



Helwan University  
Faculty of Engineering

# ***SECURE AND SAFE SMART HOME SYSTEM***

***2018-2019***

***PROF. DR. EL SAYED MOSTAFA SAAD***

***PREPARED BY***

***Mahmoud Khalid Abdelkaream***

***Mohamed Khaled Ahmed***

***Abdelrahman Mohamed Yassin***

***Marwaa Abdelall Mohamed***

***Amira Atef Hussien***

## Table of content:

1.Abstraction.....	3
2.Automation and safety tools.....	4
• Design of project.....	5
• Circuit component.....	6
-Blynk.....	6
-NodeMCU v3.....	7
-sensors.....	11
1.Gas sensor.....	11
2.DHT11-temperature and humidity sensor.....	14
3.soil moisture sensor.....	18
4.motion sensor.....	19
-Relays.....	21
-Code of project.....	26
• Home Automation using NodeMCU and google assistant.....	29
3.RFID and keypad Based Access control System using Arduino.....	36
- overview.....	36
- Hardware Components.....	36
- Story.....	37
- RFID Basics and Module Interfacing With Arduino.....	37
1-RFID Basics.....	37
2-RFID Tapes.....	38
3-RFID Reader.....	38
4-HOW RFID Works Arduino.....	39
5-RFID Arduino Interfacing.....	39
-Working of RFID and keypad Based Access Control System Using Arduino.....	42
-SMS part .....	44

-HOW It Works.....	44
-Code Of Project.....	46
-Security for Intelligent, Connected IoT Edge Nodes.....	49
-Tips to secure those IoT devices.....	50
-Conclusion and future work.....	52
-References.....	53

#### Table of Figures

Fig1 design of the project.....	5
Fig1 Blynk layout.....	6
Fig1 NodeMCU V3 Pinout.....	8
Fig1 Gas Sensor (MQ2) module.....	11
Fig1 DHT11–Temperature and Humidity Sensor.....	14
Fig1 Motion sensor.....	20
Fig1 Google Assistant .....	32
Fig1 complete circuit diagram.....	44

## 1.Abstraction



The first thing you desire when you look at your family and your home is their safety. And thus the idea of advanced home security system comes into picture. The concept of home automation and its safety has been around since late 1970s. But during the course of time with the advancement of technology, our expectation from home has changed a lot and so have the idea of home automation and its security systems. If we look at different home automation systems overtime, they have always tried to provide efficient, convenient and safe ways for home inhabitants to access their homes.

Our work focuses mainly on the security and safety aspect of home automation :

- In automation subsystem we use voice app. in controlling system for helping disabled people. Providing some level of security for home doors and windows with more than one method as RFID, Password, Notification(i.e SMS) for home owner.

- Safety for full system as detecting gas leakage, fire smoking, increase in temperature,...

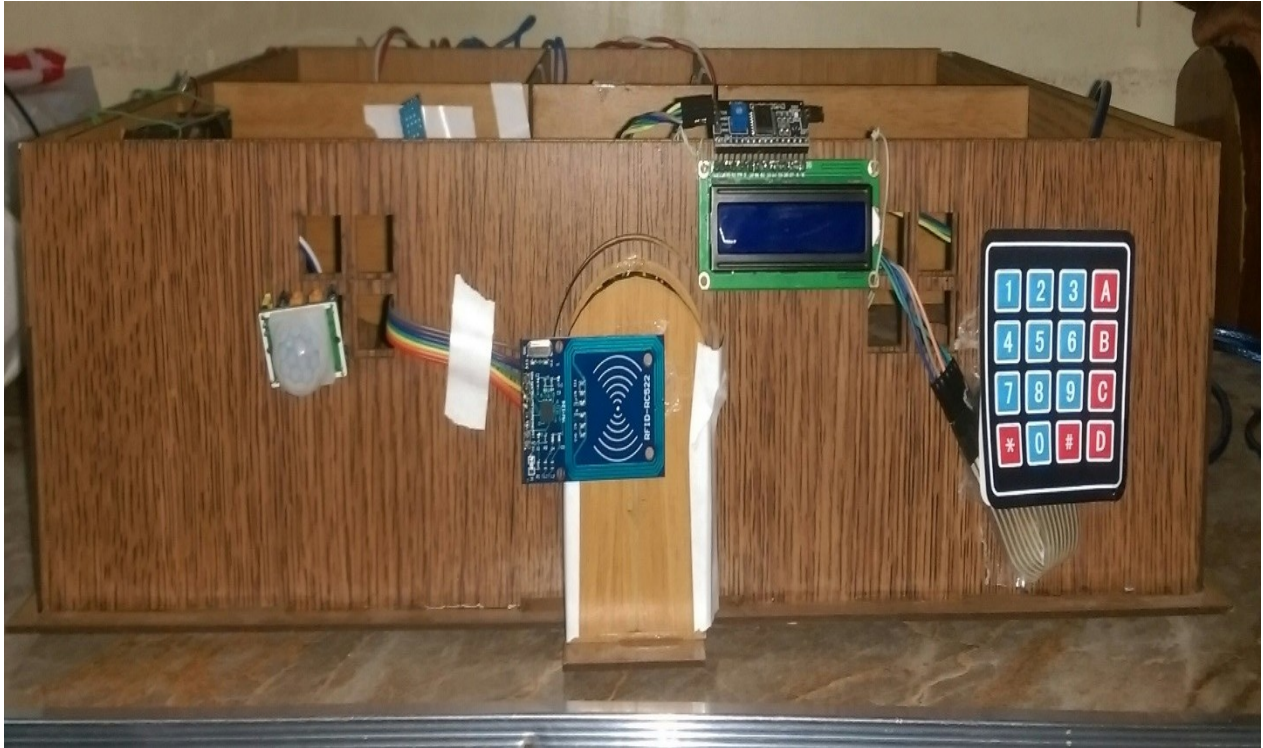
Including detecting faults and correction them.

## 2.Automation and Safety Tools



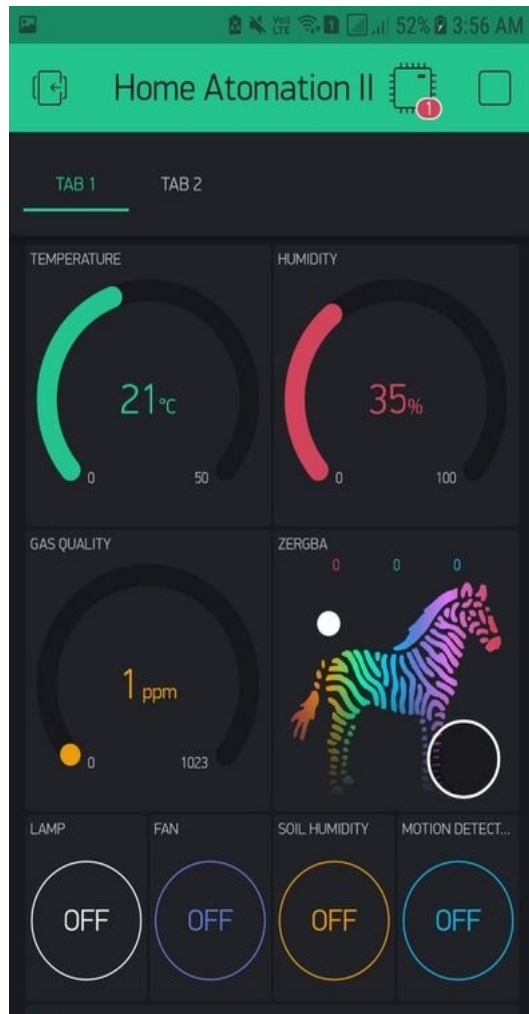






### ***Circuit component:***

***Blynk*** is a platform with iOS and Android apps to control Arduino, Raspberry Pi and the likes over the Internet. It's a digital dashboard where you can build a graphic interface for your project by simply dragging and dropping widgets. It's really simple to set everything up and you'll start tinkering in less than 5 mins. Blynk is not tied to some specific board or shield. Instead, it's supporting hardware of your choice. Whether your Arduino or Raspberry Pi is linked to the Internet over Wi-Fi, Ethernet or this new ESP8266 chip, Blynk will get you online and ready for the Internet of Your Things.



### Characteristics of Blynk are:

Similar API & UI for all supported hardware & devices  
 Connection to the cloud can be done using Ethernet, Wi-Fi, Bluetooth, BLE and USB (Serial)  
 Set of easy-to-use Widgets  
 Direct pin manipulation with no code writing  
 Easy to integrate and add new functionality using virtual pins  
 History data monitoring via History Graph widget  
 Device-to-Device communication using Bridge Widget  
 Sending emails, tweets, push notifications, etc.

The best way to develop quickly an IoT application with less Integrated circuits to add is to choose this circuit "Node MCU". Today, we will give a detailed Introduction on Node MCU V3. It is an opensource firmware and development kit that plays a vital role in



designing a proper IoT product using a few script lines. The module is mainly based on ESP8266 that is a low-cost Wi-Fi microchip incorporating both a full TCP/IP stack and microcontroller capability.

It is introduced by manufacturer Espressif Systems. The ESP8266 Node MCU is a complex device, which combines some features of the ordinary Arduino board with the possibility of connecting to the internet.

### ***NodeMCU V3:***

NodeMCU V3 is an open-source firmware and development kit that plays a vital role in designing an IoT product using a few script lines.

Multiple GPIO pins on the board allow us to connect the board with other peripherals and are capable of generating PWM, I2C, SPI, and UART serial communications. The interface of the module is mainly divided into two parts including both Firmware and Hardware where former runs on the ESP8266 Wi-Fi SoC and later is based on the ESP-12 module.

And open source firmware gives you the flexibility to edit, modify and rebuilt the existing module and keep changing the entire interface until you succeed in optimizing the module as per your requirements.

USB to UART converter is added on the module that helps in converting USB data to UART data which mainly understands the language of serial communication.

