**INTRODUCTION**

Home automation is anything that enables you to use your home’s lighting, heating and appliances more conveniently and efficiently.

It can be as simple as remote or automatic control of a few lights, or it can be a complete system that controls all major parts of your home, custom set to your own personal preference. Home automation is anything that gives you remote or automatic control of things around the home.

The aim of this **Getting Started** guide is to give you an unbiased view of what’s available and help you make an informed choice of what home automation technologies best fit your needs and budget.

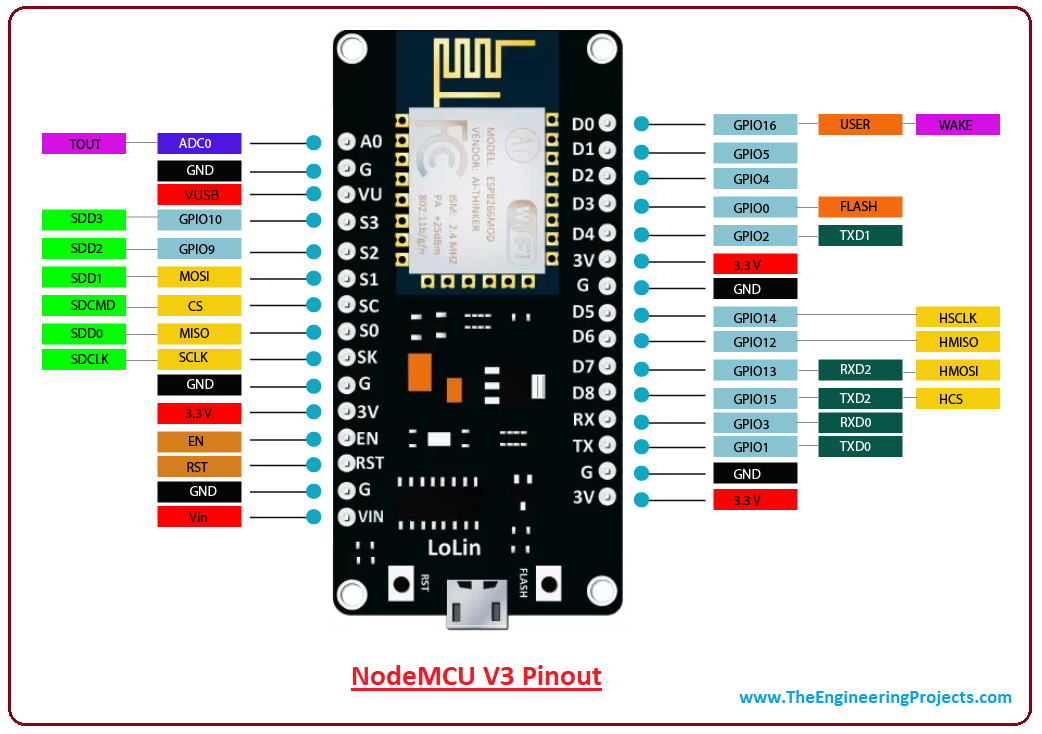
**Description**

The goal of this project is to develop a home automation system that gives the user complete control over all remotely controllable aspects of his or her home.

The automation system will have the ability to be controlled from all over the world through the Internet, and also remotely accessed via a Pocket PC with a Windows Mobile based application.

The System will also sense the Accidental Gas leakage , water level and will notify the user by SMS.

**CONTENT’S**

**ESP 8266 Wifi Module**

The ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor.

Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, you can simply hook this up to your Arduino device and get about as much WiFi-ability as a WiFi Shield offers (and that’s just out of the box)! The ESP8266 module is an extremely cost effective board with a huge, and ever growing, community.

This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its PIOs with minimal development up-front and minimal loading during runtime. Its high degree of on-chip integration allows for minimal external circuitry, including the front-end module, is designed to occupy minimal PCB area. The ESP8266 supports APSD for VoIP applications and Bluetooth co-existence interfaces, it contains a self-calibrated RF allowing

it to work under all operating conditions, and requires no external RF parts. There is an almost limitless fountain of information available for the ESP8266, all of which has been provided by amazing community support. In the Documents section below you will find many resources to aid you in using the ESP8266, even instructions on how to transforming this module into an IoT (Internet of Things) solution!

