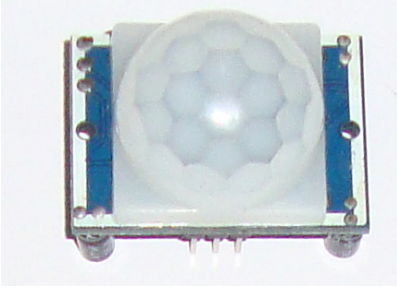
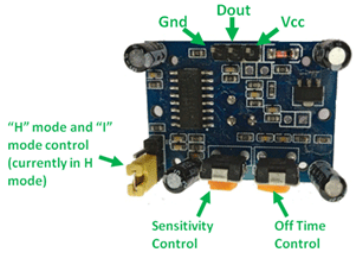
Lampiran

HC-SR501 PIR Sensor





Pin Configuration

Pin Number	Pin Name	Description
1	Vcc	Input voltage is +5V for typical applications. Can range from 4.5V- 12V
2	High/Low Ouput (Dout)	Digital pulse high (3.3V) when triggered (motion detected) digital low(0V) when idle(no motion detected
3	Ground	Connected to ground of circuit

PIR Sensor Features

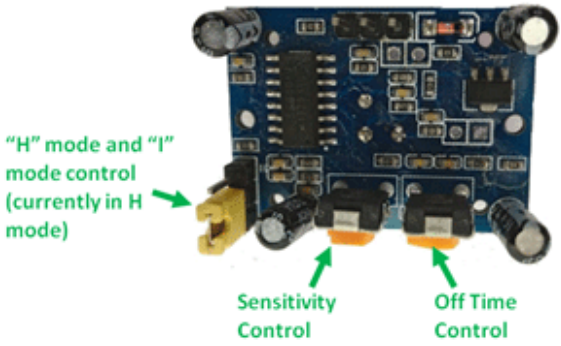
- Wide range on input voltage varying from 4.V to 12V (+5V recommended)
- Output voltage is High/Low (3.3V TTL)
- Can distinguish between object movement and human movement
- Has to operating modes - Repeatable(H) and Non-Repeatable(H)
- Cover distance of about 120° and 7 meters
- Low power consumption of 65mA
- Operating temperature from -20° to +80° Celsius.

How to use PIR Motion Sensor

The PIR sensor stands for Passive Infrared sensor. It is a low cost sensor which can detect the presence of Human beings or animals. This sensor has three output pins Vcc, Output and Ground as shown in the pin diagram above. Since the output pin is 3.3V TTL logic it can be used with any platforms like Arduino, Raspberry, PIC, ARM, 8051 etc..

The module can be powered from voltage 4.5V to 20V but, typically 5V is used. Once the module is powered allow the module to calibrate itself for few minutes, 2 minutes is a well settled time. Then observe the output on the output pin. Before we analyse the output we need to know that there are two operating modes in this sensor such as Repeatable(H) and Non- Repeatable(L) and mode. The Repeatable mode is the default mode.

The output of the sensor can be set by shorting any two pins on the left of the module as shown below. You can also notice two orange colour potentiometers that can be used to set the sensitivity and time which will be explained further below.

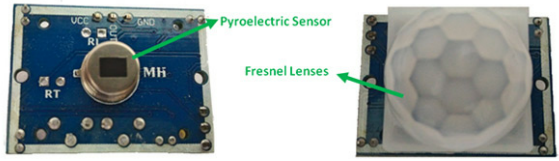


Repeatable(H) mode

In Repeatable(H) mode the output pin Dout will go high (3.3V) when a person is detected within range and goes low after a particular time (time is set by "Off time control" potentiometer). In this mode the output pin will go high irrespective of whether the person is still present inside the range or has left the area. The sensitivity can be set using the "sensitivity control" potentiometer

Non- Repeatable(L) mode

In "I" mode the output pin Dout will go high (3.3V) when a person is detected within range and will stay high as long as he/she stays within the limit of the Sensors range. Once the person has left the area the pin will go low after the particular time which can be set using the potentiometer. The sensitivity can be set using the "sensitivity control" potentiometer

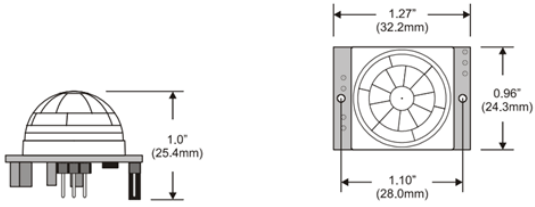


There are two important materials present in the sensor one is the pyroelectric crystal which can detect the heat signatures from a living organism (humans/animals) and the other is a Fresnel lenses which can widen the range of the sensor. Yes the white colour things is just a lense that is used to widen the range of the sensor, if you remove the lense you can find the Pyroelectric sensor inside it covered inside a protective metal casing as shown above.

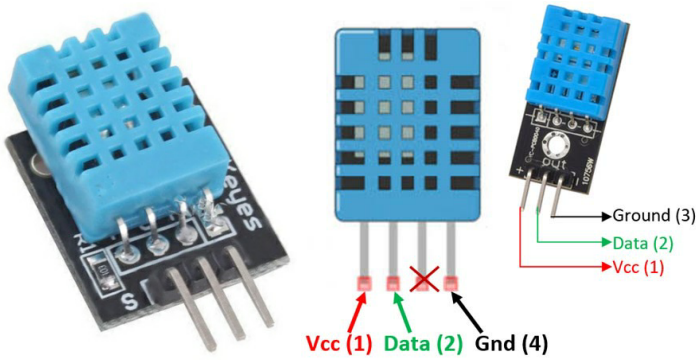
PIR Sensor Applications

- Automatic Street/Garage/Warehouse or Garden Lights
- Burglar Alarms
- Security cams as motion detectors
- Industrial Automation Control

2D model of the sensor



DHT11–Temperature and Humidity Sensor



Pin Identification and Configuration:

For DHT11 Sensor

1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	NC	No Connection and hence not used
4	Ground	Connected to the ground of the circuit

For DHT11 Sensor module

1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	Ground	Connected to the ground of the circuit

DHT11 Specifications:

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

Difference between DHT11 Sensor and module:

The **DHT11 sensor** can either be purchased as a sensor or as a module. Either way, the performance of the sensor is same. The sensor will come as a 4-pin package out of which only three pins will be used whereas the module will come with three pins as shown above.

The only difference between the sensor and module is that the module will have a filtering capacitor and pull-up resistor inbuilt, and for the sensor, you have to use them externally if

required.

