low room temperature, the fan would rotate at maximum speed. Thus we control things according to ambience around.

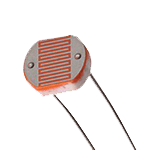
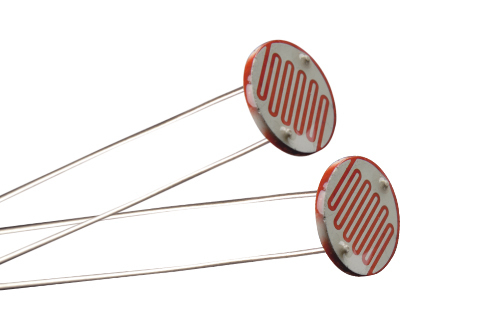
Here, they are used to detect continuously the ambient conditions in order to regulate the AC power depending on the values obtained.

**Light Dependent Resistor (LDR):**

LDR is known as ‘Light Dependent Resistor’. It varies it resistance based on the intensity of the light that falls on its surface.

Specs:

* Resistance: 20K to 100K Ω
* Dark Resistance: 1M Ω
* Operating Temperature Range: -60°C to 75 °C
* Voltage rating used: 5V
* Power dissipation: 50mW

LDR produces low value when the light intensity increases and produces a high value when light falling on it is low.

Lux value α 1/Light Intensity

LUX is the unit used to calculate the light intensity.

Calculation:

Volts = (sensorValue/1023.0)\*5.0 => ADC conversion

Rldr= 500/Lux => Lux = 500/Rldr

Vout = 5\*Rldr/(1+Rldr) 1k Resistor is used

* Lux = (2500/volts-500)/1

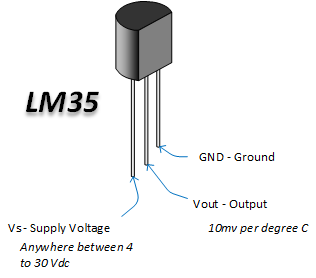
**Ambient Temperature sensor – LM35:**

Temperature sensor – LM35 is used to detect the ambient temperature.

The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly-proportional to the Centigrade temperature.

**Features:**

* Calibrated Directly in Celsius (Centigrade)
* Linear + 10-mV/°C Scale Factor
* 0.5°C Ensured Accuracy (at 25°C)
* Rated for Full −55°C to 150°C Range



Calculation:

Val – Analog value read from sensor

Celsius value = ( Val/1024.0)\*500 => ADC conversion

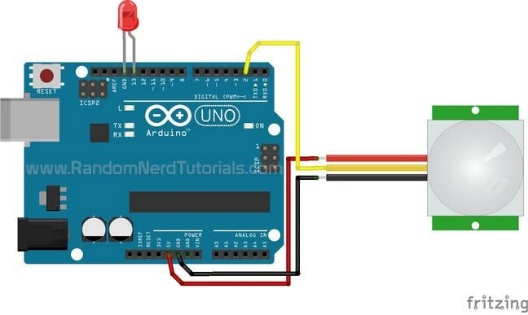
5 volts is used to power the sensor.

(10mv per degree C)

Sensors Tested and implemented:

* PIR sensor was tested and checked for motion of objects. The PIR sensor produced HIGH value when it sensed motion and LOW when no motion was detected.
* LDR sensor was tested to detect the intensity of the light in a room. The LDR gave different Analog output values for different intensities of light.
* The overall circuit was build in which,
  + - PIR was used to detect motion.
    - LDR was used to detect intensity of ambient light.

**PIR sensor test:**

****

**Code:**

int led = 13; // the pin that the LED is atteched to

int sensor = 2; // the pin that the sensor is atteched to

int val = 0; // variable to store the sensor status (value)

void setup() {

pinMode(led, OUTPUT); // initalize LED as an output

pinMode(sensor, INPUT); // initialize sensor as an input

Serial.begin(9600); // initialize serial

}

void loop(){

val = digitalRead(sensor); // read sensor value

if (val == HIGH) { // check if the sensor is HIGH

digitalWrite(led, HIGH); // turn LED ON

delay(1000); // delay 100 milliseconds

Serial.println("Motion detected!");

}

else {

digitalWrite(led, LOW); // turn LED OFF

delay(1000); // delay 200 milliseconds

Serial.println("Motion stopped!");

}

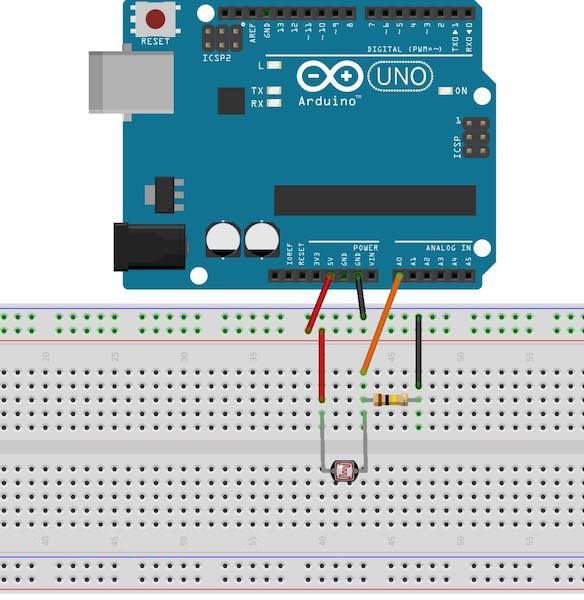
}

Output:

When Motion detected – LED ON

When No Motion detected – LED OFF

**LDR sensor test:**

****

**Code:**

int sensorPin = A0; // select the input pin for LDR

int sensorValue = 0; // variable to store the value coming from the sensor

void setup() {

pinMode(sensorPin,INPUT);

Serial.begin(9600); //sets serial port for communication

}

void loop() {

sensorValue = analogRead(sensorPin); // read the value from the sensor

float volts = (sensorValue/1023.0)\*5.0;

int lux=(2500/volts-500)/1;

Serial.print("Sensor value:");

Serial.println(sensorValue); //prints the values coming from the sensor on the screen

Serial.print("Volts:");

Serial.println(volts);

Serial.print("Lux value:");

Serial.println(lux);

delay(3000);

}

LDR code working:

LDR is connected to the Analog Pin A0 and gives value to the Arduino. LDR is connected with a 100k resistor and output is taken along the junction of these two resistors. Thus when light falls on LDR, the resistance value varies and thereby the output voltage varies.

LDR was checked

* In a room with very low light.
* In a room with light turned on.
* Using a torch emitting light over the LDR.
* In sunlight at morning,afternoon,evening.

Output:

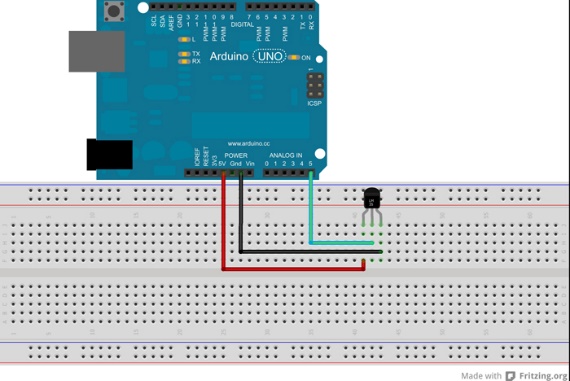
Bright Sunlight : 80 – 150

Dim light : >500

Dark : >1000

Maximum light (Torched) : <50

**LM35 sensor test:**



**Code:**

int val;

int tempPin = A5;

void setup()

{

Serial.begin(9600);

pinMode(tempPin,INPUT);

}

void loop()

{

val = analogRead(tempPin);

float mv = ( val/1024.0)\*5000;

float cel = mv/10;

Serial.print("TEMPRATURE = ");

Serial.print(cel);

Serial.print("\*C");

Serial.println();

delay(1000);

}

Working:

LM35 sensor produces a analog voltage value based on the temperature surrounding it. The voltage level is converted into corresponding Celsius value.