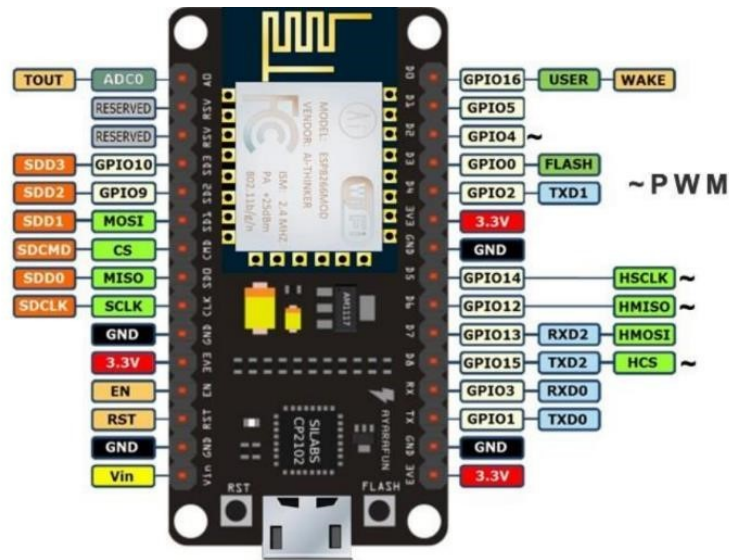
Instead of the regular USB port, MicroUSB port is included in the module that connects it with the computer for dual purposes: programming and powering up the board.

The board incorporates status LED that blinks and turns off immediately, giving you the current status of the module if it is running properly when connected with the computer.

The ability of module to establish a flawless WiFi connection between two channels makes it an ideal choice for incorporating it with other embedded devices like Raspberry Pi.

### **NodeMCU V3 Pinout**


NodeMCU V3 comes with a number of GPIO Pins. Following figure shows the Pinout of the board.



There is a candid difference between Vin and VU where former is the regulated voltage that may stand somewhere between 7 to 12 V while later is the power voltage for USB that must be kept around 5 V.

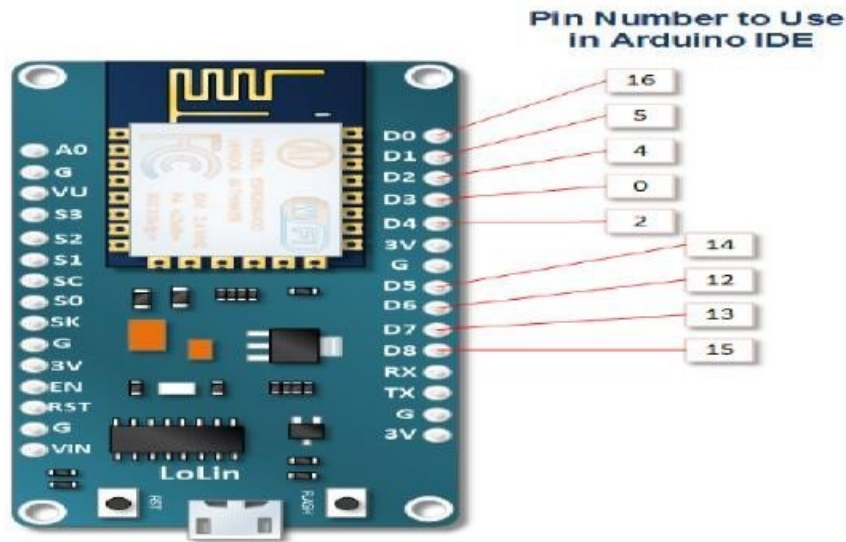
## Features

1. Open-source
2. Arduino-like hardware
3. Status LED
4. MicroUSB port
5. Reset/Flash buttons
6. Interactive and Programmable
7. Low cost
8. ESP8266 with inbuilt wifi

- 
9. USB to UART converter
  10. GPIO pins
  11. Arduino-like hardware IO
  12. Advanced API for hardware IO, which can dramatically reduce the redundant work for configuring and manipulating hardware.
  13. Code like arduino, but interactively in Lua script.
  14. Nodejs style network API
  15. Event-driven API for network applicaitons, which faciliates developers writing code running on a 5mm\*5mm sized MCU in Nodejs style.
  16. Greatly speed up your IOT application developing process.
  17. Lowest cost WI-FI
  18. Less than \$2 WI-FI MCU ESP8266 integrated and esay to prototyping development kit.
  19. We provide the best platform for IOT application development at the lowest cost

As mentioned above, a cable supporting micro USB port is used to connect the board. As you connect the board with a computer, LED will flash. You may need some drivers to be installed on your computer if it fails to detect the NodeMCU board.

**Note:** We use Arduino IDE software for programming this module. It is important to note that the pin configuration appearing on the board is different from the configuration we use to program the board on the software i.e. when we write code for targeting pin 16 on the Arduino IDE, it will actually help is laying out the communication with the D0 pin on the module. Following figure the shows the pin configuration to use in Arduino IDE.



As you connect the board with a computer, LED will flash. You may need some drivers to be installed on your computer if it fails to detect the NodeMCU board. You can download the driver from this page.

**Note:** We use Arduino IDE software for programming this module. It is important to note that the pin configuration appearing on the board is different from the configuration we use to program the board on the software i.e. when we write code for targeting pin 16 on the Arduino IDE, it will help in laying out the communication with the D0 pin on the module.

### How to Power NodeMCU V3

We can see from the pinout image above, there are five ground pins and three 3V3 pins on the board. The board can be powered up using the following three ways USB Power. It proves to be an ideal choice for loading programs unless the project you aim to design requires separate interface i.e. disconnected from the computer.

**Provide 3.3V.** This is another great option to power up the module. If you have your own off-board regulator, you can generate an instant power source for your development kit.

**Power Vin.** This is a voltage regulator that comes with the ability to support up to 800 mA. It can handle somewhere between 7 to 12 V. You cannot power the devices operating at 3.3 V, as this regulator is unable to generate as low as 3.3V.

# Sensors

## 1-The Grove - Gas Sensor (MQ2) module

It is useful for gas leakage detection (home and industry). It is suitable for detecting H<sub>2</sub>, LPG, CH<sub>4</sub>, CO, Alcohol, Smoke or Propane. Due to its high sensitivity and fast response time, measurement can be taken as soon as possible. The sensitivity of the sensor can be adjusted by potentiometer.



### Features

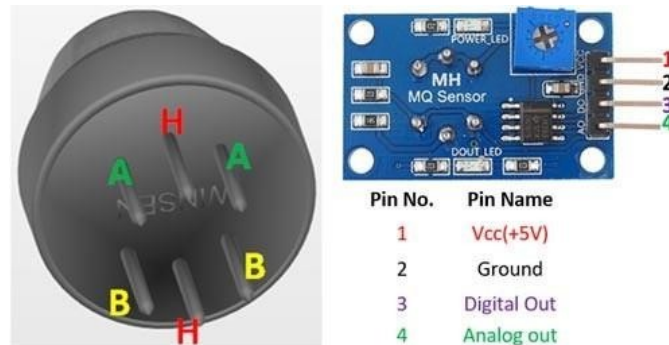
- Wide detecting scope
- Stable and long lifetime
- Fast response and High sensitivity
- High sensitivity to Combustible gas in wide range
- Stable performance, long life, low cost
- Simple drive circuit

### Application Ideas

- Gas leakage detection. □ Toys.

### How it is working?

The output voltage from the Gas sensor increases when the concentration of gas increases. Sensitivity can be adjusted by rotating the potentiometer. For detail information about the MQ-2 sensor, please refer the data-sheet provided in **Resources** section



### Where to use MQ-2 Gas sensor:

The **MQ-2 Gas sensor** can detect or measure gasses like LPG, Alcohol, Propane, Hydrogen, CO and even methane. The module version of this sensor comes with a Digital Pin which makes this sensor to operate even without a microcontroller and that comes in handy when you are only trying to detect one particular gas. When it comes to measuring the gas in ppm the analog pin has to be used, the analog pin also TTL driven and works on 5V and hence can be used with most common microcontrollers.

So, if you are looking for a sensor to detect or measure gasses like LPG, Alcohol, Propane, Hydrogen, CO and even methane with or without a microcontroller then this sensor might be the right choice for you.

### How to use MQ-2 Sensors to detect gas:

Using an MQ sensor it detects a gas is very easy. You can either use the digital pin or the analog pin to accomplish this.

Simply power the module with 5V and you should notice the power LED on the module to glow and when no gas it detected the output LED will remain turned off meaning the digital output pin will be 0V. Remember that these sensors have to be kept on for pre-heating time (mentioned in features above) before you can actually work with it. Now,

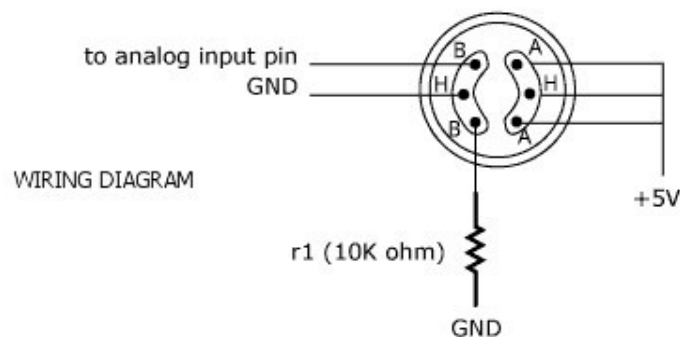


introduce the sensor to the gas you want to detect, and you should see the output LED to go high along with the digital pin, if not use the potentiometer until the output gets high. Now every time your sensor gets introduced to this gas at this particular concentration the digital pin will go high (5V) else will remain low (0V).

You can also use the analog pin to achieve the same thing. Read the analog values (0-5V) using a microcontroller, this value will be directly proportional to the concentration of the gas to which the sensor detects. You can experiment with this value and check how the sensor reacts to different concentration of gas and develop your program accordingly.

### How to use the MQ-2 sensor to measure PPM:

If you are looking for some accuracy with your readings, then measuring the PPM would be the best way to go with it. It can also help you to distinguish one gas from another. So, to measure PPM you can directly use a module. A basic wiring for the sensor from datasheet is shown below.