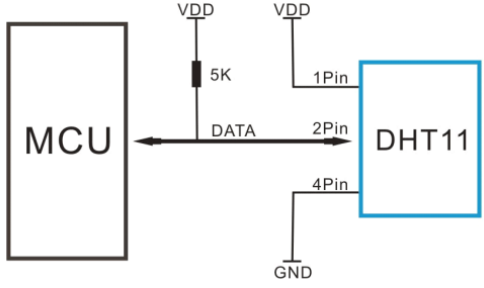
Where to use DHT11:

The **DHT11** is a commonly used **Temperature and humidity sensor.** The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. The sensor is also factory calibrated and hence easy to interface with other microcontrollers.

The sensor can measure temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of ±1°C and ±1%. So if you are looking to measure in this range then this sensor might be the right choice for you.

How to use DHT11 Sensor:

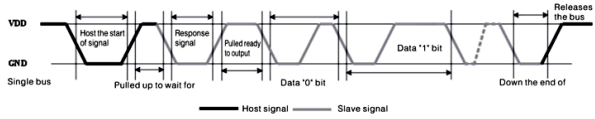
The DHT11 Sensor is factory calibrated and outputs serial data and hence it is highly easy to set it up. The connection diagram for this sensor is shown below.



As you can see the data pin is connected to an I/O pin of the MCU

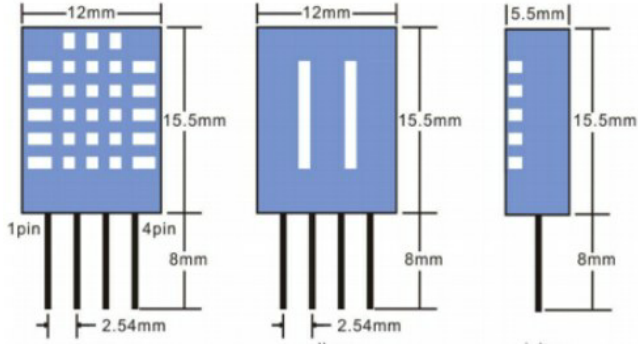
and a 5K pull-up resistor is used. This data pin outputs the value of both temperature and humidity as serial data. If you are trying to interface DHT11 with Arduino then there are ready-made libraries for it which will give you a quick start.

If you are trying to interface it with some other MCU then the datasheet given below will come in handy. The output given out by the data pin will be in the order of 8bit humidity integer data + 8bit the Humidity decimal data +8 bit temperature integer data + 8bit fractional temperature data +8 bit parity bit. To request the DHT11 module to send these data the I/O pin has to be momentarily made low and then held high as shown in the timing diagram below



The duration of each host signal is explained in the DHT11 datasheet, with neat steps and illustrative timing diagrams

2D–model of the sensor:



Coding program arduino

```
#include <LiquidCrystal_I2C.h>
#include <JC_Button.h>
#include <Rotary.h>
#include <SimpleTimer.h>
#include <IRremote.h>
#include <EEPROMex.h>
#include <avr/pgmspace.h>
#include "EmonLib.h"
#include "DHT.h"

#define RECV_PIN 10
IRrecv irrecv(RECV_PIN);
IRsend irsend;
decode_results results;

#define DHTPIN 3
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

#define PinCurrentAC    A0
#define PinCurrentLamp  A1
#define RelayAC        51
#define RelayLamp      53
#define RSTpinESP      49

EnergyMonitor SensorAC;
EnergyMonitor SensorLamp;

LiquidCrystal_I2C lcd(0x27, 20, 4);
Rotary pir = Rotary(18, 19);

SimpleTimer TimerFor_ChangeDisplay(5000);
SimpleTimer DelayFor_ShutdownAC(15000);

const PROGMEM unsigned int TurnON_AC[]  = {8850,4400,
550,550, 550,550, 550,600, 550,550, 550,550, 550,550, 550,550,
550,550, 550,1700, 550,1650, 550,1650, 550,1650, 550,1650,
550,1700, 550,1650, 550,1650, 550,550, 550,550, 550,550,
600,1650, 550,1650, 550,550, 600,500, 600,500, 550,1700,
550,1650, 550,1650, 550,550, 550,550, 550,1650, 600,1650,
500,1700, 550};
```

```
const PROGMEM unsigned int TurnOFF_AC[] = {8850,4450,
550,550, 550,550, 600,500, 600,500, 600,550, 550,550, 550,550,
600,500, 600,1600, 600,1600, 550,1700, 550,1650, 550,1650,
600,1600, 600,1600, 600,1650, 550,550, 550,550, 550,1650,
600,1600, 600,500, 600,550, 550,550, 550,550, 550,1650, 550,1650,
600,500, 600,550, 550,1650, 550,1650, 550,1650, 550,1650, 550};
const PROGMEM unsigned int temp16[] = {4500,4200, 700,1450,
650,450, 650,1450, 700,1450, 700,400, 650,400, 700,1450, 650,450,
650,400, 650,1500, 650,400, 700,400, 650,1500, 650,1500, 650,400,
650,1500, 650,1500, 650,400, 700,1450, 700,1450, 650,1500,
650,1450, 700,1450, 700,1450, 700,400, 650,1500, 650,400,
650,450, 650,400, 650,400, 700,400, 700,350, 700,1450, 700,1450,
700,400, 650,400, 650,450, 650,400, 650,450, 650,400, 650,450,
650,400, 700,1450, 700,1450, 650,1500, 650,1450, 700,1450,
700,1450, 700,5000, 4500,4200, 700,1450, 650,400, 700,1500,
650,1450, 650,450, 650,400, 650,1500, 700,350, 700,400, 700,1450,
650,400, 700,400, 650,1500, 650,1450, 700,400, 650,1500,
650,1500, 650,400, 650,1500, 650,1500, 650,1500, 650,1500,
650,1450, 650,1500, 700,350, 700,1500, 650,400, 650,450, 650,400,
650,450, 650,400, 650,400, 700,1450, 650,1500, 700,350, 700,400,
700,350, 700,400, 650,450, 650,400, 650,450, 650,400, 650,1500,
650,1450, 700,1450, 700,1450, 700,1450, 650,1500, 650};
const PROGMEM unsigned int temp17[] = {4450,4450, 550,1650,
600,1600, 600,1600, 600,550, 550,550, 600,500, 600,500, 600,500,
600,1650, 550,1650, 550,1650, 550,550, 600,500, 600,550, 550,550,
600,500, 600,500, 600,1650, 550,500, 600,550, 600,500, 600,500,
550,550, 600,550, 550,1650, 600,500, 550,1650, 550,1650,
600,1650, 600,1600, 550,1650, 600,1600, 600};
const PROGMEM unsigned int temp18[] = {4500,4250, 650,1450,
700,400, 650,1500, 600,1550, 650,400, 650,450, 650,1450, 650,450,
650,400, 700,1450, 700,350, 700,400, 700,1450, 650,1500, 650,400,
650,1500, 650,400, 700,1450, 700,1450, 650,1500, 650,1500,
650,400, 700,1450, 650,1500, 650,1500, 650,400, 700,400, 650,400,
700,400, 650,1500, 650,400, 700,400, 650,1500, 650,1450,
650,1500, 700,350, 700,400, 700,350, 700,400, 700,400, 650,400,
700,400, 650,400, 700,1450, 650,1500, 650,1450, 700,1500,
650,1450, 650,5050, 4500,4200, 700,1450, 650,450, 650,1450,
700,1450, 700,400, 650,400, 700,1450, 650,450, 650,400, 650,1500,
650,400, 700,400, 650,1450, 700,1500, 600,450, 650,1500, 650,400,
700,1450, 650,1500, 700,1450, 650,1500, 650,400, 650,1500,
650,1500, 650,1500, 650,400, 700,400, 650,400, 650,450, 650,1450,
```