LM35 was checked

* In natural environment.
* Inside an Ac room.

Electrical device ON/OFF Control:

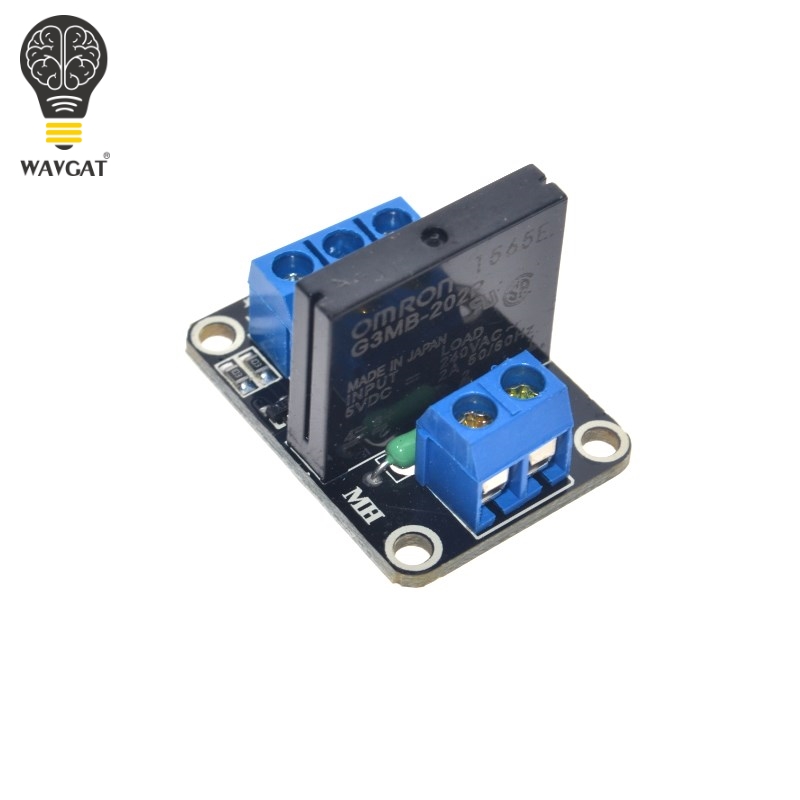
The switching operation was done with the help of a Solid State Relay.

**Relay:**

Relay is an electrical device, typically incorporating an electromagnet, which is activated by a current or signal in one circuit to open or close another circuit. The relay, receiving the signal from DC source is used to switch AC power.

**Solid State Relay:**

Unlike electro-mechanical relays (EMR) which use coils, magnetic fields, springs and mechanical contacts to operate, the solid state relay, or SSR, has no moving parts but instead uses the electrical and optical properties of solid state semiconductors to perform its input to output isolation and switching functions.

**Solid State Relay used:**

Specs:

Input:

* 5V DC

Load:

* 240V AC ~
* 2A 50/60 Hz

Pin details:

Input:

* DC+ :- 5V
* DC- :- Gnd
* CH1 :- DC output from MCU

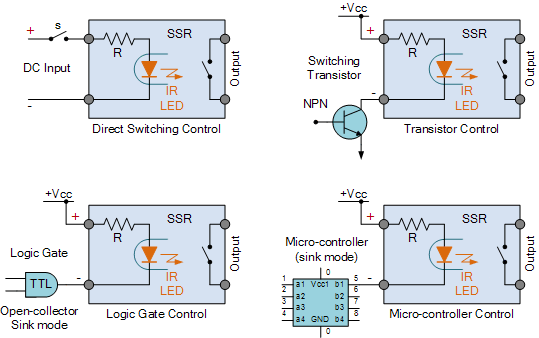
Load:

* AC Supply
* AC Device

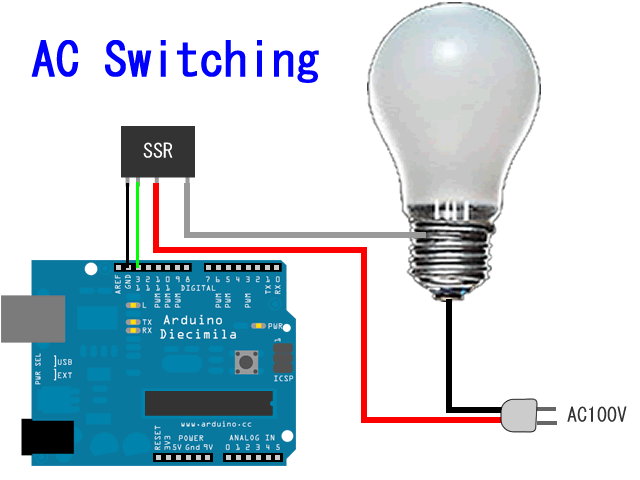
**Working:**

One of the main components of a solid state relay (SSR) is an opto-isolator (also called an optocoupler) which contains one (or more) infra-red light-emitting diode, or LED light source, and a photo sensitive device within a single case.

The opto-isolator isolates the input from the output. The LED light source is connected to the SSR’s input drive section and provides optical coupling through a gap to an adjacent photo sensitive transistor. Thus the output of an optocoupled SSR is turned “ON” by energising this LED, usually with low-voltage signal.



The SSR can be controlled by providing a DC pulse. Here, we use a microcontroller (Arduino) to trigger the SSR. The LED is energized and a photodetector detects the light and performs the switching operation based on the intensity of the light. Thus the AC power is switched on/off to the device.



**Code:**

int relay = 7;

void setup() {

// put your setup code here, to run once:

Serial.begin(9600);

pinMode(relay,OUTPUT);

}

void loop() {

// put your main code here, to run repeatedly:

digitalWrite(relay,LOW); // Turns ON Relays 1

Serial.println("Light On");

delay(2000); // Wait 2 seconds

digitalWrite(relay,HIGH); // Turns Relay Off

Serial.println("Light Off");

delay(2000);

}

The pin mode is setup to provide output to the SSR from the Arduino.

When Write is enabled – SSR turns on.

When Write is disabled – SSR turns off.

Arduino provides a 5V DC voltage to the SSR to turn it.