### Applications:

- Detects or measure Gases like LPG, Alcohol, Propane, Hydrogen, CO and even methane
- Air quality monitor
- Gas leak alarm
- Safety standard maintenance
- Maintaining environment standards in hospitals

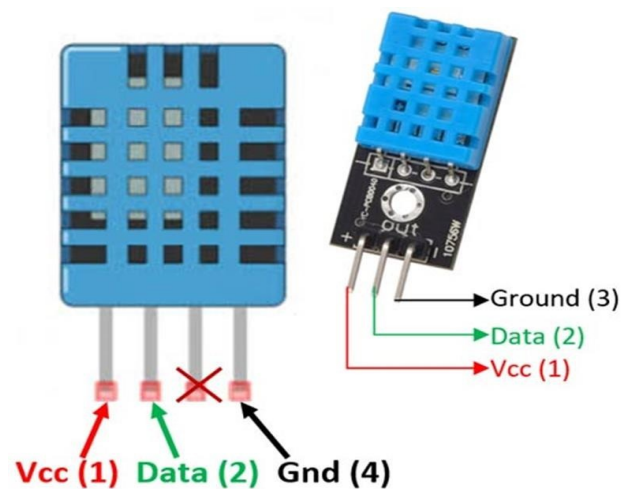
### The various types of gas sensors

Symbol	MQ-2	MQ-3	MQ-5	MQ-9
Detect Gas	Combustible Gas, Smoke	Alcohol Vapor	LPG, Natural Gas, Town Gas	Carbon Monoxide, Coal Gas, Liquefied Gas
Detect Concentration	300-10000ppm	0.04-4mg/L Alcohol	300-10000ppm	10-1000ppm CO; 100-10000PPm Gas

## 2. DHT11–Temperature and Humidity Sensor:

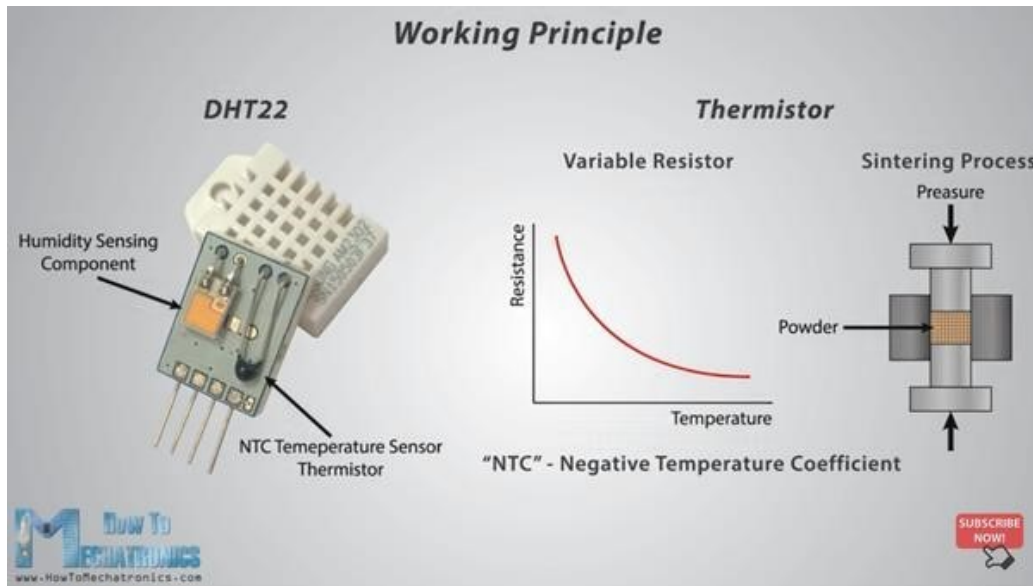
DHT11 is a single wire digital humidity and temperature sensor, which gives relative humidity in percentage and temperature in degree Celsius.

DHT11 is a single wire digital humidity and temperature sensor



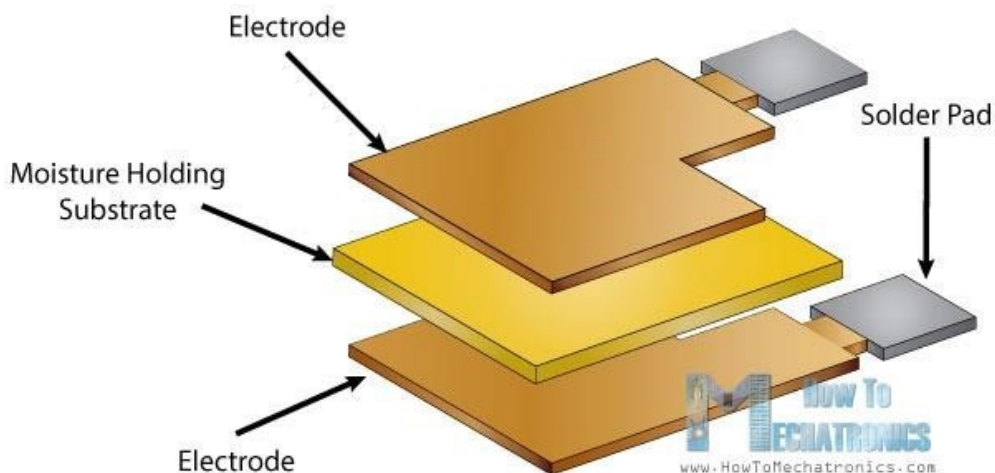
DHT11 Sensor Pinout

Dht11 Working Principle



### For measuring humidity (use capacitor)

- For measuring humidity, they use the humidity sensing component which has two electrodes with moisture holding substrate between them. So as the humidity changes, the conductivity of the substrate changes or the resistance between these electrodes' changes. This change in resistance is measured and processed by the IC which makes it ready to be read by a microcontroller.

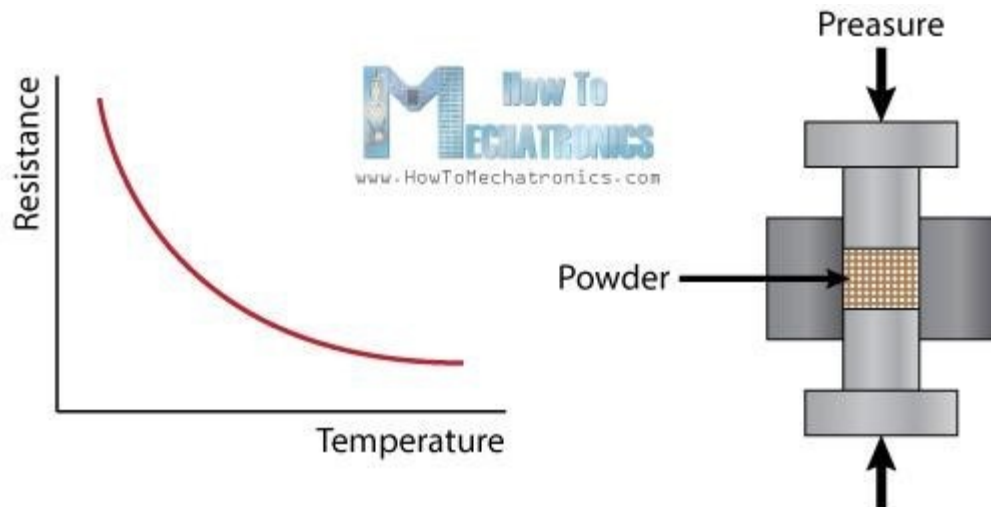


### For measuring temperature (use resistor)

- for measuring temperature these sensors use a NTC temperature sensor or a thermistor.

A thermistor is actually a variable resistor that changes its resistance with change of the temperature. These sensors are made by sintering of semiconductor materials such as ceramics or polymers in order to provide larger changes in the resistance with just small changes in temperature. The term

“NTC” means “Negative Temperature Coefficient”, which means that the resistance decreases with increase of the temperature

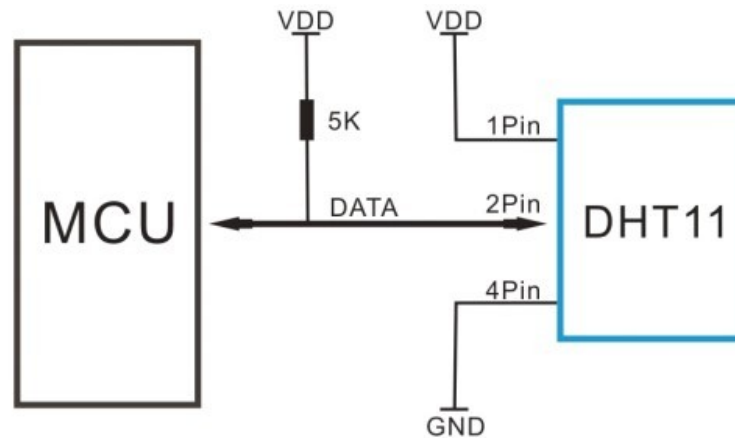


#### **DHT11 Specifications:**

- 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:  $\pm 1^{\circ}\text{C}$  and  $\pm 1\%$

#### **How to use DHT11 Sensor:**

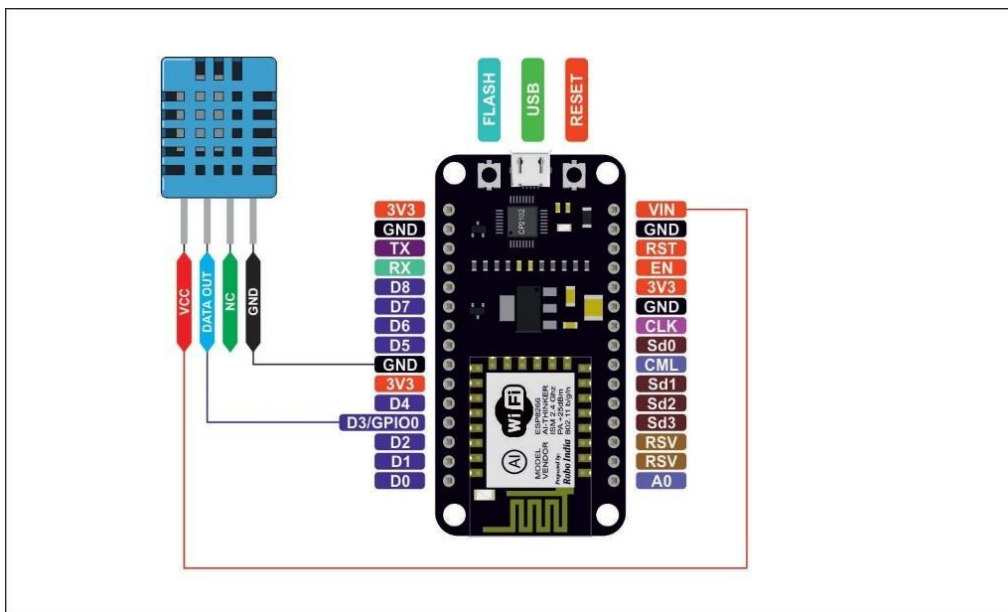
The DHT11 Sensor is factory calibrated and outputs serial data and hence it is highly easy to set it up. The connection diagram for this sensor is shown below



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.