700,400, 650,400, 700,1450, 700,1450, 700,1450, 650,450, 650,400,
650,400, 700,400, 650,400, 700,400, 650,400, 700,400, 650,1500,
650,1500, 650,1450, 700,1450, 650,1500, 700};
const PROGMEM unsigned int temp19[] = {8850,4400, 600,500,
600,500, 600,550, 550,550, 600,500, 550,550, 600,500, 600,500,
600,1650, 600,1600, 600,1600, 600,1600, 600,1600, 600,1650,
550,1650, 550,1650, 600,500, 600,1600, 600,500, 600,1650,
550,550, 550,550, 600,1600, 600,500, 600,1600, 600,550, 550,1650,
550,550, 600,1600, 600,1600, 600,550, 550,1650, 550};
const PROGMEM unsigned int temp20[] = {8900,4400, 550,550,
600,500, 600,500, 600,550, 550,550, 550,550, 550,550, 600,500,
600,1600, 600,1650, 600,1600, 550,1650, 550,1650, 600,1600,
600,1650, 550,1650, 550,550, 550,550, 600,500, 600,1600, 600,550,
550,550, 550,1650, 550,550, 600,1600, 600,1600, 600,1650,
600,500, 550,1650, 550,1650, 600,500, 600,1650, 550};
const PROGMEM unsigned int temp21[] = {8850,4400, 550,550,
600,500, 600,500, 600,550, 550,550, 550,550, 550,550, 600,500,
600,1600, 600,1650, 550,1650, 550,1650, 550,1650, 600,1600,
600,1650, 550,1650, 550,1650, 550,1650, 600,1600, 550,600,
550,550, 550,550, 550,1650, 600,500, 600,500, 600,550, 550,550,
550,1650, 550,1650, 600,1600, 550,550, 600,1650, 550};
const PROGMEM unsigned int temp22[] = {8900,4400, 550,550,
550,550, 600,500, 600,550, 550,550, 550,550, 550,550, 600,500,
600,1600, 600,1650, 550,1650, 550,1650, 550,1650, 600,1600,
600,1600, 600,1650, 600,1600, 550,1650, 600,500, 600,1600,
600,550, 550,550, 550,1650, 600,500, 600,500, 600,500, 600,1650,
550,550, 550,1650, 550,1650, 600,500, 550,1700, 550};
const PROGMEM unsigned int temp23[] = {4500,4250, 650,1450,
700,400, 650,1500, 600,1550, 650,400, 650,450, 650,1450, 650,450,
650,400, 700,1450, 700,350, 700,400, 700,1450, 650,1500, 650,400,
650,1500, 650,400, 700,1450, 700,1450, 650,1500, 650,1500,
650,400, 700,1450, 650,1500, 650,1500, 650,400, 700,400, 650,400,
700,400, 650,1500, 650,400, 700,400, 650,1500, 650,1450,
650,1500, 700,350, 700,400, 700,350, 700,400, 700,400, 650,400,
700,400, 650,400, 700,1450, 650,1500, 650,1450, 700,1500,
650,1450, 650,5050, 4500,4200, 700,1450, 650,450, 650,1450,
700,1450, 700,400, 650,400, 700,1450, 650,450, 650,400, 650,1500,
650,400, 700,400, 650,1450, 700,1500, 600,450, 650,1500, 650,400,
700,1450, 650,1500, 700,1450, 650,1500, 650,400, 650,1500,
650,1500, 650,1500, 650,400, 700,400, 650,400, 650,450, 650,1450,
700,400, 650,400, 700,1450, 700,1450, 700,1450, 650,450, 650,400,