The following code is for Switching operation of the circuit.

int LDR\_sensor = A5; // select the input pin for ldr

int tempPin = A0; // input pin for lm35

int Lamp\_led = 13; // the pin that the Lamp\_LED is atteched to

int Fan\_led = 12; // the pin that the Fan\_LED is atteched to

int PIR\_sensor = 2; // the pin that the sensor is atteched to

int relay\_light = 7; // relay pin for lamp is attached

int relay\_fan = 8; // relay pin for fan is attached

int val = 0;

int temp\_val;

int ldr\_val = 0;

int fan\_mode = 0;

int light\_mode = 0;

void setup() {

pinMode(PIR\_sensor, INPUT); // initialize PIR sensor as an input

pinMode(LDR\_sensor, INPUT); // initialize LDR sensor as an input

pinMode(tempPin, INPUT); // initializeTemp sensor as an input

pinMode(Lamp\_led, OUTPUT); // initalize Lamp LED as an output

pinMode(Fan\_led, OUTPUT); // initalize Fan LED as an output

pinMode(relay\_light,OUTPUT); // initialize Lamp output

pinMode(relay\_fan,OUTPUT); // initialize Fan output

Serial.begin(9600);

}

void loop(){

val = digitalRead(PIR\_sensor);

// Program for Fan

temp\_val = analogRead(tempPin);

floatcel = ( temp\_val/1024.0)\*500.0;

Serial.print("TEMPRATURE = ");

Serial.print(cel);

Serial.print("\*C");

Serial.println();

if(val == HIGH)

{

if(cel> 30)

{

fan\_mode = 1;

}

else

{

fan\_mode = 0;

digitalWrite(Fan\_led,LOW);

digitalWrite(relay\_fan, HIGH); // turn Fan ON

Serial.println("Motion detected but Fan is OFF");

delay(500);

}

}

else

{

fan\_mode = 0;

Serial.println("No Motion detected(Fan)!");

digitalWrite(Fan\_led, LOW); // turn LED OFF

//digitalWrite(relay\_fan,LOW); // turn Fan OFF

delay(500); // delay 200 milliseconds

}

// Program for Light

ldr\_val = analogRead(LDR\_sensor);

float volts = (ldr\_val/1024.0)\*5.0;

int lux=(2500/volts-500)/1;

Serial.print("Lux value:");

Serial.println(lux);

if(val == HIGH)

{

if(lux <=50)

light\_mode = 1;

else

{

light\_mode = 0;

digitalWrite(Lamp\_led,LOW);

digitalWrite(relay\_light,LOW); // turn Lamp OFF

Serial.println("Motion detected but Lamp is OFF");

delay(500);

}

}

else

{

light\_mode = 0;

Serial.println("No Motion detected(Light)!");

digitalWrite(Lamp\_led, LOW); // turn LED OFF

digitalWrite(relay\_light,LOW); // turn Lamp OFF

delay(500); // delay 200 milliseconds

}

if(fan\_mode == 1 &&light\_mode == 0)

{

Serial.println("Fan is on & Light is off");

digitalWrite(Fan\_led,HIGH);

digitalWrite(relay\_fan,LOW); // turn Fan ON

delay1(4800);

}

else if(fan\_mode == 0 &&light\_mode == 1)

{

Serial.println("Fan is off & Light is on");

digitalWrite(Lamp\_led,HIGH);

digitalWrite(relay\_light,LOW); // turn Lamp ON

delay1(4800);

}

else if(fan\_mode == 1 &&light\_mode == 1)

{

Serial.println("Fan is on & Light is on");

digitalWrite(Fan\_led,HIGH);

digitalWrite(relay\_fan,LOW); // turn Fan ON

digitalWrite(Lamp\_led,HIGH);

digitalWrite(relay\_light,LOW); // turn Lamp ON

delay1(4800);

}

}

void delay1(int x)

{

int b=x;

repeat:

int flag=0;

for(x=b;x>0;x--)

{

Serial.println(x);

if(digitalRead(PIR\_sensor)==HIGH)

flag=1;

}

if(flag==1)

{

Serial.println("Motion detected during high state!");

goto repeat;

}

}

**Explanation for Coding:**

Input: Output:

PIR sensor – Digital pin LED

LM35, LDR – Analog pins Relay=>Light, Fan

The necessary sensors and output devices are connected and initialized. The ambient temperature and light intensity values are recorded. PIR is checked for motion and once detected, the various conditions are checked for.

Once satisfied, delay function is not implemented immediately, but a flag is set and the condition for other device is checked.

At the end of the loop, the flags are checked for HIGH values and if found, the corresponding device(s) is (are) switched on, either individually or together. Thus, this allows independent operation.

A Manual delay is written in order to sense the PIR sensor while the Light/Fan is On, instead of remaining in sleep state during normal delay function.

**Output**:

When Motion detected & Lux <= 50 : Light Turns On

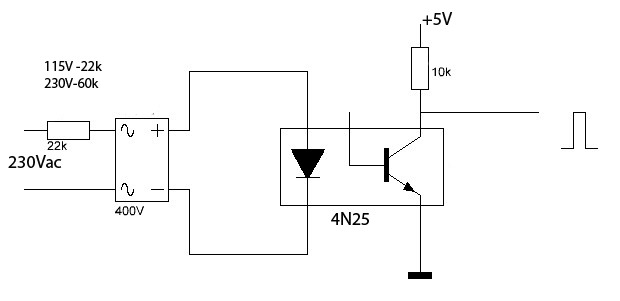
When Motion detected & Temperature >30 : Fan Turns On

Voltage Regulation Unit:

Method:

The method used is:

* Detect zero-crossings using an optocoupler. Here, optocoupler 4N25 is used.



When LED is energized the transistor is switched ON and thereby connection is established. The circuit is established during most of the period except at Zero-crossings of the AC pulse. At zero-crossings, no power is transferred to the opto-coupler and thereby a narrow-pulse is generated at the output end which signals the Arduino.

Once, zero-crossing is detected, the triac is fired after a delay of few microseconds.

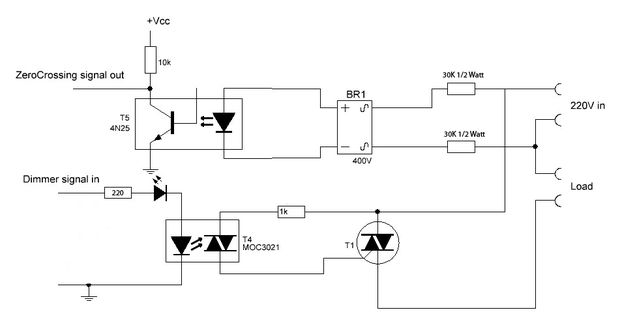
AC Signal: 230V, 50Hz

T=1/50 = 20ms.

Hence, zero-crossings occur every 10ms and thereby this is used as reference to trigger the triac. The triac is triggered using a narrow pulse and it allows the regulated AC power to pass through it. Hence, the bulb glows at a brightness level corresponding to the power given to it.

Also, a bridge rectifier is used to use both halves of AC signal.

Overall circuit:



Components:

* Optocoupler
* Triac
* Bridge rectifier
* AC controlled device – Fan (or) Light

**Optocoupler:**

 Optocoupler or optical isolatoris a component that transfers electrical signals between two isolated circuits by using light. Opto-isolators prevent high voltages from affecting the system receiving the signal.A common type of opto-isolator consists of an LED and a phototransistor in the same opaque package. The LED converts electrical input signal into light. A closed optical channel, and a photosensor, detects incoming light and either generates electric energy directly, or modulates electric current flowing from an external power supply.

Optocouplers used:

4n25

MOC3021

**Triac:**

Triac is basically known as **TRI**ode for **A**lternating**C**urrent.It is a three terminal AC switch which is different from the other SCRs in the sense that it can conduct in both the directions that is whether the applied gate signal ispositive or negative, it will conduct. Thus, this device can be used for AC systems as a switch.

**Working:**

* The output from the microcontroller is fed as input to the LED of the optocoupler which inturn produces an output voltage depending on the light intensity.
* The output voltage of the optocoupler is used to trigger the gate of Triac.
* The triac has an offset, switching at different gate voltages for each half of the cycle.
* Hence the average power across the load will be varied.

Due to the bidirectional control capability of both TRIAC, it is possible to control the firing angle of the TRIAC in both positive and negative peaks of the input.