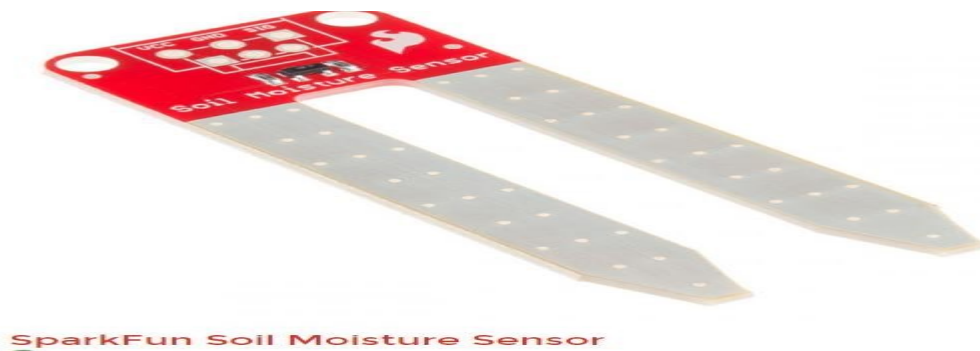
If you are trying to interface it with some other MCU then the datasheet given below will come in handy. The output given out by the data pin will be in the order of 8bit humidity integer data + 8bit the Humidity decimal data + 8 bit temperature integer data + 8bit fractional temperature data + 8 bit 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.

#### Dht11 interface with node mcu:



## Soil Moisture Sensor

To know when plants want to be watered in the garden



The Soil Moisture Sensor is pretty straightforward when it comes to hookup. There are only three pins to connect: **VCC**, **GND**, and **SIG**.

You need to supply power to VCC and GND. We recommend not powering the sensor constantly to prevent corrosion of the probes (more on this in a bit). SIG provides an analog signal out that can be attached to the ADC pin on any microcontroller. The value read on SIG will vary depending on the voltage with which you power the sensor.

### Theory of Operation

The two probes are acting as a variable resistor – more water in the soil means better conductivity and results in a lower resistance and a higher SIG out. Your analog readings will vary depending on what voltage you use for Vcc as well as the resolution of your ADC pins.

#### Powering the Soil Moisture Sensor

We recommend powering the Soil Moisture Sensor with between **3.3V - 5V**. Please note that the analog value returned will vary depending on what voltage is provided for the sensor.

One commonly known issue with soil moisture sensor is their short lifespan when exposed to a moist environment. To combat this, we've had the PCB coated in Gold Finishing ([Electroless Nickel Immersion Gold](#)).

Another way to extend the lifespan of your sensor is to only power it when you take a reading. Using a digital pin set to HIGH on an Arduino, for example, is an easy way to accomplish this. If



you wish to power the sensor with more than a digital pin on your microcontroller can provide, you could always use a [transistor](#).

### System Calibration

To get any sort of useful data out of your Soil Moisture Sensor, it is advised that you calibrate it to whatever soil you plan to monitor. Different types of soil can affect the sensor, and you may get different readings from one composition to the next. Before you start storing moisture data or triggering events based on that value, you should see what values you are actually getting from your sensor. Using the sketch above, note what values your sensor outputs when the sensor is completely dry vs when the sensor is completely submerged in a shallow cup of water. Depending on what microcontroller you're using, the operating voltage of that microcontroller, and the resolution of its analog-to-digital converter, your results will vary.

Once you have an idea what values your sensor is outputting in completely dry and completely wet situations, it's time to calibrate your sensor for the specific soil you want to monitor. Do the same test above, only this time test your soil when it is as dry as possible, then measure it when the soil is completely saturated with moisture. Getting these values and comparing them to the ones from the previous calibration will give you the best insight into what values mean for your specific plant and soil. This test may take some trial and error and patience. Be careful not to over-water (or under-water) your plants during these tests.

## Motion sensor (PIR ("Passive Infrared"))

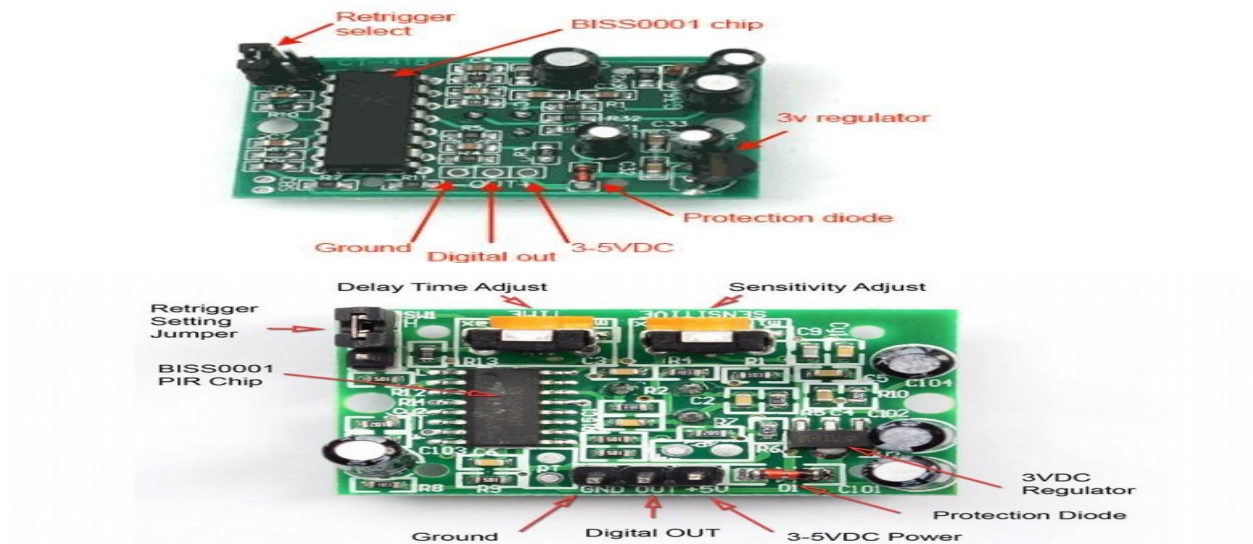
It used to detect whether a human has moved in or out of the sensors range. They are small, inexpensive, low-power, easy to use and don't wear out.

### How PIRs Work

PIR sensors are more complicated than many of the other sensors explained in these tutorials (like photocells, FSRs and tilt switches) because there are multiple variables that affect the sensors input and output. To begin explaining how a basic sensor works, we'll use this rather nice diagram

The PIR sensor itself has two slots in it, each slot is made of a special material that is sensitive to IR. The lens used here is not really doing much and so we see that the two slots can 'see' out past some distance (basically the sensitivity of the sensor). When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors. When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a *positive differential* change between the two halves. When the

warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.



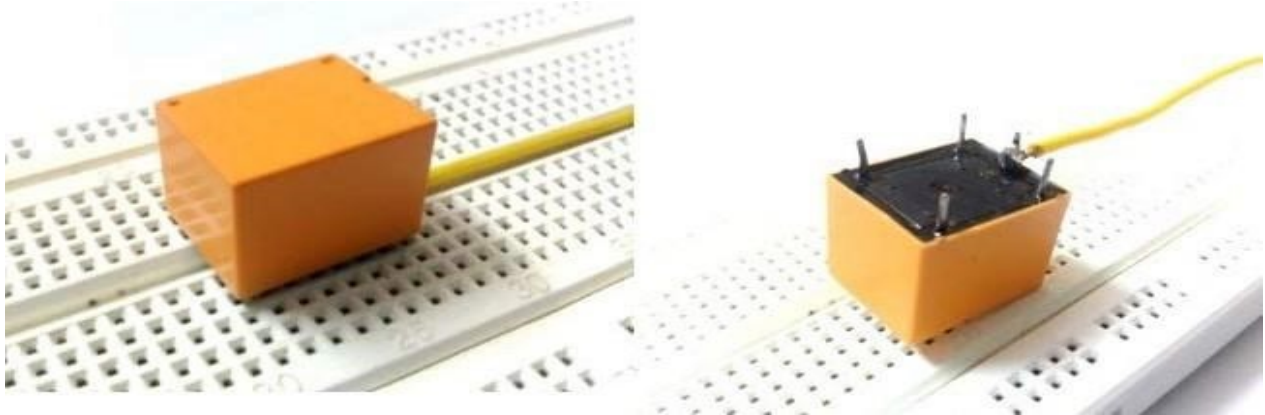
## The PIR Sensor

The IR sensor itself is housed in a hermetically sealed metal can to improve noise/temperature/humidity immunity. There is a window made of IRtransmissive material (typically coated silicon since that is very easy to come by) that protects the sensing element. Behind the window are the two balanced sensors. Theory of Operation

Now when the PIR detects motion, the output pin will go "high" to 3.3V and light up the LED!

Once you have the breadboard wired up, insert batteries and wait 30-60 seconds for the PIR to 'stabilize'. During that time the LED may blink a little. Wait until the LED is off and then move around in front of it, waving a hand, etc, to see the LED light up!

## Relays:



### What is a Relay?

It is used as a switch. Dictionary says that relay means *the act of passing something from one thing*

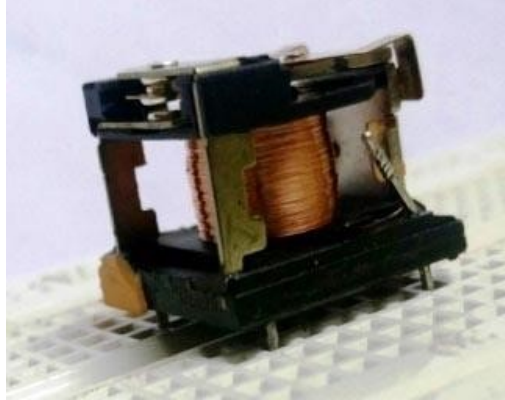
*to another*, the same meaning can be applied to this device because the signal received from one side of the device controls the switching operation on the other side. So relay is a switch which controls (open and close) circuits electromechanically.

### Operation of relays

The main operation of this device is to make or break contact with the help of a signal without any human involvement in order to switch it ON or OFF. It is mainly used to control a high powered circuit using a low power signal. Generally, a DC signal is used to control circuit which is driven by high voltage like controlling AC home appliances with DC signals from microcontrollers.

### Design of a Relay

An electromechanical relay is basically designed using few mechanical parts like Electromagnet, a movable armature, contacts, yoke, and a spring/frame/stand, these parts are showing in the internal pictures of Relay below. All these are arranged logically to form into a relay.



### **The mechanical operation of relays**

An Electromagnet plays a major role in the **working of a relay**. It is a metal which doesn't have magnetic property, but it can be converted into a magnet with the help of an electrical signal. We know that when current passes through the conductor it acquires the properties of a magnet. So, when a metal winded with a copper wire and driven by the sufficient power supply, that metal can act as a magnet and can attract the metals within its range **Movable Armature:**

Movable armature is a simple metal piece which is balanced on a pivot or a stand. It helps in making or breaking the connection with the contacts connected to it.