650,400, 700,400, 650,400, 700,400, 650,400, 700,400, 650,1500,
650,1500, 650,1450, 700,1450, 650,1500, 700};
const PROGMEM unsigned int temp24[] = {4450,4250, 650,1450,
650,450, 650,1500, 650,1500, 650,400, 650,450, 650,1450, 650,450,
650,400, 650,1500, 650,450, 650,450, 600,1500, 650,1500, 650,400,
650,1500, 650,1500, 650,450, 650,1500, 650,1450, 650,1500,
650,1500, 650,1500, 650,1500, 650,400, 650,1500, 650,450,
600,450, 650,450, 600,450, 650,450, 600,450, 650,450, 600,450,
650,450, 600,450, 650,450, 650,400, 650,400, 650,450, 650,1500,
600,1550, 600,1550, 600,1500, 650,1500, 650,1500, 650,1500,
650,1500, 650,5050, 4500,4200, 600,1500, 650,450, 650,1500,
650,1500, 600,450, 650,450, 600,1550, 600,450, 650,400, 650,1500,
650,450, 650,400, 650,1500, 650,1500, 650,450, 600,1500,
650,1500, 650,400, 650,1500, 650,1500, 650,1500, 650,1500,
650,1500, 600,1550, 600,450, 650,1500, 650,400, 650,450, 650,400,
650,450, 650,400, 650,450, 650,400, 650,450, 600,450, 650,450,
600,450, 650,450, 650,400, 650,450, 650,1500, 650,1450, 650,1500,
650,1500, 650,1500, 650,1500, 650,1500, 600,1550, 600};
const PROGMEM unsigned int temp25[] = {4500,4200, 700,1450,
650,450, 650,1450, 700,1450, 700,400, 650,400, 700,1450, 650,450,
650,400, 650,1500, 650,400, 700,400, 650,1500, 650,1500, 650,400,
650,1500, 650,1500, 650,400, 700,1450, 700,1450, 650,1500,
650,1450, 700,1450, 700,1450, 700,400, 650,1500, 650,400,
650,450, 650,400, 650,400, 700,400, 700,350, 700,1450, 700,1450,
700,400, 650,400, 650,450, 650,400, 650,450, 650,400, 650,450,
650,400, 700,1450, 700,1450, 650,1500, 650,1450, 700,1450,
700,1450, 700,5000, 4500,4200, 700,1450, 650,400, 700,1500,
650,1450, 650,450, 650,400, 650,1500, 700,350, 700,400, 700,1450,
650,400, 700,400, 650,1500, 650,1450, 700,400, 650,1500,
650,1500, 650,400, 650,1500, 650,1500, 650,1500, 650,1500,
650,1450, 650,1500, 700,350, 700,1500, 650,400, 650,450, 650,400,
650,450, 650,400, 650,400, 700,1450, 650,1500, 700,350, 700,400,
700,350, 700,400, 650,450, 650,400, 650,450, 650,400, 650,1500,
650,1450, 700,1450, 700,1450, 700,1450, 650,1500, 650};
const PROGMEM unsigned int temp26[] = {4450,4450, 550,1650,
600,1600, 600,1600, 600,550, 550,550, 600,500, 600,500, 600,500,
600,1650, 550,1650, 550,1650, 550,550, 600,500, 600,550, 550,550,
600,500, 600,500, 600,1650, 550,500, 600,550, 600,500, 600,500,
550,550, 600,550, 550,1650, 600,500, 550,1650, 550,1650,
600,1650, 600,1600, 550,1650, 600,1600, 600};

```
const byte numChars = 128;
char receivedChars[numChars];
char tempChars[numChars];       // temporary array for use when
parsing
char DataFromEsp01[numChars] = {0};
boolean newData = false;
String str;

const byte
   PIN_UP(7),
   PIN_OK(6),
   PIN_DOWN(5);
Button BUTTON_UP(PIN_UP), BUTTON_OK(PIN_OK),
BUTTON_DOWN(PIN_DOWN);

const int
  addr_MinimumOrang = 5,
  numReadingsADC    = 50;

int
  MinimumOrang,
  JumlahOrang=0,
  khz = 38,
  humid = 0,
  LastValue_SetTempAC = 22,
  Perintah_SetTempAC = 22,
  Perintah_OnOff_AC = 1,
  Perintah_OnOff_Lamp = 1,
  LastPerintah_OnOff_AC = 1,
  LastPerintah_OnOff_Lamp = 1,
  LastStatusRelayAC = 1,
  LastStatusRelayLamp = 1;

unsigned int
  buffer[199];

unsigned long
  sendDataPrevMillis = 0;

float
  temp = 0.0,
```

```
  Current_AC,
  Current_LAMP;

byte
  valueMenu,
  ValueMenuDtaRemot,
  RsetDataFromServer,
  Button_Pressed = 0,
  Status_OnOff_AC = 1,
  Status_OnOff_Lamp = 1;

static bool
  RelayAC_State = true,
  RelayLamp_State = true,
  PerintahFormServer_AC = false,
  PerintahFormServer_Lamp = false;

boolean
  Button_LongPress = false,
  MenuSetting = false,
  SettingMinimumOrang = false,
  SettingRemot = false,
  RecordDtaIR = false,
  ChangeDisplay = true;

void setup()
{
  Serial.begin(9600);
  Serial2.begin(57600);

  SensorAC.current(PinCurrentAC, 0.95);       // Current: input pin,
calibration.
  SensorLamp.current(PinCurrentLamp, 0.95);

  BUTTON_UP.begin();
  BUTTON_OK.begin();
  BUTTON_DOWN.begin();

  irrecv.enableIRIn();
  dht.begin();
```

```
  lcd.begin();
  lcd.backlight();

  pir.begin();
  pinMode(18, INPUT_PULLUP);
  pinMode(19, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(19), Read_PIR_In, CHANGE);
  attachInterrupt(digitalPinToInterrupt(18), Read_PIR_Out,
CHANGE);

  pinMode(PinCurrentAC, INPUT);
  pinMode(PinCurrentLamp, INPUT);
  pinMode(RelayAC, OUTPUT);
  pinMode(RelayLamp, OUTPUT);
  digitalWrite(RelayAC, HIGH);
  digitalWrite(RelayLamp, HIGH);

  pinMode(4,OUTPUT); // power supply to sensor dht
  pinMode(2,OUTPUT);
  digitalWrite(4, HIGH);
  digitalWrite(2, LOW);

  pinMode(12,OUTPUT); // power supply to ir reciever
  pinMode(11,OUTPUT);
  digitalWrite(12, HIGH);
  digitalWrite(11, LOW);

  ResetESP();
  Read_Eeprom();
  delay(3000);
}

void loop()
{
  Read_SensorDHT();
  Read_CurrentSensor();
  Read_AllButton();
  read_AllLogic();

  if (millis() - sendDataPrevMillis > 500)
  {
```

```
    sendDataPrevMillis = millis();
    readIncomingData();
    delay(10);
    sendtoIoT();
  }

  if(MenuSetting == true || SettingMinimumOrang == true ||
SettingRemot == true){
    DisplayMenuSetting();
  }else{
    HomeDisplay();
  }
}

void read_AllLogic()
{
  if(JumlahOrang == 0 && RsetDataFromServer == 1)
  {
    RsetDataFromServer = 0;
    PerintahFormServer_AC = false;