**Features:**

 802.11 b/g/n

 Wi-Fi Direct (P2P), soft-AP

 Integrated TCP/IP protocol stack

 Integrated TR switch, balun, LNA, power amplifier and matching network

 Integrated PLLs, regulators, DCXO and power management units

 +19.5dBm output power in 802.11b mode

 Power down leakage current of <10uA

 1MB Flash Memory

 Integrated low power 32-bit CPU could be used as application processor

 SDIO 1.1 / 2.0, SPI, UART

 STBC, 1×1 MIMO, 2×1 MIMO

 A-MPDU & A-MSDU aggregation & 0.4ms guard interval

 Wake up and transmit packets in < 2ms

 Standby power consumption of < 1.0mW (DTIM3)

**Specification of ESP 8266:**

 Wi-Fi Direct (P2P), soft-AP

 Integrated TCP/IP protocol stack

 Integrated TR switch, balun, LNA, power amplifier and matching network

 Integrated PLLs, regulators, DCXO and power management units

 19.5dBm output power in 802.11b mode

 Power down leakage current of <10uA

 1MB Flash Memory

 Integrated low power 32-bit CPU could be used as application processor

 Standby power consumption of < 1.0mW (DTIM3)

**Relay module 4 channel**

****

This relay module allows you to combine the processing power of the Arduino to devices that use higher current and voltage.  It does so by providing four relays that are rated for 7A at either 28VDC or 10A at 125VAC.

Each relay has a Normally Open (NO) and a Normally Closed (NC) contact.

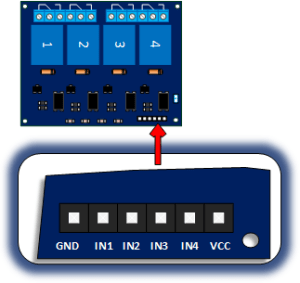
With these relays you can control:

**Appliances**

* Motors
* Lights
* Other Relay

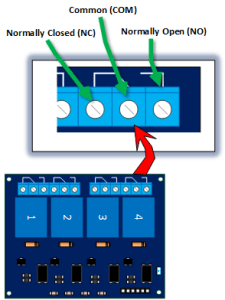
**Relay Module Inputs**

* The module is supplied with power via the pin labeled VCC and ground via the pin labeled GND.
* The relays are energized with low inputs to the IN1, IN2, IN3 and IN4 inputs

[](https://i0.wp.com/henrysbench.capnfatz.com/wp-content/uploads/2015/04/Four-Channel-Relay-Inputs.png)

**Relay Module Outputs.**

* There are four relays that each provide dry contact outputs.  That is to say that each relay provides a common (COM) , normally open (NO) and a normally closed (NC) terminal.

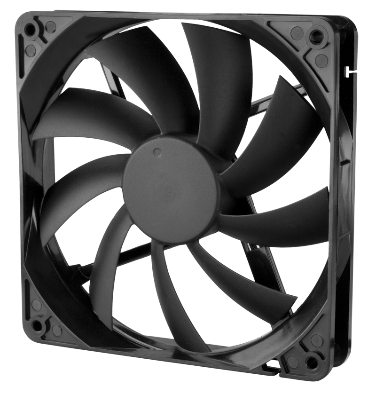
[](https://i2.wp.com/henrysbench.capnfatz.com/wp-content/uploads/2015/04/Four-Channel-Relay-Outputs.png)

****

**PIR sensor**

A passive infrared **sensor** (**PIR sensor**) is an electronic **sensor** that measures infrared (IR) light radiating from objects in its field of view. They are most often used in **PIR**-based **motion** detectors.

**FAN Cooler’s**

****A **computer fan** is any [fan](https://en.wikipedia.org/wiki/Fan_(machine)) inside, or attached to, a [computer case](https://en.wikipedia.org/wiki/Computer_case) used for [active cooling](https://en.wikipedia.org/wiki/Active_cooling). Fans are used to draw cooler air into the case from the outside, expel warm air from inside, and move air across a [heat sink](https://en.wikipedia.org/wiki/Heat_sink) to cool a particular component. Both [axial](https://en.wikipedia.org/wiki/Axial_fan) and sometimes [centrifugal](https://en.wikipedia.org/wiki/Centrifugal_fan) (blower/squirrel-cage) fans are used in computers. Computer fans commonly come in standard sizes, and are powered and controlled using 3- or 4-pin fan [connectors](https://en.wikipedia.org/wiki/Electrical_connector).