#### **Contacts:**

These are the conductors that exist within the device and are connected to the terminals.

#### **Yoke:**

It is a small metal piece fixed on a core in order to attract and hold the armature when the coil is energized.

#### **Spring (optional):**

It is connected to one end of the armature to ensure its easy and free movement. Instead of a spring, a metal stand like structure can be used.

### **Construction of Relay and its operation:**

A core with copper windings (forms a coil) winded on it is placed. A movable armature consists of a spring support or stand like structure connected to one end, and a metal contact connected to another side, all these arrangements are placed over the core such that, when the coil is energized, it attracts the armature. The movable armature is

generally considered as a common terminal which is to be connected to the external circuitry.

The relay also has two pins namely ***normally closed and normally opened (NC and NO)***, the normally closed pin is connected to the armature or the common terminal whereas the normally opened pin is left free (when the coil is not energized). When the coil is energized the armature moves and is get connected to the normally opened contact till there exists flow of current through the coil. When it is de-energized it goes to its initial position.

The general **circuit representation of the relay** is as shown in the figure below

### **How Relay Works:**

#### **Relay in NORMALLY CLOSED condition:**

When no voltage is applied to the core, it cannot generate any magnetic field and it doesn't act as a magnet. Therefore, it cannot attract the movable armature. Thus, the initial position itself is the armature connected in normally closed position (NC).

#### **Relay in NORMALLY OPENED condition:**

When sufficient voltage is applied to the core it starts to create a magnetic field around it and acts as a magnet. Since the movable armature is placed within its range, it gets attracted to that magnetic field created by the core, thus the position of the armature is being altered. It is now connected to the normally opened pin of the relay and external circuit connected to it function in a different manner.

**Note:** The functionality of the external circuit depends upon the connection made to the relay pins.

So finally, we can say that when a coil is energized the armature is attracted and the switching action can be seen, if the coil is de-energized it loses its magnetic property and the armature goes back to its initial position.

## **Different Types of Relay:**

Other than the Electromagnetic relay there are many other **types of relays** which work on different principles. Its classification is as follows

### **Types of Relay Based on the principle of operation**

- **Electrothermal relay:**

When two different materials are joined together it forms into a bimetallic strip. When this strip is energized it tends to bend, this property is used in such a way that the bending nature makes a connection with the contacts.

- **Electromechanical relay:**

With the help of few mechanical parts and based on the property of an electromagnet a connection is made with the contacts.

- **Solid State relay:**

Instead of using mechanical parts as in electrothermal and electromechanical relays, it uses semiconductor devices. So, the switching speed of the device can be made easier and faster. The main advantages of this relay are its more life span and faster switching operation compared to other relays.

- **Hybrid relay:**

It is the combination of both electromechanical and solid state relays.

### **Types of Relay Based on the polarity:**

- **Polarized relay:**

These are similar to the electromechanical relays but there exists both permanent magnet and electromagnet in it, the movement of the armature depends on the polarity of the input signal applied to the coil. Used in telegraphy applications.

- **Non-polarized relay:**

The coil in these relays doesn't have any polarities and its operation remains unchanged even if the polarity of the input signal is altered.



## Applications of Relay:

The **applications of the relay** are limitless, its main function is to control the high voltage circuit (230V circuit AC) with the low voltage power supply (a DC voltage).

- Relays are not only used in the large electrical circuits but also used in the computer circuits in order to perform the arithmetic and mathematical operations in it.
- Used to control the electric motor switches. To turn ON an electric motor we need 230V AC supply but in few cases/applications, there may be a situation to switch ON the motor with a DC supply voltage. In those cases, a relay can be used.
- Automatic stabilizers are one of its applications where a relay is used. When the supply voltage is other than the rated voltage, set of relays sense the voltage variations and controls the load circuit with the help of circuit breakers.
- Used for the circuit selection if there exists more than one circuit in a system.
- Used in Televisions. An old picture tube television's internal circuitry works with the DC voltage but the picture tube needs a very high AC voltage, in order to turn on the picture tube with a DC supply we can use a relay.
- Used in the traffic signal controllers, temperature controllers.

## Code of project :

```
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <Adafruit_Sensor.h>
#include <DHT.h> //Include DHT sensor library
#define DHTPIN 2 // What digital pin temperature and humidity sensor is connected to
DHT dht(DHTPIN, DHTTYPE);

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "25bcde6b820642a39af04db6ea73fc85";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "ROCK";
char pass[] = "*****";
void setup()
{
  // Debug console
  Serial.begin(9600);

  Blynk.begin(auth, ssid, pass);
  // You can also specify server:

  Blynk.begin(auth, ssid, pass);
  // You can also specify server:
  Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 80);
  Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);
  dht.begin(); // Begins DHT reading

  Blynk tweet("SMART HOME IS ONLINE! "); // Tweating on your Twitter Handle that you project is online

  pinMode(pirPin,INPUT); // Defining that Pir Pin is meant to take Input Only
  pinMode(soilPin,INPUT); // Defining that Soil Sensor Pin is meant to take Input Only

  // Setup a function to be called every second
  timer.setInterval(1000L, sendSensor);
}
#define soilPin 4 // What digital pin soil moisture sensor is connected to
#define gasPin A0 // What analog pin gas sensor is connected to
#define pirPin 12 // What digital pin //soil moisture// sensor is connected to

int pirValue; // Place to store read PIR Value
int soilValue; // Place to store read Soil Moisture Value
int PIRpinValue; // Place to store the value sent by Blynk App Pin V0
int SOILpinValue; // Place to store the value sent by Blynk App Pin V1

// Uncomment whatever type you're using!
#define DHTTYPE DHT11 // DHT 11
// #define DHTTYPE DHT22 // DHT 22 (AM2302) (I2C)
```

```

BlynkTimer timer;
// This function sends Arduino's up time every second to Virtual Pin (5).
// In the app, Widget's reading frequency should be set to PUSH. This means
// that you define how often to send data to Blynk App.

BLYNK_WRITE(V0)          //V0 pin from Blynk app tells if Motion Detection is ON
{
  PIRpinValue = param.asInt();
}

BLYNK_WRITE(V1)          //V1 pin from Blynk app tells if Soil Moisture is ON
{
  SOILpinValue = param.asInt();
}

void sendSensor()
{
  int h = dht.readHumidity();
  int t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!"); // to check if sensor is not sending any false values
    return;
  }

```

```

//, // don't send more than 10 values per second.
Blynk.virtualWrite(V5, h); // send humidity to pin V5
Blynk.virtualWrite(V6, t); // send temperature to pin V6
}

void getPirValue(void)
{
  pirValue = digitalRead(pirPin);
  if (pirValue == HIGH) //digital pin of PIR gives high value on human detection
  {
    Serial.println("Motion detected");
    Blynk.notify("Motion detected");
  }
}

void getSoilValue(void)
{
  soilValue = digitalRead(soilPin);
  if (soilValue == HIGH) //digital pin of soil sensor give low value when humidity is less
  {
    Serial.println("Water Plants");
    Blynk.notify("Water Plants");
  }
}

```

```
soilValue = digitalRead(soilPin);
if (soilValue == HIGH)    //digital pin of soil sensor give low value when humidity is less
{
    Serial.println("Water Plants");
    Blynk.notify("Water Plants");
}

void loop()
{
    Blynk.run();
    timer.run();

    if (PIRpinValue == HIGH)    //V0 pin from Blynk app tells if Motion Detection is ON
    {
        getPirValue();
    }

    if (SOILpinValue == HIGH)    //V1 pin from Blynk app tells if Soil Moisture is ON
    {
        getSoilValue();
    }
}
```

# Home Automation Using NodeMCU and Google Assistant.

## Step 1: Downloading and Installing the Blynk App on the Smartphone.

1. Open Play store and download and install the Blynk App.



2. Once the app is installed either login to it using Facebook or create a new account on Blynk and login using that. In this case I have logged in using Facebook.
3. After logging in, create a new project by clicking 'New Project'.
4. Give the project a name of your liking. Select the hardware device as NodeMCU and select the connection type as WIFI, and hit create.
5. At this point Blynk will send an *Auth token* to your email id. We will use this 'Auth token' later in the tutorial to link our app with the NodeMCU.

## Step 2: Downloading Arduino IDE and Configuring Blynk Libraries.

1. Now let's install the Arduino IDE. Go to **this** link and download the IDE for your preferred operating system.
2. Download the latest Blynk libraries from **this** link. These libraries will help us connect the Blynk app with the NodeMCU.
3. Extract the download zip file in a folder.
4. Open up *Arduino IDE*, go to: File > Preferences and under the *Settings* tab, copy the sketchbook location .
5. Now open the file explorer and go to the copied path location. This is the path where all the Blynk libraries are installed. So, we'll have to copy all the newly downloaded Blynk libraries into this folder.

6. Copy the files/folders from the *Libraries* folder of the downloaded Blynk directory, and paste it to the *Libraries* folder of your Arduino IDE's directory (The path that we copied in step 5).
7. Similarly, copy the files/folder from the *Tools* folder of the downloaded Blynk directory, and paste it to the *Tools* folder of your Arduino IDE's directory.