**Testing:**

void setup()

{

Serial.begin(9600);

attachInterrupt(1, zero\_crosss\_int, RISING);

}

voidzero\_crosss\_int() //function to be fired at the zero crossing to dim the light

{

Serial.println("Zero Cross detected");

}

void loop() {

Serial.println("No"); }

**Overall Working:**

No

Yes

Yes

Turn Fan On

Turn Light On

Voltage Regulation

Check for LDR value

Voltage Regulation

Check for Lm35 value

Is PIR = High?

Wait for Motion

Turn on the circuit

The circuit waits till Motion is detected and then the Ambience light and temperature values are checked for. The voltage regulation is done based on the LDR/Lm35 values obtained.

**Code:**

**Light:**

#define fullOn 10

#define fullOff 127

#define FQ\_50 1 // in case of 50Hz

#define FQ\_60 0 // in case of 50Hz

#define VER "2.0"

intinbyte;

int AC\_LOAD = 11; // Output to OptoTriac pin

int dimming = fullOff; // Dimming level (0-128) 0 = ON, 128 = OFF

intpir = 8; // Input from PIR Sensor

intldr = A0;

booleanfrq = FQ\_50; // change the frequency here.

intcountUp = 0;

boolean simulate = false;

volatileintval = 0;

volatile float ldr\_val = 0.0;

void setup()

{

pinMode(AC\_LOAD, OUTPUT); // Set the AC Load as output

attachInterrupt(1, zero\_crosss\_int, RISING); // Choose the zero cross interrupt # from the table above

Serial.begin(115200);

}

voidzero\_crosss\_int() // function to be fired at the zero crossing to dim the light

{

// Firing angle calculation

// 50Hz-> 10ms (1/2 Cycle) → (10000us - 10us) / 128 = 78 (Approx)

// 60Hz-> 8.33ms (1/2 Cycle) → (8333us - 8.33us) / 128 = 65 (Approx)

intdimtime = 0;

floatpropTime = 0;

if(frq){

dimtime = (78\*dimming);

propTime = 10.0;

}

else{

dimtime = (65\*dimming);

propTime = 8.33;

}

//Serial.println(dimtime);

delayMicroseconds(dimtime); // Off cycle

if (dimtime > 9600)

goto skip;

digitalWrite(AC\_LOAD, HIGH); // triac firing

delayMicroseconds(propTime); // triac On propogation delay

skip:

digitalWrite(AC\_LOAD, LOW); // triac Off

}

void loop() {

val = digitalRead(pir);

if (val != 0)

simulate = true;

repeat:

ldr\_val = analogRead(ldr); // read the value from the sensor

float volts = (ldr\_val/1023.0)\*5.0;

int lux=(2500/volts-500)/1;

if (lux > 0 && lux <= 200) //Change LDR values here

countUp = 0;

else if (lux > 200 && lux <= 800)

countUp = 1;

else if (lux > 800 && lux <= 1200)

countUp = 2;

else if (lux > 1200 && lux <= 2000)

countUp = 3;

else

countUp = 4;

if(simulate){

int x = 9600;

simulate = false;

Serial.println(countUp);

if(countUp == 0)

dimming = 128;

else if(countUp == 1)

dimming = 90;

else if(countUp == 2)

dimming = 60;

else if(countUp == 3)

dimming = 30;

else

dimming = 15;

int b=x; //Change delay value here

int flag=0;

for(x=b;x>0;x--)

{

Serial.println(x);

if(digitalRead(PIR\_sensor)==HIGH)

flag=1;

}

if(flag==1)

{

simulate = true;

Serial.println("Motion detected during high state!");

goto repeat;

}

}

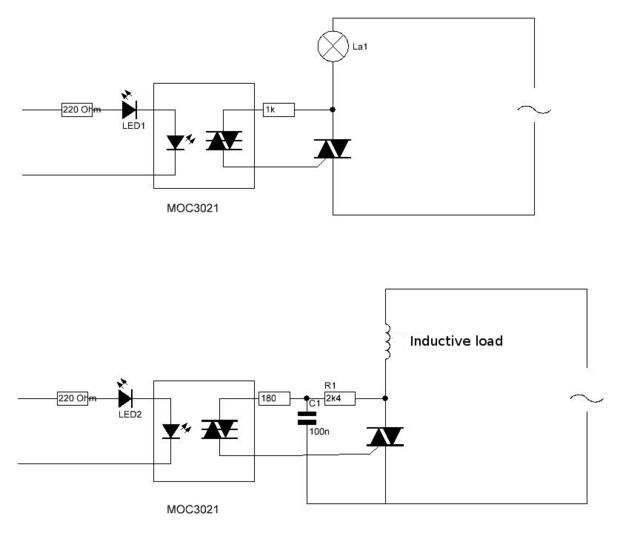
else

dimming = 128;

}

|  |  |
| --- | --- |
| **Lux value** | **Dimming** |
| <200 | 128 |
| 200-800 | 90 |
| 800-1200 | 60 |
| 1200-2000 | 30 |
| >2000 | 15 |

**Fan:**

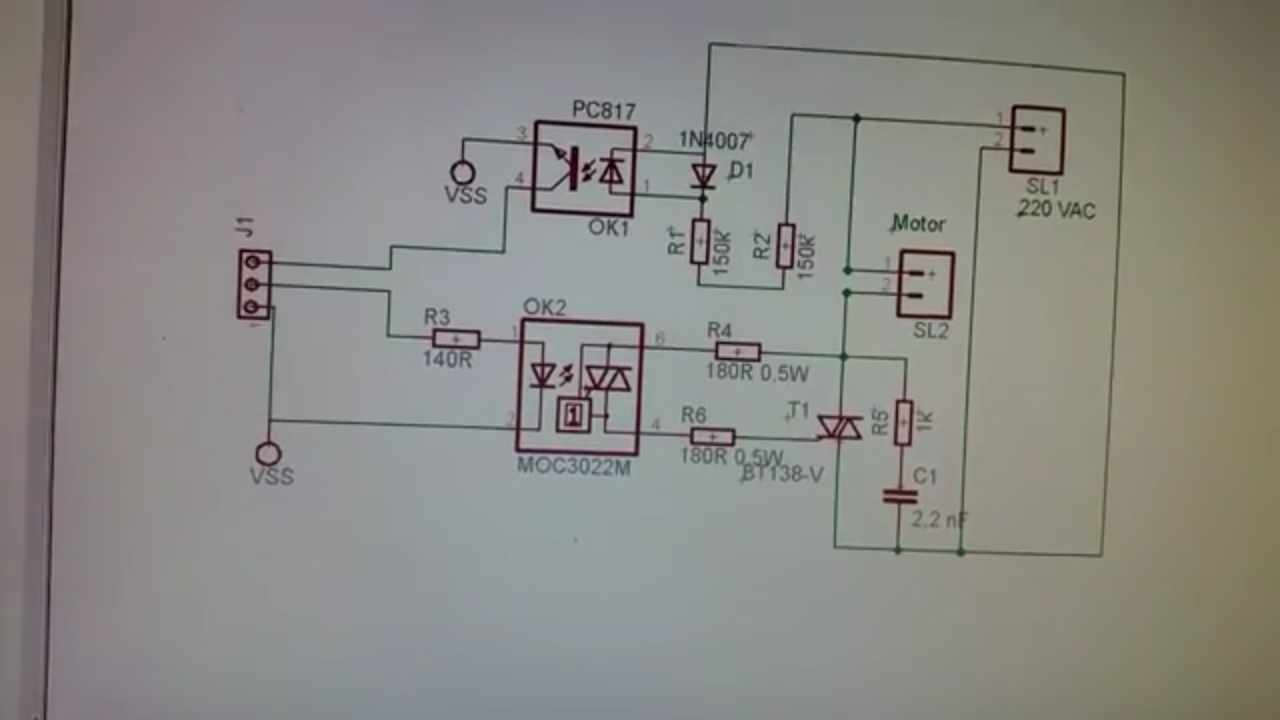


A similar circuit is used for connecting the Fan as a load, except that a Snubber circuit is included.