** Jumper wires**

Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with [breadboards](https://blog.sparkfuneducation.com/what-is-a-breadboard) and other prototyping tools in order to make it easy to change a circuit as needed. Fairly simple. In fact, it doesn’t get much more basic than jumper wires.

**Led 12v**

A **light-emitting diode** (**LED**) is a [semiconductor](https://en.wikipedia.org/wiki/Semiconductor) [light source](https://en.wikipedia.org/wiki/Light_source) that emits light when [current](https://en.wikipedia.org/wiki/Electric_current) flows through it. [Electrons](https://en.wikipedia.org/wiki/Electron) in the semiconductor recombine with [electron holes](https://en.wikipedia.org/wiki/Electron_hole), releasing energy in the form of [photons](https://en.wikipedia.org/wiki/Photon). This effect is called [electroluminescence](https://en.wikipedia.org/wiki/Electroluminescence).[[5]](https://en.wikipedia.org/wiki/Light-emitting_diode#cite_note-5) The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the [band gap](https://en.wikipedia.org/wiki/Band_gap) of the semiconductor.[[6]](https://en.wikipedia.org/wiki/Light-emitting_diode#cite_note-6) White light is obtained by using multiple semiconductors or a layer of light-emitting phosphor on the semiconductor device

**SCOPE OF THE PROJECT**

Day by day, the field of automation is blooming and these systems are having great impact on human beings. The project which is to be implemented is a home automation using Easy IOT Webserver and WIFI and has very good future development.

In the current system webserver is installed on a windows PC so the home appliances can be controlled using only by using the device on which webserver is installed. This can be further developed installing webserver on cloud.

Advantage of installing webserver on the cloud is that home can be controlled by using any device which has WIFI 802.1 and a web browser. By visiting the IP address of the cloud the control actions can be taken.

**CODING**

//Google Assistant Home Automation

#include <ESP8266WiFi.h>

#include "Adafruit\_MQTT.h"

#include "Adafruit\_MQTT\_Client.h"

#include <Servo.h>

Servo servo;

Servo ser;

int ledPin = D0; // choose the pin for the LED (Pin D5, GPIO 14) I tried using pin D4 but had problems for some reason

int inputPin = D6; // choose the input pin (for PIR sensor) (Pin D6, GPIO 12)

int pirState = LOW; // we start, assuming no motion detected

int val = 0; // variable for reading the pin status

#define SERVO\_PIN 15

#define SERVO\_piont 14

#define Relay1 D1

#define Relay2 D2

#define Relay3 D3

#define Relay4 D4

#define Relay5 D8

#define Relay7 D5

//char ssid[] = "Admin Wifi"; //Enter your WIFI Name

//char pass[] = "03152010409";

#define WLAN\_SSID "Admin Wifi" // Your SSID

#define WLAN\_PASS "03152010409" // Your password

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Adafruit.io Setup \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define AIO\_SERVER "io.adafruit.com" //Adafruit Server

#define AIO\_SERVERPORT 1883

#define AIO\_USERNAME "Ali\_jaffri" // Username

#define AIO\_KEY "2709846a49b443aba52bf43c69bed308" // Auth Key

//WIFI CLIENT

WiFiClient client;

Adafruit\_MQTT\_Client mqtt(&client, AIO\_SERVER, AIO\_SERVERPORT, AIO\_USERNAME, AIO\_KEY);

Adafruit\_MQTT\_Subscribe Light1 = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/Relay1"); // Feeds name should be same everywhere

Adafruit\_MQTT\_Subscribe Light2 = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/Relay2");

Adafruit\_MQTT\_Subscribe Light3 = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/Relay3");

Adafruit\_MQTT\_Subscribe Light4 = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/Relay4");

Adafruit\_MQTT\_Subscribe Light5 = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/Relay5");

Adafruit\_MQTT\_Subscribe Light7 = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/Relay7");

const char motion\_FEED[] PROGMEM = AIO\_USERNAME "/feeds/Relay6";

Adafruit\_MQTT\_Publish motion = Adafruit\_MQTT\_Publish(&mqtt, motion\_FEED);

void MQTT\_connect();

void setup() {

Serial.begin(115200);

pinMode(Relay1, OUTPUT);

pinMode(Relay2, OUTPUT);

pinMode(Relay3, OUTPUT);

pinMode(Relay4, OUTPUT);

pinMode(Relay5, OUTPUT);

pinMode(Relay7, OUTPUT);

pinMode(D7,OUTPUT);

pinMode(ledPin, OUTPUT); // declare LED for motion sensor as output

pinMode(inputPin, INPUT);

// Connect to WiFi access point.