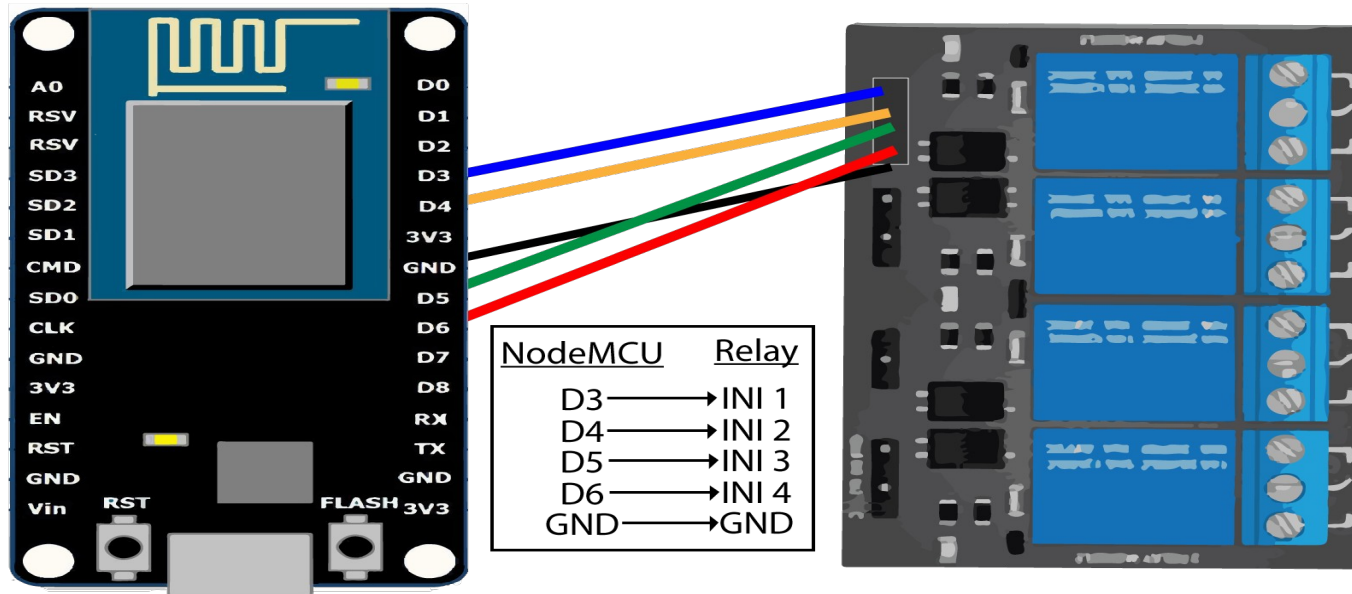
### **Step 3: Uploading the code to NodeMCU.**

1. Connect the NodeMCU to your PC using a USB cable.
2. Now, we'll set up the Arduino IDE by changing some settings. So, open up the Arduino IDE.
3. Go to *Tools > Port* and make sure an appropriate port is selected. In my case it's *COM 4*. This is the USB port in which the NodeMCU is connected.
4. Now Go to *Tools > Board* and select '*NodeMCU 1.0 (ESP-12E Module)*' as the board. And that's all the settings we need to change. So now let's begin writing some code.
5. Go to *Files > Examples > Blynk > Boards\_WIFI > ESP8266\_Standalone*. A new file with some prewritten code will open.
6. Now, in this file we only need to change 3 lines of code.
  1. Change the line where it says '*char auth[] = "YourAuthToken"*' and replace the '*YourAuthToken*' part with your Blynk's auth token that you received in your email.
  2. Change the line where it says '*char ssid[] = "YourNetworkName"*' and replace the '*YourNetworkName*' part with the name of your WIFI network that you want your NodeMCU to connect to. In my case the name of my WIFI network is '*The Network*'
  3. Change the line where it says '*char pass[] = "YourPassword"*' and replace the '*YourPassword*' part with the password of your WIFI network. In my case passwo
7. That's really all the code that we need to write! We are now ready to upload this code to the NodeMCU. So directly hit upload button at the top (besides the button that has a checkmark), and wait for it to process.
8. The code will be uploaded to the NodeMCU and the next time you power it on, it will automatically connect to the specified WIFI network.

### **Step 4: Hardware Assembly.**



1. We'll have to connect the NodeMCU with the Relay board, you can choose to do it with a bread board or without. But I prefer doing it using a Breadboard.
2. Connect Ground Pin of Relay with Ground Pin of NodeMCU.



3. Now to power up the NodeMCU you can use a normal phone charger, just make sure its voltage is not too high. And to power up the Relay board, you can use a battery or a separate **breadboard power supplier**.
4. As we are using a four-channel relay you can connect at most 4 electronic appliances to the Relay and control them over the internet.
5. Now if you want to connect your household appliances like Fan, Lights etc. which are connected to the main power of your house, I would recommend you take the help of a professional electrician and ask him/her to connect those appliances to the relay. Because working with the mains is no joke and if not done properly, can cause a serious damage.

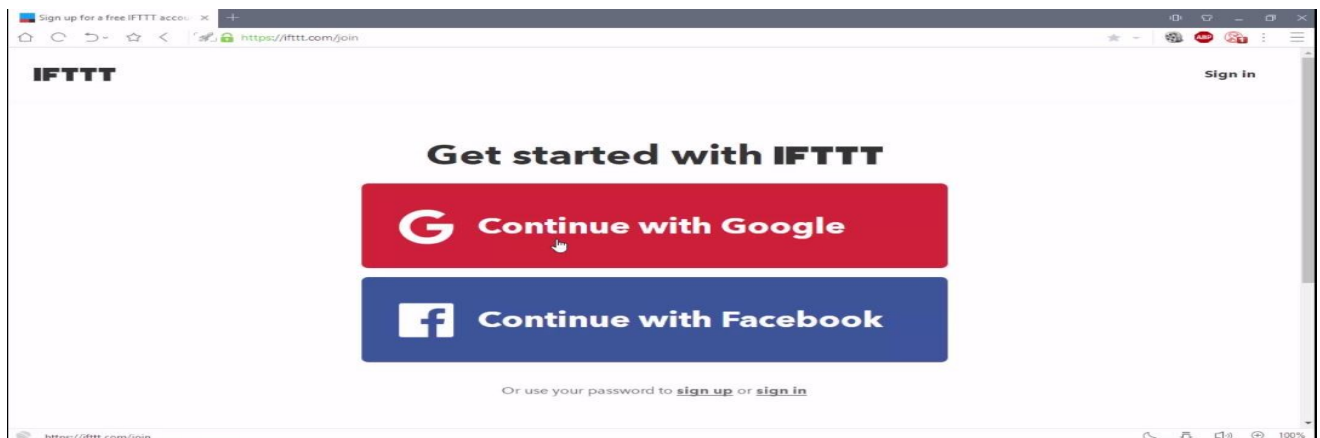
At this point, we have a fully functional connection between the NodeMCU, Blynk app and our electrical appliances. So, you can directly run your Blynk project from your phone and turn the electrical appliances on or off using the buttons that we created in the app. And if you are satisfied with this and don't want to connect the NodeMCU with the Google Assistant and control the appliances using voice commands, then you don't have to read the remaining tutorial and you can stop right here. Otherwise let's move forward.

### Step 5: Connecting Google Assistant (using IFTTT) to make the NodeMCU work with voice commands.

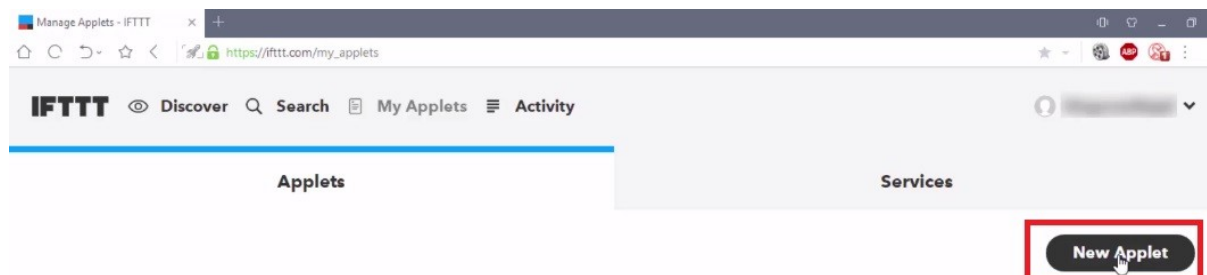
We cannot connect the Google Assistant to the NodeMCU directly, and that is the only reason we are using the Blynk app. Blynk app can directly connect to the NodeMCU and send data to it. So, if we can send the voice commands interpreted by google assistant directly to the Blynk app, the Blynk app can then forward those commands to the NodeMCU. But the problem is Google Assistant cannot directly understand foreign commands like “turn on the fan” or “turn on relay one” etc. on its own. So, to solve this we use another intermediate app/website called **IFTTT**.


Simply, to control our home appliances over the internet we are using NodeMCU and to connect NodeMCU with the home appliances we use a relay board. Now to send on or off signals to the NodeMCU we use our smartphone, and we do this using the Blynk app. But we want to send the on or off signals using voice commands. To do this we use google assistant in our smartphone and an app called IFTTT.

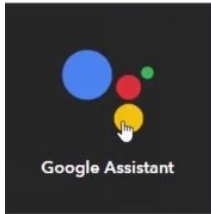
So, in the end what will happen is, when we say a voice command like “ok google turn on the light” to the Google Assistant, Google Assistant sends that this foreign command to IFTTT. IFTTT interprets this command and sends an On or Off signal to the Blynk app via the Blynk Server. Blynk will then send this signal to the NodeMCU and then to our electrical appliances. Enough said, let's configure IFTTT. Go to IFTTT's website and sign up to it using your Google Account.



1. After Signing in click on *My Applets* from the header and select *New Applet*.



2. Click on *'this'*.  **this**
3. Search for *Google Assistant* and select it. And then Click on *Connect*.



4. At this point IFTTT will ask you permission to use your google account to add voice commands to it. Which you simply allow by clicking on *'Allow'*.
5. Select the card that says *"Say a simple phrase"*.
6. Next, for the first textbox type the phrase that you want to say to Google Assistant. It can be anything such as *"Turn on the T.V"*, *"Turn on the fan"* or anything you like.
7. For the next two text boxes, you write some other ways to say the first command. For example, if in the first textbox you wrote *"Turn on the T.V"*, then in the second and third textboxes you can write something like *"Turn the T.V On"* or *"Please Turn on the T.V"* or *"Turn the Idiot Box On"*.
8. In the fourth textbox type the reply that Google Assistant should respond with. For example, *"Okay, Turning on the T.V"*.

you choose. For example, say "Ok Google, I'm running late" to text a family member that you're on your way home.

What do you want to say?

turn on relay one

What's another way to say it? (optional)

turn the first relay on

And another way? (optional)

turn on the first relay

What do you want the Assistant to say in response?

ok, turning on relay one

Create trigger

9. Finally, click on *'Create Trigger'*.

10. Now, click on that **+that** and type webhooks select it, and click connect. Webhooks will allow us to send commands to the Blynk Server.
11. Now, in the URL field type this URL:

This is the URL of Blynk Server , but it should work for other places as well. Replace the “YourAuthTokenHere” part with your Blynk Auth token that you received in the mail. And “DigitalPinToBeUpdateHere” part with the Digital pin of NodeMCU that is to be updated. So, as we assigned the Digital Pin D3 of NodeMCU to relay one we must write D3 in place of “DigitalPinToBeUpdateHere”. But wait we cannot write D3 there, because when Blynk Server receives this command from IFTTT it assumes as if the command it received was to be sent to an ‘Arduino Uno’ board, but in our case, we are sending it to NodeMCU.