    PerintahFormServer_Lamp = false;
  }

  if(JumlahOrang >= MinimumOrang)
  {
    if(digitalRead(RelayAC)==HIGH &&
PerintahFormServer_AC==false)
    {
      digitalWrite(RelayAC, LOW);
      RelayAC_State = false;
      Status_OnOff_AC = 0;
      RsetDataFromServer = 0;
    }
    if(digitalRead(RelayLamp)==HIGH &&
PerintahFormServer_Lamp==false)
    {
      digitalWrite(RelayLamp, LOW);
      RelayLamp_State = false;
      Status_OnOff_Lamp = 0;
      RsetDataFromServer = 0;
    }
```

```
  }else if(JumlahOrang == (MinimumOrang - 1)){
    if(digitalRead(RelayAC)==LOW)
    {
      RelayAC_State = true;
      Status_OnOff_AC = 1;
      DelayFor_ShutdownAC.reset();
    }
    if(digitalRead(RelayLamp)==LOW)
    {
      digitalWrite(RelayLamp, HIGH);
      RelayLamp_State = true;
      Status_OnOff_Lamp = 1;
    }
  }

  if(Perintah_SetTempAC != LastValue_SetTempAC &&
Status_OnOff_AC == 0 && Current_AC > 0.05)
  {
    LastValue_SetTempAC = Perintah_SetTempAC;
    switch(Perintah_SetTempAC)
    {
      case 16:
        sendCode(temp16, (sizeof(temp16)/sizeof(temp16[0])));
        break;
      case 17:
        sendCode(temp17, (sizeof(temp17)/sizeof(temp17[0])));
        break;
      case 18:
        sendCode(temp18, (sizeof(temp18)/sizeof(temp18[0])));
        break;
      case 19:
        sendCode(temp19, (sizeof(temp19)/sizeof(temp19[0])));
        break;
      case 20:
        sendCode(temp20, (sizeof(temp20)/sizeof(temp20[0])));
        break;
      case 21:
        sendCode(temp21, (sizeof(temp21)/sizeof(temp21[0])));
        break;
      case 22:
        sendCode(temp22, (sizeof(temp22)/sizeof(temp22[0])));
```

```
      break;
    case 23:
      sendCode(temp23, (sizeof(temp23)/sizeof(temp23[0])));
      break;
    case 24:
      sendCode(temp24, (sizeof(temp24)/sizeof(temp24[0])));
      break;
    case 25:
      sendCode(temp25, (sizeof(temp25)/sizeof(temp25[0])));
      break;
    case 26:
      sendCode(temp26, (sizeof(temp26)/sizeof(temp26[0])));
      break;
  }
 }

 if(Perintah_OnOff_AC != LastPerintah_OnOff_AC)
 {
  RsetDataFromServer = 1;
  LastPerintah_OnOff_AC = Perintah_OnOff_AC;

  if(PerintahFormServer_AC==false){
   PerintahFormServer_AC = true;
  }else{
   PerintahFormServer_AC = false;
  }

  if(RelayAC_State == false){
   RelayAC_State = true;
   Status_OnOff_AC = 1;
   DelayFor_ShutdownAC.reset();
   // Turn OFF AC
   sendCode(TurnOFF_AC,
(sizeof(TurnOFF_AC)/sizeof(TurnOFF_AC[0])));
  }else{
   RelayAC_State = false;
   Status_OnOff_AC = 0;
   digitalWrite(RelayAC, RelayAC_State);
  }

 }
```

```
  if(Perintah_OnOff_Lamp != LastPerintah_OnOff_Lamp)
  {
    RsetDataFromServer = 1;
    LastPerintah_OnOff_Lamp = Perintah_OnOff_Lamp;

    if(PerintahFormServer_Lamp==false){
      PerintahFormServer_Lamp = true;
    }else{
      PerintahFormServer_Lamp = false;
    }

    if(RelayLamp_State == false){
      RelayLamp_State = true;
      Status_OnOff_Lamp = 1;
    }else{
      RelayLamp_State = false;
      Status_OnOff_Lamp = 0;
    }
    digitalWrite(RelayLamp, RelayLamp_State);
  }

  if(Status_OnOff_AC == 0 && Current_AC <= 0.2)
  {
    // Turn ON AC
    delay(2500);
    sendCode(TurnON_AC,
(sizeof(TurnON_AC)/sizeof(TurnON_AC[0])));
  }

  if(Status_OnOff_AC == 1)
  {
    if(DelayFor_ShutdownAC.isReady()) {
      digitalWrite(RelayAC, RelayAC_State);  // Turn Off Relay AC
after turning off via remote and past timer delay shutdown
    }
  }
}

void storeCode(decode_results *results)
{
```

```
  // Start declaration
  Serial.print("unsigned int ");          // variable type
  Serial.print("rawData[");               // array name
  Serial.print(results->rawlen - 1, DEC);  // array size
  Serial.print("] = {");                  // Start declaration

  // Dump data
  for (int i = 1;  i < results->rawlen;  i++) {
    Serial.print(results->rawbuf[i] * USECPERTICK, DEC);
    if ( i < results->rawlen-1 ) Serial.print(","); // ',' not needed on last
one
    if (!(i & 1))  Serial.print(" ");
  }

  // End declaration
  Serial.print("};");
}

void sendCode(const unsigned int data[], int size)
{
  PGMtoIntArray(data, size);
  irsend.sendRaw(buffer, size, khz);
}

void PGMtoIntArray(const unsigned int data[], int size)
{
  for (byte i = 0; i < size; i++) {
    buffer[i]=pgm_read_word(&data[i]);
  }
}

void Read_AllButton()
{
  BUTTON_UP.read();
  BUTTON_OK.read();
  BUTTON_DOWN.read();

  if(BUTTON_OK.pressedFor(1000) && Button_LongPress==false){
    lcd.clear();
    Button_LongPress = true;
    Button_Pressed = 1;
```

```
   valueMenu = 1;
  }

  if(BUTTON_OK.wasReleased()){
   if(Button_LongPress==false && RecordDtaIR==false){
    if(SettingRemot==true){
     RecordDtaIR = true;
     MenuSetting = false;
     Button_Pressed = 1;
     if(ValueMenuDtaRemot==1){
      Serial.println("_____");
      Serial.println(" **** READY TO READ DATA IR FROM REMOTE
**** ");
      Serial.println("----------------------------------------");
      Serial.println("");
     }
    }

    if(valueMenu == 1 && Button_Pressed==0){
     Read_Eeprom();
     lcd.clear();
     SettingMinimumOrang = true;
     MenuSetting = false;
     valueMenu = 0;
    }

    if(valueMenu == 2){
     Read_Eeprom();
     lcd.clear();
     SettingRemot = true;