**Snubber circuit:**

RC **SNUBBERS** (SMPS) **Snubbers** are energy-absorbing **circuits** used to suppress the voltage spikes caused by the **circuit's** inductance when a switch, electrical or mechanical, opens. The most common **snubber circuit** is a capacitor and resistor connected in series across the switch.

**Overall circuit:**

****

**Code:**

#define fullOn 10

#define fullOff 127

#define FQ\_50 1 // in case of 50Hz

#define FQ\_60 0 // in case of 50Hz

#define VER "2.0"

intinbyte;

int AC\_LOAD = 11; // Output to OptoTriac pin

int dimming = fullOff; // Dimming level (0-128) 0 = ON, 128 = OFF

intpir = 8; // Input from PIR Sensor

intldr = A0;

booleanfrq = FQ\_50; // change the frequency here.

intcountUp = 0;

boolean simulate = false;

volatileintval = 0;

volatile float ldr\_val = 0.0;

void setup()

{

pinMode(AC\_LOAD, OUTPUT); // Set the AC Load as output

attachInterrupt(1, zero\_crosss\_int, RISING); // Choose the zero cross interrupt # from the table above

Serial.begin(115200);

}

voidzero\_crosss\_int() // function to be fired at the zero crossing to dim the light

{

// Firing angle calculation

// 50Hz-> 10ms (1/2 Cycle) → (10000us - 10us) / 128 = 78 (Approx)

// 60Hz-> 8.33ms (1/2 Cycle) → (8333us - 8.33us) / 128 = 65 (Approx)

intdimtime = 0;

floatpropTime = 0;

if(frq){

dimtime = (78\*dimming);

propTime = 10.0;

}

else{

dimtime = (65\*dimming);

propTime = 8.33;

}

//Serial.println(dimtime);

delayMicroseconds(dimtime); // Off cycle

if(dimtime > 9600)

goto skip;

digitalWrite(AC\_LOAD, HIGH); // triac firing

delayMicroseconds(propTime); // triac On propogation delay

skip:

digitalWrite(AC\_LOAD, LOW); // triac Off

}

void loop() {

val = digitalRead(pir);

if (val != 0)

simulate = true;

repeat:

val = analogRead(tempPin);

float mv = ( val/1024.0)\*5000;

floatcel = mv/10;

if (cel> 0 &&cel<= 20)

countUp = 0;

else if (cel> 20 &&cel<= 25)

countUp = 1;

else if (cel> 25 &&cel<= 30)

countUp = 2;

else if (cel> 30 &&cel<= 35)

countUp = 3;

else

countUp = 4;

if(simulate){

int x = 9600;

simulate = false;

Serial.println(countUp);

if(countUp == 0)

dimming = 128;

else if(countUp == 1)

dimming = 90;

else if(countUp == 2)

dimming = 60;

else if(countUp == 3)

dimming = 30;

else

dimming = 15;

int b=x; //Change delay value here

int flag=0;

for(x=b;x>0;x--)

{

if(digitalRead(PIR\_sensor)==HIGH)

flag=1;

}

if(flag==1)

{

simulate = true;

Serial.println("Motion detected during high state!");

goto repeat;

}

}

else

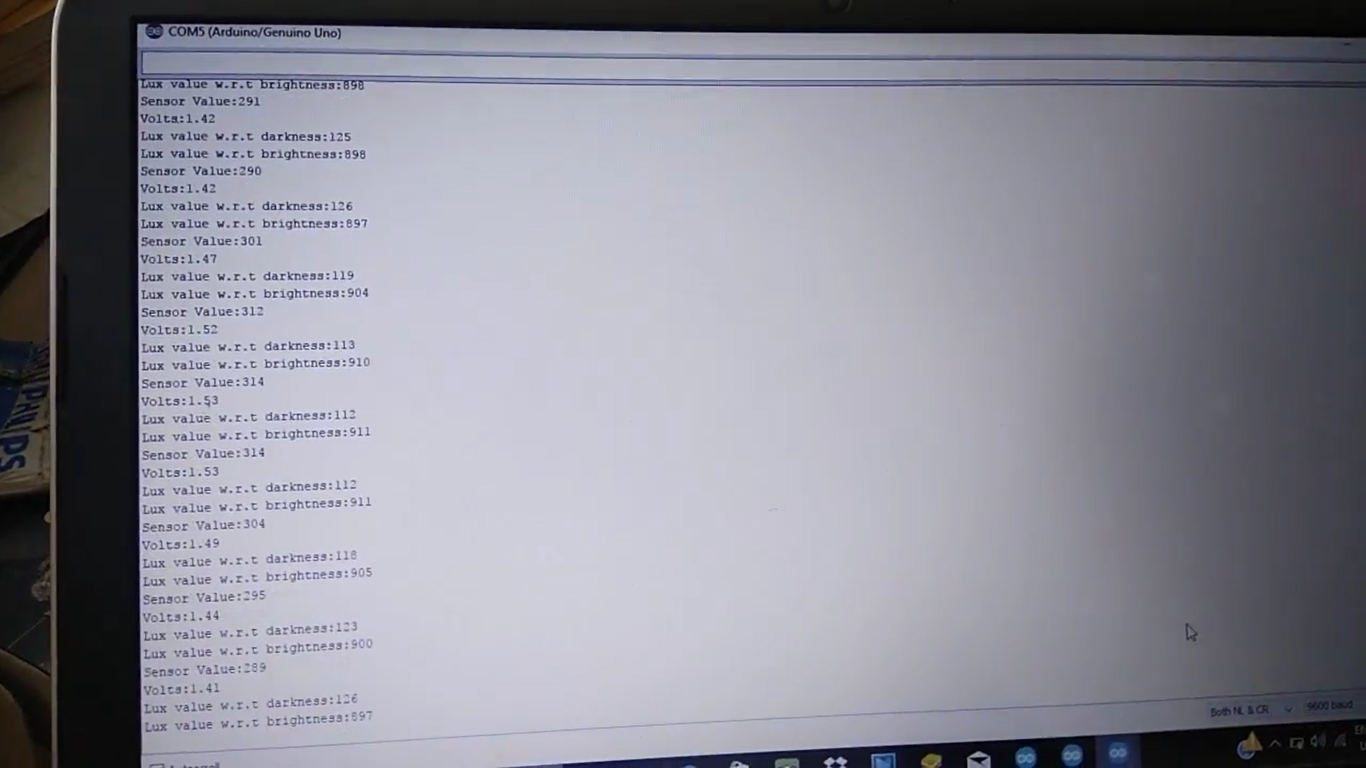
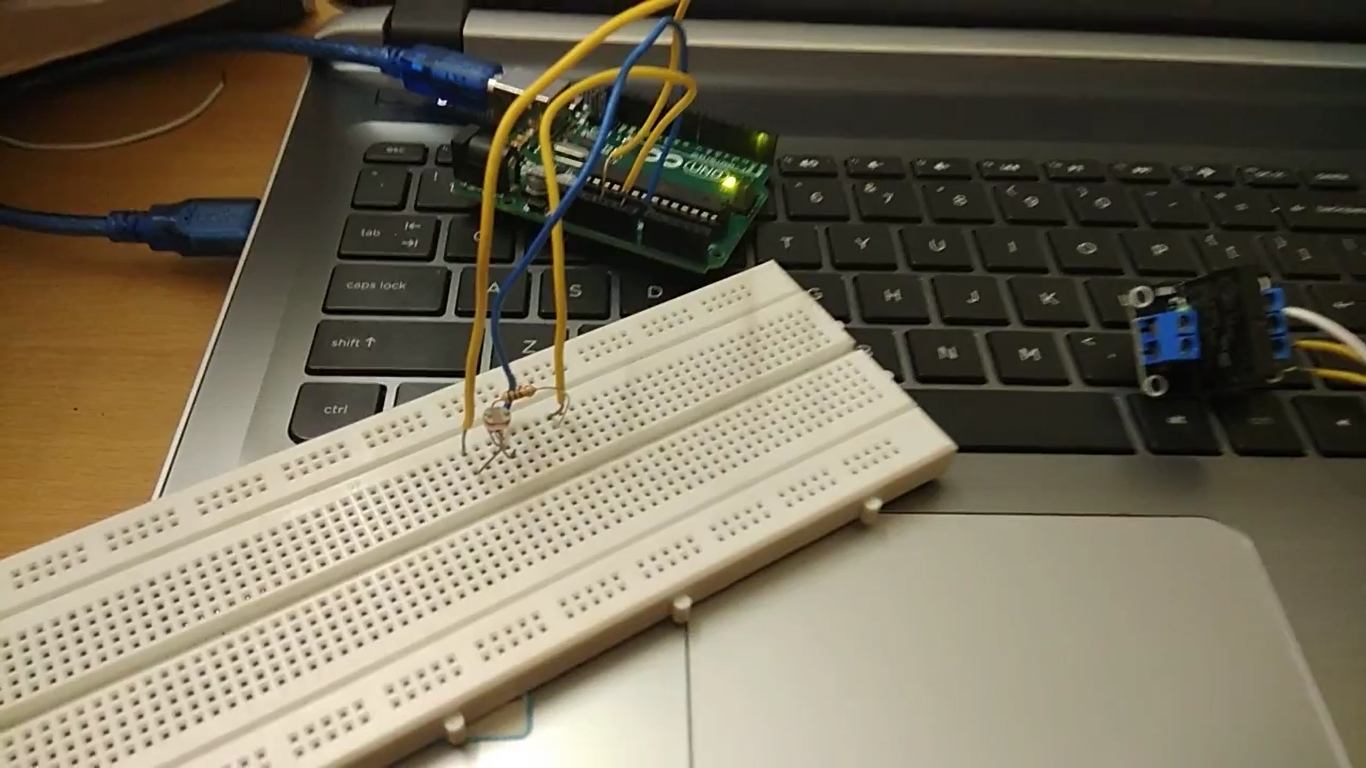
dimming = 128;