To solve this, we must type the Digital pin of Arduino which corresponds with the NodeMCU. You can find the mapping

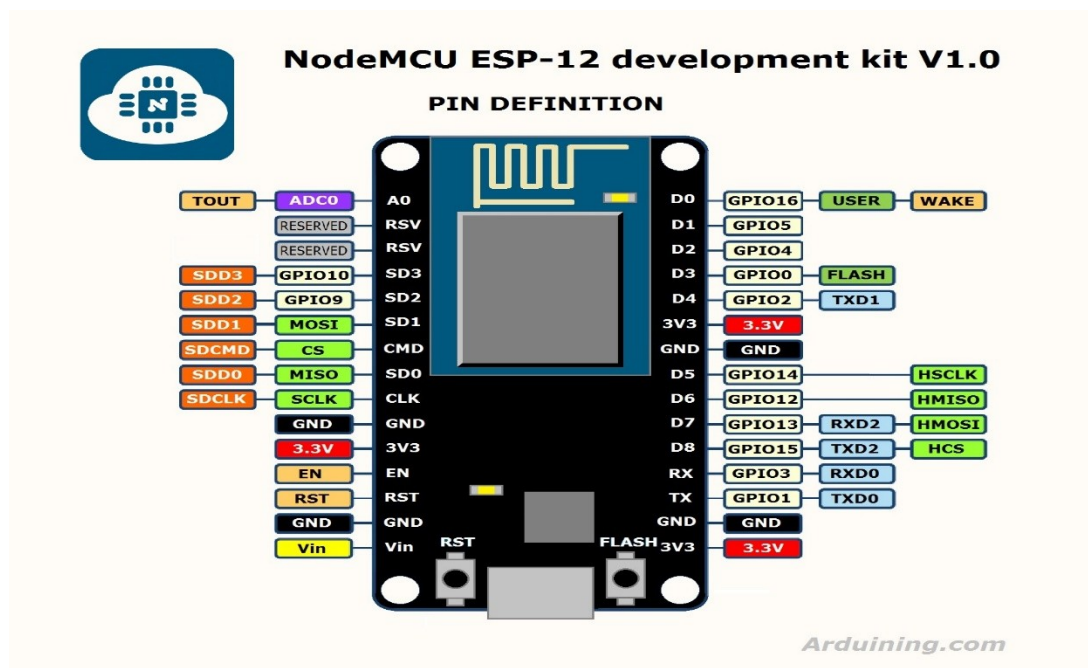


image below.

As you can see in the image, Digital Pin D3 of NodeMCU corresponds to Pin D0 of Arduino. So instead of D3, I'll write D0 as the pin. In the end the URL should look

be rate limited.

URL

`http://188.166.206.43/97952012247af6/update/D0`

Surround any text with "<<<" and ">>>" to escape the content

Add ingredient

Method

PUT

The method of the request e.g. GET, POST, DELETE

Content Type

application/json

Optional

Body

something like this:

Next, Select the 'Method' field as PUT

12. Select 'Content type' as *Application/JSON*.

13. For the 'Body' type this: ["0"]

Body

["0"]

Here '0' means to turn on, so we are basically saying Blynk to turn on relay that is connected to pin D3, which in our case is Relay one.

14. Now click on 'Create Action' and then Finish.

15. Similarly, we create another applet to turn off the relay. Repeat all the steps above from step 4 except the following changes:

1. In step 8 and 9, instead of writing "Turn on the T.V", type "Turn off the T.V"

2. In step 15, instead of ["0"], type ["1"].

16. So now we have successfully created two triggers to turn on and off one Relay. So Similarly, we create triggers for remaining 3 relays. Just change the phrase and Digital pin for each Relay. All the other steps will remain the same. So, in the end for 4 relays, we should have 8 triggers to turn each of them on or off.

### 3.RFID and Keypad Based Access Control System Using

**Arduino:**



**Overview:** User will first scan the right tag and then enter the password for that tag to open the door lock. Master tag will add/rem other tags.

#### **Hardware components:**

1-arduino uno.

2-lcd.

3-keypad4x4.

4-leds(red,green,blue).

5-servo.

6-buzzer.

## Story

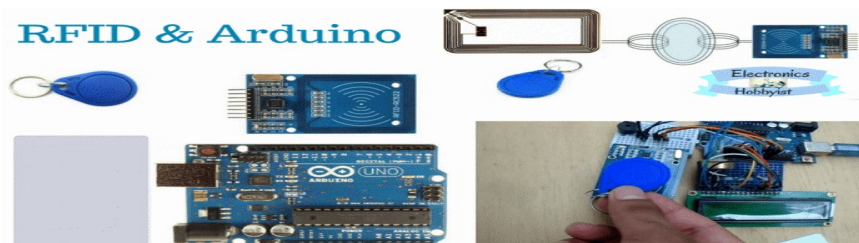
In the previous post, we made an RFID based access control and alert system using Arduino in which system sends us messages when the access was granted or denied. We was also be able to open the door lock and halt the system by sending the message to Arduino. There was a master tag that was used to add/remove other tags.

In this post, we are going to build an RFID and Keypad based access control system using Arduino in which the user will have to first scan the right tag and then he will have to enter the password for that tag to open the door lock. There will be a master tag that will be used to add/remove other tags and each tag will have its own password.

The saved tags and password will remain saved even after powering off the module. The only way to reset the system is by using the wipe button that will erase all the data in the EEPROM. EEPROM has roughly 100, 000 limited Write cycle.

## RFID basics and RFID Module Interfacing with Arduino

In this article, you are going to learn about RFID Arduino interfacing using MFRC522 RFID Reader with the help of two examples. In the first example, we will simply make an Arduino RFID Reader and in the second example, we will make an Arduino RFID Door Lock.



## RFID Basics

RFID stands for radio frequency identification and it basically uses the radio waves to read the information on the tag. The RFID tags contains the embedded transmitter and receiver attached to an object.

RFID is fast and does not require any contact between the reader and the tag and they can be read from feet's away.



An RFID system consists of two parts: Tag and Reader

## RFID Tag

An RFID tag contains a chip for storing information about physical object and an antenna to receive and transmit a signal. A RFID tag can usually store 1KB of data but it is enough for storing the name, credit card number, unique identification number, birth date and some more information.



RFID Tags can be passive or active

- 20. A **Passive tag** has no battery and it uses the energy transmitted by the reader.
- 21. An **Active tag** contains a built in battery which makes it able to send a stronger signal and the range increases to 100 feet. Other features are same as the passive tags.

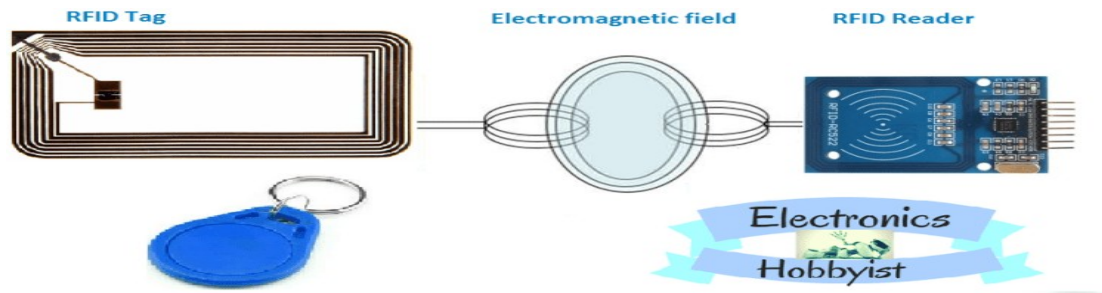
## RFID Reader

The RFID reader performs two functions: Transmit and receive. So you can also say it a transceiver. The RFID reader contains an antenna, radio frequency module and a control unit.



## How RFID works

The RFID reader generates a high frequency electromagnetic field and when the tag comes near it, a voltage is induced in tags antenna coil due to induction. This induced voltage acts as power for the tag. The tag in return converts the signal in power and responds to the reader.



## RFID Arduino Interfacing

Now let's interface the RFID reader module with Arduino. The RFID reader we are going to use is MFRC522 reader module and it communicates with the Arduino through SPI protocol. It operates at 13.56 MHz frequency.

The tags are based on MIFARE protocol and they have 1kb of memory. They also have a microchip that can perform arithmetic operations.

For RFID Arduino interfacing, you are going to require the following parts

- Arduino Uno or any other Arduino
- MFRC 522 RFID Module
- I2C LCD Display
- Three LED's (Green, Red, Blue)
- Sg90 servo motor
- 3 X 220 ohm resistors
- Buzzer

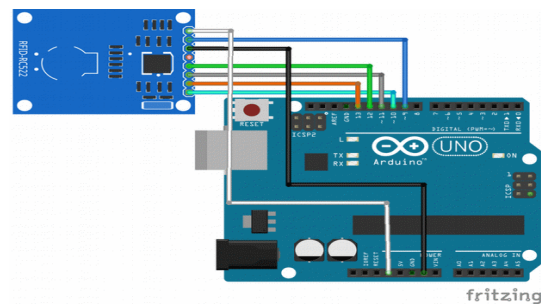
Example 1 – Let's make an Arduino RFID Reader

In the first example, we are going to make a Arduino RFID Reader that will read the information on the tag and will display it on the serial monitor.

Connect the MFRC522 module with the Arduino as described in the below table. Different Arduino's have different SPI pins so i have shown the connections for different Arduino's. I am using the Arduino Uno so i am going to connect it according to that. If you are using different Arduino, then make connections according to that.

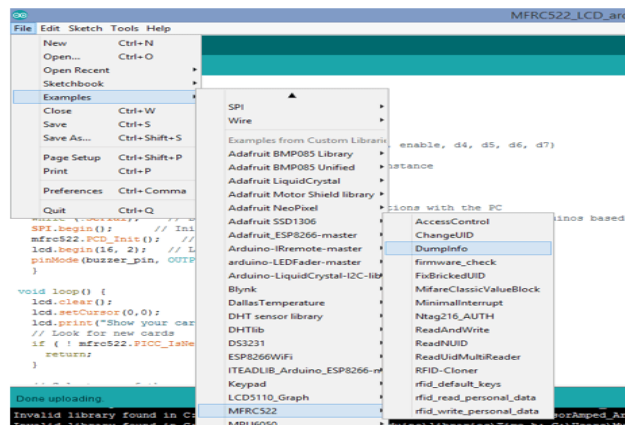
MFRC522	Arduino Uno /101
3.3V	3.3V Pin
RST	9
GND	GND
NC	NC
MISO	12/ICSP-1
MOSI	11/ICSP-4
SCK	13/ICSP-3
SDA	10

The circuit diagram is shown below

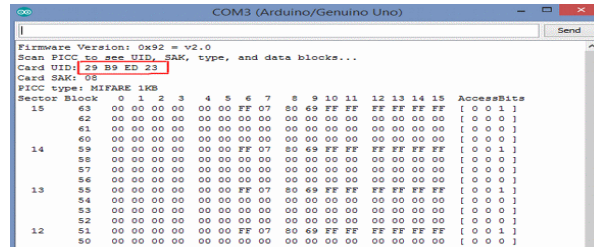


*Download Library and Upload Code*

After that, open the “Dumpinfo” from examples and upload it in your Arduino IDE.