     MenuSetting = false;
     RecordDtaIR = false;
     valueMenu = 0;
     ValueMenuDtaRemot = 1;
    }

    if(valueMenu == 3){
     lcd.clear();
     SubSimpanData();
    }
   }else{
```

```
  Button_LongPress = false;
  delay(100);
 }
}

if(Button_LongPress==true){
  if(SettingMinimumOrang==true){
    MenuSetting = true;
    SettingMinimumOrang = false;
    valueMenu = 1;
  }else if(SettingRemot==true && RecordDtaIR==false){
    if(Button_Pressed==1){
      MenuSetting = true;
      SettingRemot = false;
      RecordDtaIR = false;
      valueMenu = 2;
    }
  }else if(RecordDtaIR==true && Button_Pressed==1){
    SettingRemot = true;
    RecordDtaIR = false;
    Button_Pressed = 0;
  }else if(SettingRemot==false && SettingMinimumOrang==false){
    MenuSetting = true;
    Button_Pressed = 0;
  }
}

if(BUTTON_UP.wasReleased()){
  lcd.clear();
  if(MenuSetting==true){
    valueMenu--;
    if(valueMenu < 1){
      valueMenu = 3;
    }
  }
  if(SettingMinimumOrang==true){
    MinimumOrang++;
    if(MinimumOrang > 30){
      MinimumOrang = 30;
    }
  }
```

```
  if(SettingRemot==true){

    ValueMenuDtaRemot++;
    if(ValueMenuDtaRemot > 14){
      ValueMenuDtaRemot = 14;
    }
  }
}

if(BUTTON_DOWN.wasReleased()){
  lcd.clear();
  if(MenuSetting==true){
    valueMenu++;
    if(valueMenu > 3){
      valueMenu = 1;
    }
  }
  if(SettingMinimumOrang==true){
    MinimumOrang--;
    if(MinimumOrang < 1){
      MinimumOrang = 1;
    }
  }
  if(SettingRemot==true){
    ValueMenuDtaRemot--;
    if(ValueMenuDtaRemot < 1){
      ValueMenuDtaRemot = 1;
    }
  }
}

if(MenuSetting==true){
switch(valueMenu)
  {
    case 1:
      lcd.setCursor(0,1);
      lcd.print(F(">>"));
      lcd.setCursor(18,1);
      lcd.print(F("<<"));
    break;
```

```
    case 2:
      lcd.setCursor(0,2);
      lcd.print(F(">>"));
      lcd.setCursor(18,2);
      lcd.print(F("<<"));
    break;

    case 3:
      lcd.setCursor(0,3);
      lcd.print(F(">>"));
      lcd.setCursor(18,3);
      lcd.print(F("<<"));
    break;
   }
  }

}

void sendtoIoT()
{
  str =
String("*Data_toServer,")+String(temp)+String("|")+String(humid)+
String("|")+String(JumlahOrang)+String("|")+String(Status_OnOff_A
C)+String("|")+String(Status_OnOff_Lamp)+String("|#");
  Serial2.println(str);
}

void HomeDisplay()
{

  if(TimerFor_ChangeDisplay.isReady()) {
    ChangeDisplay = !ChangeDisplay;
    TimerFor_ChangeDisplay.reset();
  }

  lcd.setCursor(0,0);
  lcd.print(F("LAMP>"));
  if(Status_OnOff_Lamp == 0 && Current_LAMP > 0.04){
    lcd.print(F("ON "));
  }else{
    lcd.print(F("OFF"));
```

```
}
lcd.print(F(",  "));
lcd.print(Current_LAMP);
lcd.print(F(" A "));

lcd.setCursor(0,1);
lcd.print(F("AC  >"));
if(Status_OnOff_AC == 0 && Current_AC > 0.04){
  lcd.print(F("ON "));
}else{
  lcd.print(F("OFF"));
}
if(ChangeDisplay){
  lcd.print(F(",Temp: "));
  lcd.print(Perintah_SetTempAC);
  lcd.print((char)223); //Simbol Derajat di LCD
  lcd.print(F("C "));
}else{
  lcd.print(F(",  "));
  lcd.print(Current_AC);
  lcd.print(F(" A  "));
}


lcd.setCursor(0,2);
lcd.print(F("Room> "));
if(ChangeDisplay){
  lcd.print(F("Temp: "));
  lcd.print(temp);
  lcd.print((char)223);
  lcd.print(F("C "));
}else{
  lcd.print(F("humidity: "));
  lcd.print(humid);
  lcd.print(F("% "));
}

lcd.setCursor(0,3);
lcd.print(F("People in Room: "));
lcd.print(JumlahOrang); /* HASIL PERHITUNGAN SENSOR PIR */
lcd.print(F(" "));
```

```
}

void DisplayMenuSetting()
{
  if(MenuSetting == true && SettingMinimumOrang == false &&
SettingRemot == false){
    lcd.setCursor(0,0);
    lcd.print(F("==== MAIN  MENU ===="));
    lcd.setCursor(2,1);
    lcd.print(F(" MINIMUM  ORANG "));
    lcd.setCursor(2,2);
    lcd.print(F("   DATA  REMOT  "));
    lcd.setCursor(2,3);
    lcd.print(F("  SIMPAN  DATA  "));
  }

  if(MenuSetting == false && SettingMinimumOrang == true &&
SettingRemot == false){
    lcd.setCursor(0,0);
    lcd.print(F("JUMLAH MINIMUM ORANG"));
    lcd.setCursor(9,1);
    lcd.print(MinimumOrang);
  }

  if(MenuSetting == false && SettingMinimumOrang == false &&
SettingRemot == true){
    lcd.setCursor(0,0);
    if(RecordDtaIR==true){
      if(ValueMenuDtaRemot==1){
        lcd.print(F("=INPUT DATA  REMOTE="));
        lcd.setCursor(0,1);
        lcd.print(F(" IR Receiver Ready, "));
        lcd.setCursor(0,2);
        lcd.print(F(" printed on serial  "));
        lcd.setCursor(0,3);
        lcd.print(F("     USB Arduino    "));
      }else{
        lcd.print(F("=TEST  DATA  REMOTE="));
        lcd.setCursor(0,3);
        lcd.print(F(" Press OK, For Test "));
      }
```

```
  }else{
    lcd.print(F("=== DATA  REMOTE ==="));
  }

  lcd.setCursor(0,1);
  switch(ValueMenuDtaRemot)
   {
     case 1:
      if(RecordDtaIR==true){
       decode_results  results;

       if(irrecv.decode(&results)) {
         storeCode(&results);
         Serial.println("");
         irrecv.resume();       // Prepare for the next value
       }
      }else{
       lcd.print(F(" Press OK For Start "));
       lcd.setCursor(0,2);
       lcd.print(F("  Read IR Data From "));
       lcd.setCursor(0,3);
       lcd.print(F("       Remote       "));
      }
      break;

     case 2:
      lcd.print(F(" FOR TURN ON        "));
      if(BUTTON_OK.wasReleased()){