}

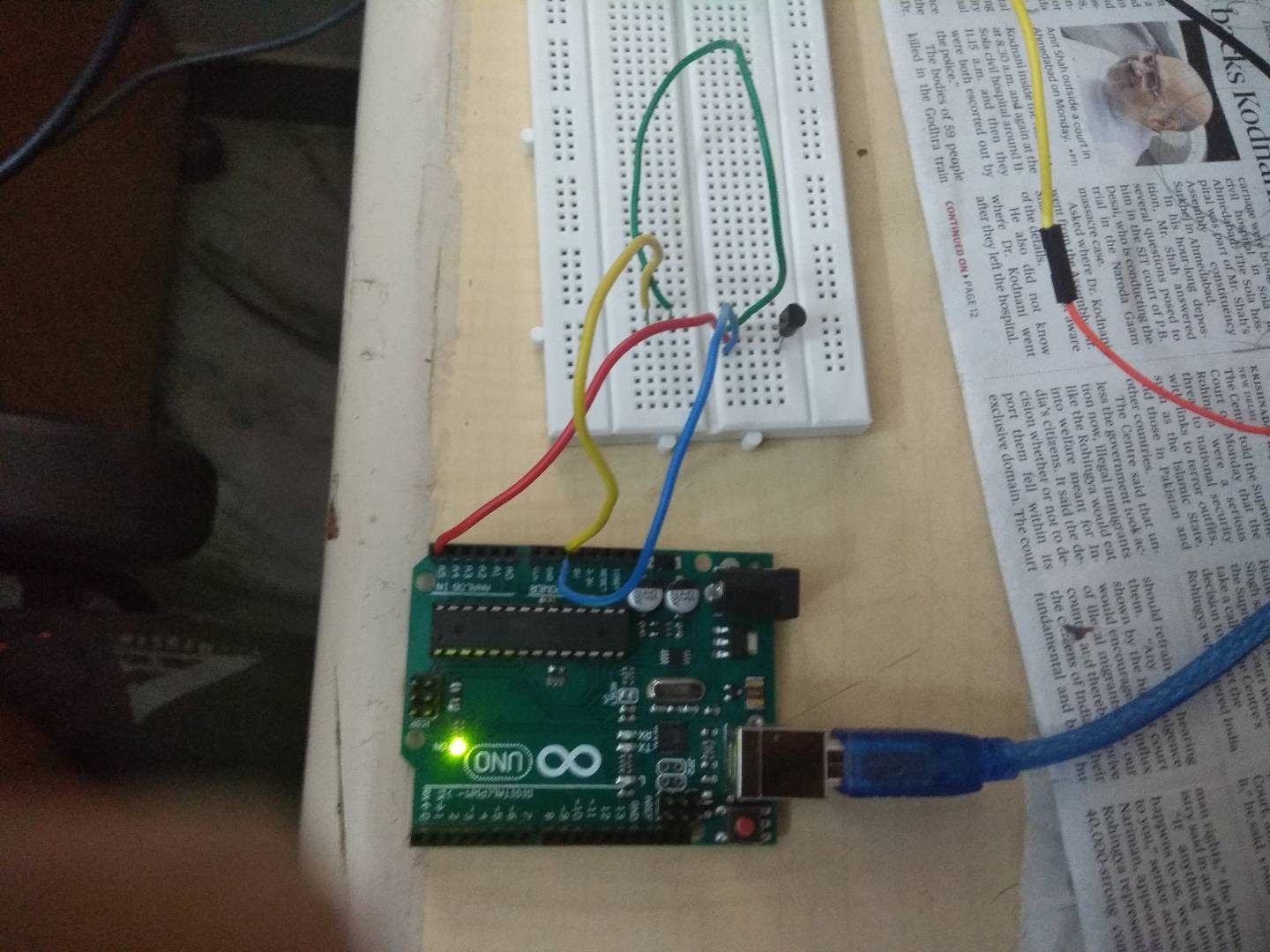
|  |  |
| --- | --- |
| **Temperature in Celsius** | **Dimming** |
| <20 | 128 |
| 20-25 | 90 |
| 25-30 | 60 |
| 30-35 | 30 |
| >35 | 15 |