Serial.println(); Serial.println();

Serial.print("Connecting to ");

Serial.println(WLAN\_SSID);

WiFi.begin(WLAN\_SSID, WLAN\_PASS);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println();

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

mqtt.subscribe(&Light1);

mqtt.subscribe(&Light3);

mqtt.subscribe(&Light2);

mqtt.subscribe(&Light4);

mqtt.subscribe(&Light5);

mqtt.subscribe(&Light7);

servo.attach(SERVO\_PIN);

ser.attach(14);

}

void loop() {

MQTT\_connect();

Adafruit\_MQTT\_Subscribe \*subscription;

while ((subscription = mqtt.readSubscription(20000))) {

if (subscription == &Light1) {

Serial.print(F("Got: "));

Serial.println((char \*)Light1.lastread);

int Light1\_State = atoi((char \*)Light1.lastread);

digitalWrite(Relay1, Light1\_State);

}

if (subscription == &Light2) {

Serial.print(F("Got: "));

Serial.println((char \*)Light2.lastread);

int Light2\_State = atoi((char \*)Light2.lastread);

digitalWrite(Relay2, Light2\_State);

}

if (subscription == &Light3) {

Serial.print(F("Got: "));

Serial.println((char \*)Light3.lastread);

int Light3\_State = atoi((char \*)Light3.lastread);

digitalWrite(Relay3, Light3\_State);

}

if (subscription == &Light4) {

Serial.print(F("Got: "));

Serial.println((char \*)Light4.lastread);

int Light4\_State = atoi((char \*)Light4.lastread);

digitalWrite(Relay4, Light4\_State);

}//////////////

if (subscription == &Light5) {

Serial.print(F("Got: "));

Serial.println((char \*)Light5.lastread);

int Light5\_State = atoi((char \*)Light5.lastread);

servo.write(Light5\_State);

if(Light5\_State==10){

digitalWrite(D7,LOW);

}else{

digitalWrite(D7,HIGH);

}

}

if (subscription == &Light7) {

Serial.print(F("Got: "));

Serial.println((char \*)Light7.lastread);

int Light7\_State = atoi((char \*)Light7.lastread);

ser.write(Light7\_State);

}

/////////////

disPlay();

}

}

void disPlay(){

val = digitalRead(inputPin); // read input value of PIR motion sensor

if (val == HIGH) // check if the input is HIGH

{

digitalWrite(ledPin,LOW); // Turn the LED ON

if (pirState == LOW)

{

Serial.println("Motion detected!"); // we have just turned on

pirState = HIGH; // We only want to print on the output change, not state

if (! motion.publish(val)) // Publish to Adafruit the PIR sensor value '1'

{

Serial.println(F("Failed")); // If it failed to publish, print Failed

} else

{

Serial.println(F("Data Sent!")); // If data successfully published

}

}

} else

{

digitalWrite(ledPin, HIGH); // Turn LED OFF

if (pirState == HIGH) // we have just turned off

{

Serial.println("Motion ended!"); // We only want to print on the output change, not state

pirState = LOW;

motion.publish(val); // Publish to Adafruit the PIR sensor value '0'

}

}

}

void MQTT\_connect() {

int8\_t ret;

if (mqtt.connected()) {

return;

}

Serial.print("Connecting to MQTT... ");

uint8\_t retries = 3;

while ((ret = mqtt.connect()) != 0) {

Serial.println(mqtt.connectErrorString(ret));

Serial.println("Retrying MQTT connection in 5 seconds...");

mqtt.disconnect();

delay(5000);

retries--;

if (retries == 0) {

while (1);

}

}

Serial.println("MQTT Connected!");

}