       sendCode(TurnON_AC,
(sizeof(TurnON_AC)/sizeof(TurnON_AC[0])));
      }
      break;

     case 3:
      lcd.print(F(" FOR TURN OFF       "));
      if(BUTTON_OK.wasReleased()){
       sendCode(TurnOFF_AC,
(sizeof(TurnOFF_AC)/sizeof(TurnOFF_AC[0])));
      }
     break;
```

```
case 4:
  lcd.print(F(" FOR SET 16 degrees "));
  if(BUTTON_OK.wasReleased()){
    sendCode(temp16, (sizeof(temp16)/sizeof(temp16[0])));
  }
break;

case 5:
  lcd.print(F(" FOR SET 17 degrees "));
  if(BUTTON_OK.wasReleased()){
    sendCode(temp17, (sizeof(temp17)/sizeof(temp17[0])));
  }
break;

case 6:
  lcd.print(F(" FOR SET 18 degrees "));
  if(BUTTON_OK.wasReleased()){
    sendCode(temp18, (sizeof(temp18)/sizeof(temp18[0])));
  }
break;

case 7:
  lcd.print(F(" FOR SET 19 degrees "));
  if(BUTTON_OK.wasReleased()){
    sendCode(temp19, (sizeof(temp19)/sizeof(temp19[0])));
  }
break;

case 8:
  lcd.print(F(" FOR SET 20 degrees "));
  if(BUTTON_OK.wasReleased()){
    sendCode(temp20, (sizeof(temp20)/sizeof(temp20[0])));
  }
break;

case 9:
  lcd.print(F(" FOR SET 21 degrees "));
  if(BUTTON_OK.wasReleased()){
    sendCode(temp21, (sizeof(temp21)/sizeof(temp21[0])));
  }
break;
```

```
      case 10:
        lcd.print(F(" FOR SET 22 degrees "));
        if(BUTTON_OK.wasReleased()){
          sendCode(temp22, (sizeof(temp22)/sizeof(temp22[0])));
        }
      break;

      case 11:
        lcd.print(F(" FOR SET 23 degrees "));
        if(BUTTON_OK.wasReleased()){
          sendCode(temp23, (sizeof(temp23)/sizeof(temp23[0])));
        }
      break;

      case 12:
        lcd.print(F(" FOR SET 24 degrees "));
        if(BUTTON_OK.wasReleased()){
          sendCode(temp24, (sizeof(temp24)/sizeof(temp24[0])));
        }
      break;

      case 13:
        lcd.print(F(" FOR SET 25 degrees "));
        if(BUTTON_OK.wasReleased()){
          sendCode(temp25, (sizeof(temp25)/sizeof(temp25[0])));
        }
      break;

      case 14:
        lcd.print(F(" FOR SET 26 degrees "));
        if(BUTTON_OK.wasReleased()){
          sendCode(temp26, (sizeof(temp26)/sizeof(temp26[0])));
        }
      break;
    }
  }
}

void Read_SensorDHT()
{
```

```
 humid = dht.readHumidity();
 temp = dht.readTemperature();
}

void Read_CurrentSensor()
{

  Current_AC = SensorAC.calcIrms(285); // Calculate Irms only,
285 times
  Current_LAMP = SensorLamp.calcIrms(285);
}

void readIncomingData()
{
   static boolean recvInProgress = false;
   static byte ndx = 0;
   char startMarker = '*';
   char endMarker = '#';
   char rc;

   while (Serial2.available() > 0 && newData == false) {
      rc = Serial2.read();

      if (recvInProgress == true) {
         if (rc != endMarker) {
            receivedChars[ndx] = rc;
            ndx++;
            if (ndx >= numChars) {
               ndx = numChars - 1;
            }
         }
         else {
            receivedChars[ndx] = '\0'; // terminate the string
            recvInProgress = false;
            ndx = 0;
            newData = true;
         }
      }

      else if (rc == startMarker) {
         recvInProgress = true;
```

```
      }
    }

  if (newData == true) {
    strcpy(tempChars, receivedChars);
    parseData();
    newData = false;
  }
}

void parseData()
{
    // split the data into its parts
    char * strtokIndx; // this is used by strtok() as an index

    strtokIndx = strtok(tempChars,",");      // get the first part - the
string
    strcpy(DataFromEsp01, strtokIndx); // copy it to
messageFromPC

    strtokIndx = strtok(NULL, "|");
    Perintah_SetTempAC = atof(strtokIndx);     // convert this part

    strtokIndx = strtok(NULL, "|");
    Perintah_OnOff_AC = atof(strtokIndx);     // convert this part

    strtokIndx = strtok(NULL, "|");
    Perintah_OnOff_Lamp = atof(strtokIndx);     // convert this part
}

void SubSimpanData()
{
  lcd.setCursor(0,0);
  lcd.print(F("MENYIMPAN SEMUA DATA"));
  delay(150);
  lcd.setCursor(0,1);
  lcd.print(F("**                "));
  delay(150);
  lcd.setCursor(0,1);
  lcd.print(F("****               "));
  delay(300);
```

```
  lcd.setCursor(0,1);
  lcd.print(F("*******             "));
  delay(400);
  lcd.setCursor(0,1);
  lcd.print(F("***********         "));
  delay(70);
  lcd.setCursor(0,1);
  lcd.print(F("***************     "));
  delay(50);
  lcd.setCursor(0,1);
  lcd.print(F("******************  "));
  delay(600);
  lcd.setCursor(0,1);
  lcd.print(F("********************"));
  delay(800);
  lcd.clear();
  lcd.setCursor(0,1);
  lcd.print(F("   DATA TERSIMPAN   "));
  delay(1500);
  lcd.clear();
  MenuSetting = false;
  valueMenu = 0;
  EEPROM.writeInt(addr_MinimumOrang,MinimumOrang);
}

void Read_Eeprom()
{
  MinimumOrang = EEPROM.readInt(addr_MinimumOrang);
}

void Read_PIR_In()
{
  int x = digitalRead(19);
  byte n;
  if(x == 1){
    n = 1;
  }

  if(x == 0 && n == 1){
    n = 0;
    JumlahOrang = JumlahOrang + 1;
```

```
  }
}

void Read_PIR_Out()
{
  int xx = digitalRead(18);
  byte nn;
  if(xx == 1){
    nn = 1;
  }

  if(xx == 0 && nn == 1){
    nn = 0;
    if (JumlahOrang >= 1){
      JumlahOrang = JumlahOrang -1;
    }
  }
}

void ResetESP()
{
  pinMode(RSTpinESP, OUTPUT);
  digitalWrite(RSTpinESP, LOW);
  delay(100);
  digitalWrite(RSTpinESP, HIGH);
}
```