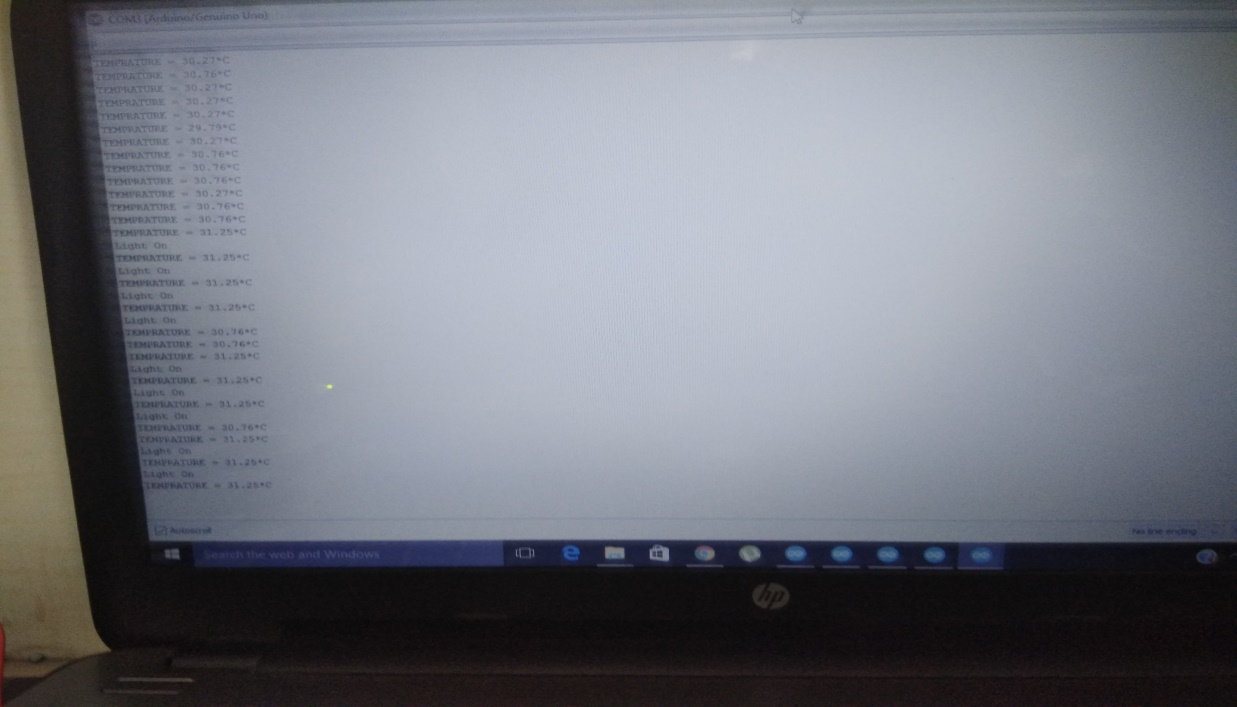
**Outputs:**

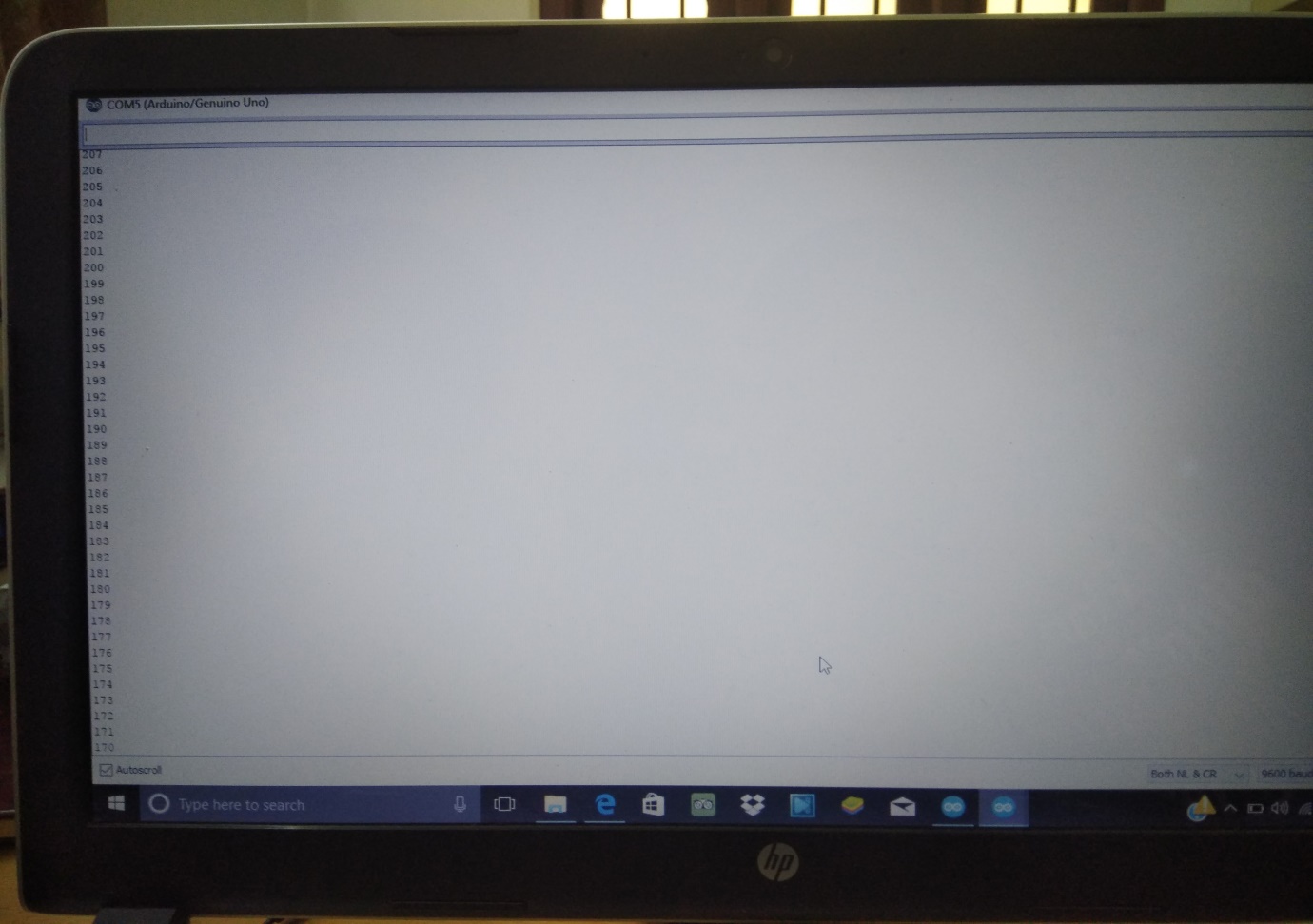
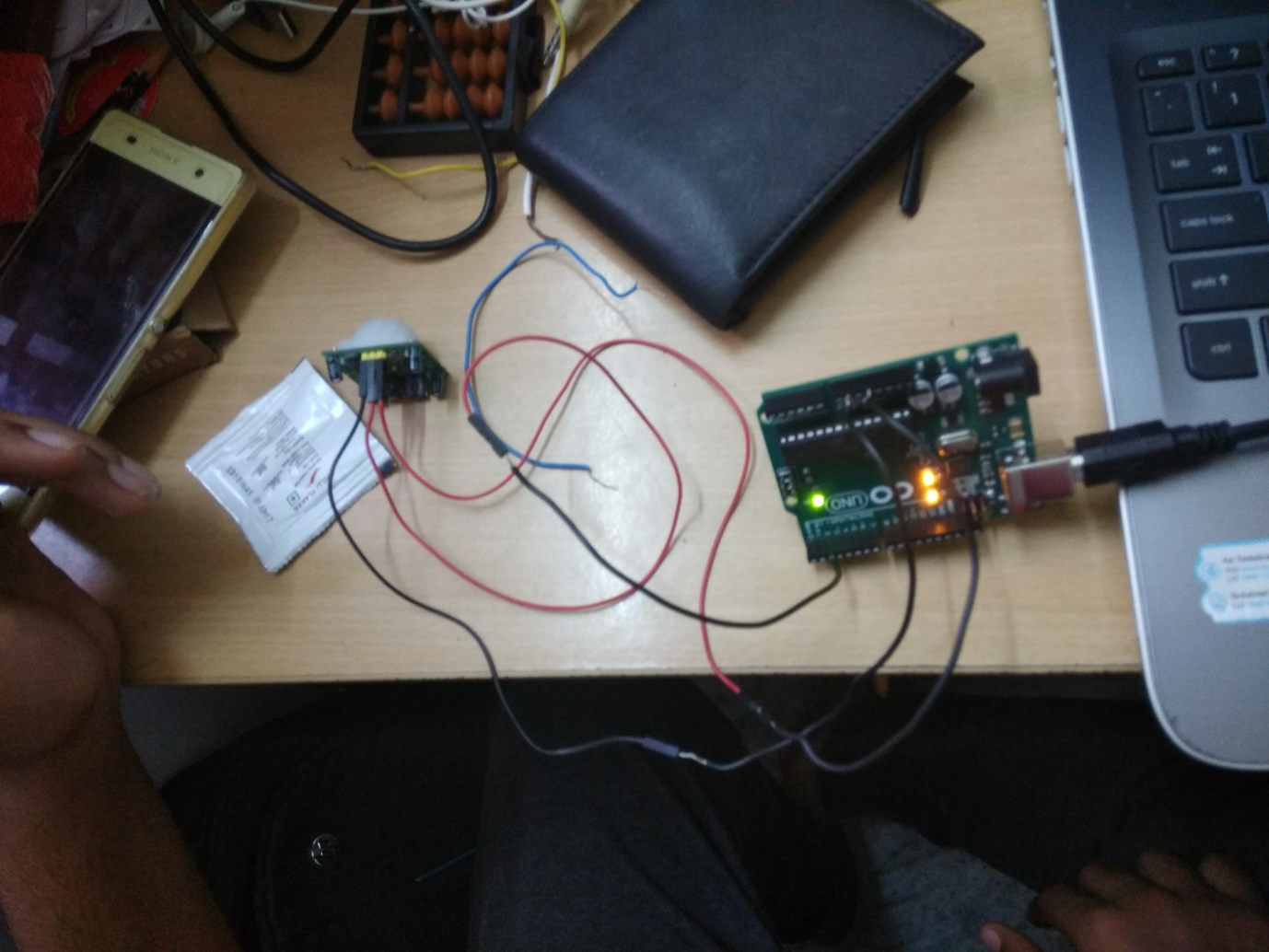
LDR :



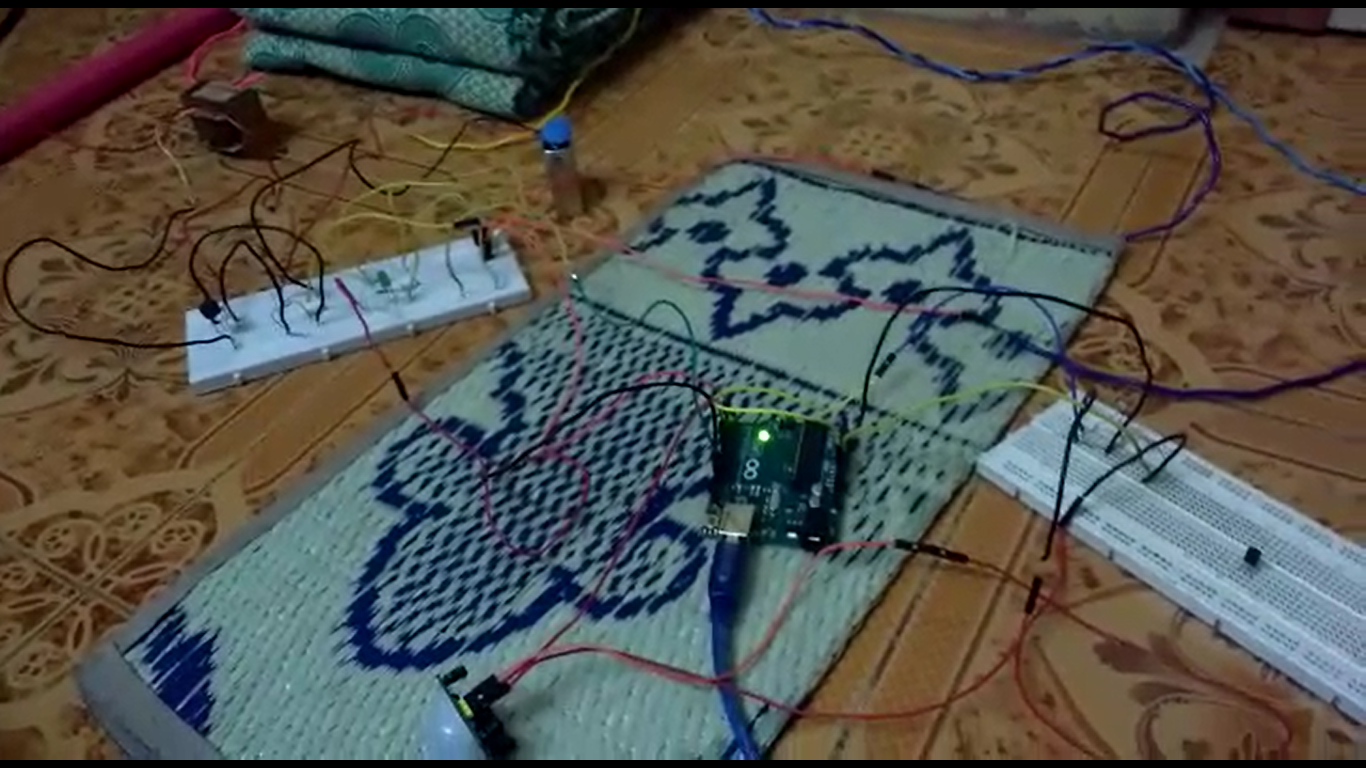
**LM-35:**

****

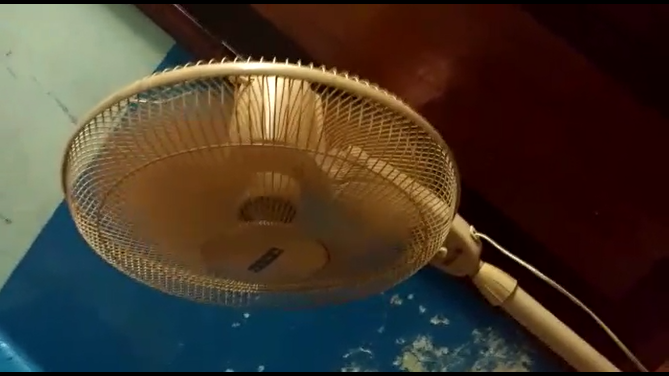
****

**PIR SENSOR:**

**FINAL OUTPUT:**

****

**Fan Light**

** **

**CONCLUSION:**

Thus the project is successfully implemented using PIR sensor and voltage to the electrical device is regulated according to the ambience,which was determined by LDR and LM35 sensors.

**REFERENCES:**

IEEE paper**“PIR-sensor-based Lighting Device with Ultra-low**

**Standby Power Consumption”** by-Cheng-Hung Tsai, Ying-Wen Bai, Chun-An Chu, Chih-Yu Chung and Ming-Bo Lin, Member, IEEE