Coding program esp 8266

```
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
#include <SoftwareSerial.h>
SoftwareSerial SerialESP01(0, 2); // RX, TX

#define WIFI_SSID "vivo1606"
#define WIFI_PASSWORD "12345678"

#define FIREBASE_HOST "ac-internetofthink-default-
rtdb.firebaseio.com"
#define FIREBASE_AUTH
"iZqIFjlDxbV0NUvFwqa7XOoGmtO7JPR9dnTvzLdi"
```

```
FirebaseData fbdo;

int
  JumlahOrang,
  humid,
  intPerintah_SetTempAC,
  intPerintah_OnOff_AC,
  intPerintah_OnOff_Lamp,
  Status_OnOff_AC,
  Status_OnOff_Lamp,
  Perintah_OnOff_AC = 1,
  Perintah_OnOff_Lamp = 1,
  LastPerintah_OnOff_AC = 1,
  LastPerintah_OnOff_Lamp = 1,
  String_len;

float temp;

String
  stringPerintah_SetTempAC,
  stringPerintah_OnOff_AC,
  stringPerintah_OnOff_Lamp,
  inString = "",
  str;

unsigned long sendDataPrevMillis = 0;

const byte numChars = 128;
char receivedChars[numChars];
char tempChars[numChars];        // temporary array for use when
parsing
char DataFromUno[numChars] = {0};
boolean newData = false;

void setup() {
  SerialESP01.begin(57600);
  Serial.begin(9600);

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
```

```
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);

  //Set the size of WiFi rx/tx buffers in the case where we want to
work with large data.
  fbdo.setBSSLBufferSize(1024, 1024);

  //Set the size of HTTP response buffers in the case where we
want to work with large data.
  fbdo.setResponseSize(1024);

  if (Firebase.setInt(fbdo, "Perintah_SetTempAC", 22)){
    Serial.println("Perintah_OnOff_AC Begin PASSED");
  }
  delay(10);
  if (Firebase.setInt(fbdo, "Perintah_OnOff_AC", 1)){
    Serial.println("SetTempAC Begin PASSED");
  }
  delay(10);
  if (Firebase.setInt(fbdo, "Perintah_OnOff_Lamp", 1)){
    Serial.println("Perintah_OnOff_Lamp Begin PASSED");
  }
}

void SendToServer(){
   // set value
    if (Firebase.setFloat(fbdo, "temperature", temp))
    {
      Serial.println("temperature PASSED");
    }else{
      Serial.println("temperature FAILED");
```

```
  }
  delay(1);
  if (Firebase.setInt(fbdo, "humidity", humid))
  {
   Serial.println("humidity PASSED");
  }else{
   Serial.println("humidity FAILED");
  }
  delay(1);
  if (Firebase.setInt(fbdo, "Jumlah_Orang", JumlahOrang))
  {
   Serial.println("Jumlah_Orang PASSED");
  }else{
   Serial.println("Jumlah_Orang FAILED");
  }
  delay(1);
  if (Status_OnOff_AC <= 1){
   if(Firebase.setInt(fbdo, "Status_OnOff_AC", Status_OnOff_AC))
   {
    Serial.println("Status_OnOff_AC PASSED");
   }else{
    Serial.println("Status_OnOff_AC FAILED");
   }
  }
  delay(1);
  if (Status_OnOff_Lamp <= 1){
   if(Firebase.setInt(fbdo, "Status_OnOff_Lamp",
Status_OnOff_Lamp))
   {
    Serial.println("Status_OnOff_Lamp PASSED");
   }else{
    Serial.println("Status_OnOff_Lamp FAILED");
   }
  }
/*
  delay(1);
  if(Perintah_OnOff_AC != LastPerintah_OnOff_AC){
   LastPerintah_OnOff_AC = Perintah_OnOff_AC;
   if (Perintah_OnOff_AC <= 1){
    if(Firebase.setInt(fbdo, "Perintah_OnOff_AC",
Perintah_OnOff_AC))
```

```
      {
        Serial.println("Perintah_OnOff_AC PASSED");
      }else{
        Serial.println("Perintah_OnOff_AC FAILED");
      }
     }
   }
   delay(1);
   if(Perintah_OnOff_Lamp != LastPerintah_OnOff_Lamp){
    LastPerintah_OnOff_Lamp = Perintah_OnOff_Lamp;
    if (Perintah_OnOff_Lamp <= 1){
     if(Firebase.setInt(fbdo, "Perintah_OnOff_Lamp",
Perintah_OnOff_Lamp))
      {
        Serial.println("Perintah_OnOff_Lamp PASSED");
      }else{
        Serial.println("Perintah_OnOff_Lamp FAILED");
      }
     }
   } */
}

// ********** Wait Data For Send To IoT ********* //
void readIncomingData() {
   static boolean recvInProgress = false;
   static byte ndx = 0;
   char startMarker = '*';
   char endMarker = '#';
   char rc;

   while (SerialESP01.available() > 0 && newData == false) {
      rc = SerialESP01.read();

      if (recvInProgress == true) {
         if (rc != endMarker) {
            receivedChars[ndx] = rc;
            ndx++;
            if (ndx >= numChars) {
               ndx = numChars - 1;
            }
         }
```

```
            else {
               receivedChars[ndx] = '\0'; // terminate the string
               recvInProgress = false;
               ndx = 0;
               newData = true;
            }
         }

         else if (rc == startMarker) {
            recvInProgress = true;
         }
      }

   if (newData == true) {
      strcpy(tempChars, receivedChars);
      parseData();
      readFromServer();
      delay(10);
      SendToServer();
      newData = false;
   }
}

//============
void parseData() {      // split the data into its parts

    char * strtokIndx; // this is used by strtok() as an index

    strtokIndx = strtok(tempChars,",");      // get the first part - the
string
    strcpy(DataFromUno, strtokIndx); // copy it to messageFromPC

    strtokIndx = strtok(NULL, "|");
    temp = atof(strtokIndx);     // convert this part

    strtokIndx = strtok(NULL, "|");
    humid = atof(strtokIndx);     // convert this part

    strtokIndx = strtok(NULL, "|");
    JumlahOrang = atof(strtokIndx);     // convert this part
```

```
  strtokIndx = strtok(NULL, "|");
  Status_OnOff_AC = atof(strtokIndx);     // convert this part

  strtokIndx = strtok(NULL, "|");
  Status_OnOff_Lamp = atof(strtokIndx);     // convert this part
/*
  strtokIndx = strtok(NULL, "|");
  Perintah_OnOff_AC = atof(strtokIndx);     // convert this part

  strtokIndx = strtok(NULL, "|");
  Perintah_OnOff_Lamp = atof(strtokIndx);     // convert this part
  */
}

void readFromServer() {

 if (Firebase.get(fbdo, "/Perintah_SetTempAC")) {
  if (fbdo.dataType() == "string")
  {
   stringPerintah_SetTempAC = fbdo.stringData();

   String_len = stringPerintah_SetTempAC.length()+1;
   for (int i=0; i<String_len; i++)
   {
    char inChar = stringPerintah_SetTempAC[i];
    if (isDigit(inChar))
    {
     inString += inChar;
    }
   }
   intPerintah_SetTempAC = inString.toInt();
   inString = "";
   String_len = 0;
  }else{
   intPerintah_SetTempAC = fbdo.intData();
  }
 }

 if (Firebase.get(fbdo, "/Perintah_OnOff_AC")) {
  if (fbdo.dataType() == "string")
  {
```

```
    stringPerintah_OnOff_AC = fbdo.stringData();

    String_len = stringPerintah_OnOff_AC.length()+1;
     for (int i=0; i<String_len; i++)
     {
       char inChar = stringPerintah_OnOff_AC[i];
       if (isDigit(inChar))
       {
         inString += inChar;
       }
     }
     intPerintah_OnOff_AC = inString.toInt();
     inString = "";
     String_len = 0;
   }else{
     intPerintah_OnOff_AC = fbdo.intData();
   }
 }

 if (Firebase.get(fbdo, "/Perintah_OnOff_Lamp")) {
   if (fbdo.dataType() == "string")
   {
     stringPerintah_OnOff_Lamp = fbdo.stringData();

     String_len = stringPerintah_OnOff_Lamp.length()+1;
     for (int i=0; i<String_len; i++)
     {
       char inChar = stringPerintah_OnOff_Lamp[i];
       if (isDigit(inChar))
       {
         inString += inChar;
       }
     }
     intPerintah_OnOff_Lamp = inString.toInt();
     inString = "";
     String_len = 0;
   }else{
     intPerintah_OnOff_Lamp = fbdo.intData();
   }
 }
```

```
  str =
String("*Data_from_server,")+String(intPerintah_SetTempAC)+Strin
g("|")+String(intPerintah_OnOff_AC)+String("|")+String(intPerintah_
OnOff_Lamp)+String("|#");
  Serial.println(str);
  SerialESP01.println(str);
}

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