Now open the serial monitor and bring the RFID card in front of the reader, it will show you the information of the RFID tag. Save the UID number, we are going to use it in the second example.



In the above image, you can see the UID number of the tag and also the 1kb of memory which is divided into 16 sectors. Each sector has 4 blocks and each block can store 4 bytes.

## Example 2 – Let's make an Arduino RFID Door Lock

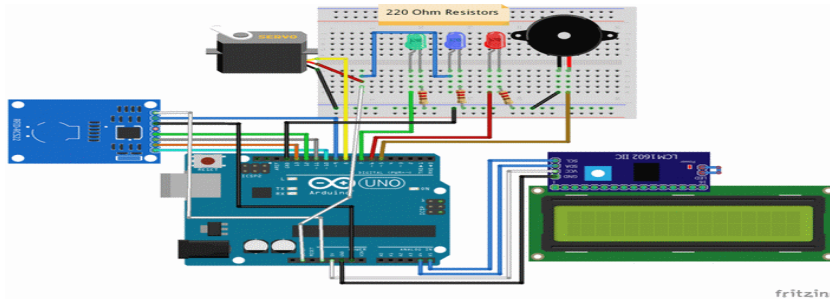
In the second example, we are going to make an Arduino RFID Door Lock that will open the door on scanning the right tag and will deny access on scanning the wrong tag.

So you already have connected the RFID module with Arduino in the previous example. Let's now connect the I2C LCD with the Arduino. Different Arduino boards have different I2C pins so i have explained connections for different Arduino boards. Make the connections as described in below table.

I2C LCD	Arduino Uno/101
SCL	A5/ SCL
SDA	A4/ SDA
GND	GND
VCC	5V

The below table describes the connections of Arduino with other components.

Arduino	LED's	Servo	Buzzer
5V	Blue LED	Positive Wire (Red )	
D8		Signal Wire (Yellow)	
D7	Green LED		
D6	Red LED		
D5			+
GND	Negative leads through 220 ohm resistor	GND Wire (Brown)	-



Next we are going to connect the keypad with Arduino. The 4X4 keypad has 8 connections but we don't require the last column of keypad. We only require numbers for the password. So we won't use the last pin of keypad which is for fourth column. Connections for keypad with Arduino are given in below table. You can also use 4X3 keypad instead of 4X4 keypad.

4X4 Keypad	Arduino
1 <sup>st</sup> Pin	A0
2 <sup>nd</sup> Pin	A1
3 <sup>rd</sup> Pin	A2
4 <sup>th</sup> Pin	A3
5 <sup>th</sup> Pin	D2
6 <sup>th</sup> Pin	D1
7 <sup>th</sup> Pin	D0

## Working of RFID and Keypad based access control system using Arduino

On starting the project for the first time, it will ask you to **define a master tag and password for it**. The master tag will act as programmer and you can use it to add or remove other tags.

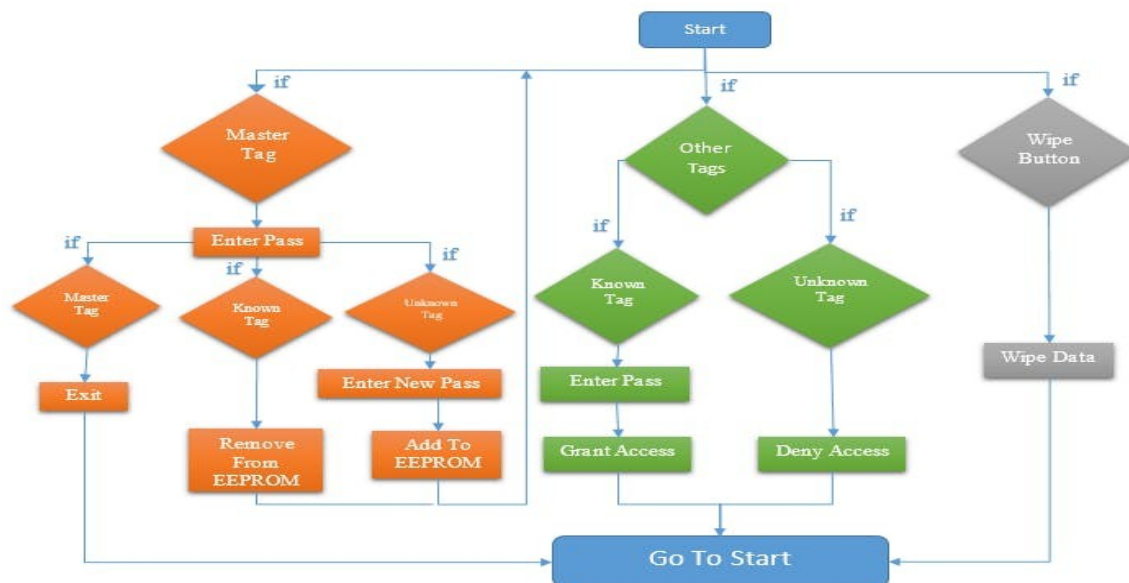
After defining the master tag, you will have to add other tags that you can use to open the door. To do this, scan the master tag and enter the password for it and it will take the system into **program mode**.

In the program mode, scanning the tags will **add/Remove** these from the system. Scan the tags that you want to use to open the door and enter the password for that tag. The system will store the UID and password of these tags in the EEPROM. Scan the tag

again and enter its password to remove it from the EEPROM. To exit the program mode, scan the master tag.

Now scan the tags that you have added in the system and enter their passwords to **open the door** and on scanning the wrong tag or on entering the wrong password, the door will remain closed.

To **reset the system**, press the reset button of Arduino and then long press the wipe button for 10 seconds. This will remove all the data from the EEPROM including the master tag.



### Circuit Diagram and Explanation

The RFID reader communicates with the Arduino through the SPI protocol and different Arduino boards have different SPI pins.

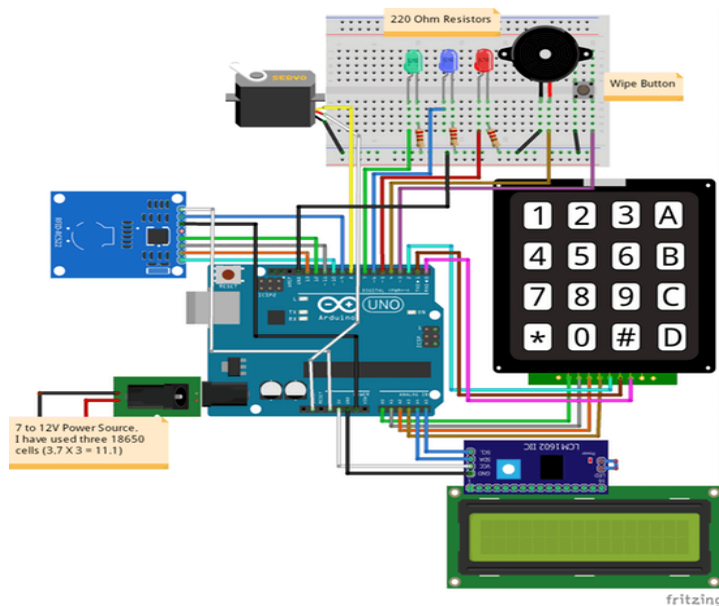
To test if the RFID reader is working properly or not, upload the “dumpinfo” from the examples in the Arduino and see if it is showing the information of the tags on the serial monitor or not. If you are new to RFID, then follow this tutorial | **RFID basics and RFID module interfacing with Arduino**

The I2C LCD communicates with the Arduino through the I2C protocol. Different Arduino boards have different I2C pins. The I2C pins on Arduino Uno and Arduino Nano are A4, A5.

Next we are going to connect the keypad with Arduino. The 4X4 keypad has 8 connections but we don't require the last column of keypad. We only require numbers for the password. So we won't use the last pin of keypad which is for fourth column. You can also use 4X3 keypad instead of 4X4 keypad.

In the end, connect the power source to the Arduino. I have used three 18650 cells. We can give 6 to 12V to the Arduino through the barrel jack.

The complete circuit diagram is as follows

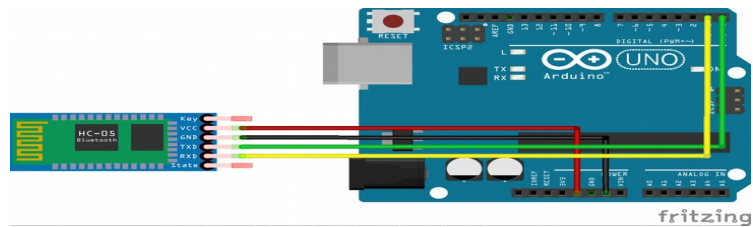


## SMS Part :

- Using Bluetooth module(HC-05):

Interfacing with Arduino:





Bluetooth Pin	To Arduino Pin
TX	(RX (Pin 0
RX	(TX (Pin 1
VCC	5V
GND	GND

How it works?

The Bluetooth module works in the form of Serial Communication, where the Android program sends data to the Bluetooth module in the case of pressing a button, and on the other hand when the Bluetooth module receives any data from the device associated with it (mobile in this case) ), Sends this data to the Arduino panel via the Tx of the module (Rx in the Arduino panel).

The Android Program:

You can use any program designed to handle Bluetooth and Android based on Google Store, and you can develop your own program (BlueTerm).

## Code of project :

```
#include <MFRC522.h>
#include <MFRC522Extended.h>
#include <deprecated.h>
#include <require_cpp11.h>

// Include required libraries
#include <MFRC522.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <Servo.h>
#include <SPI.h>
#include <SoftwareSerial.h>
SoftwareSerial BTserial(0, 1); // RX | TX
// Create instances
LiquidCrystal_I2C lcd(0x27, 16, 2);
MFRC522 mfrc522(10, 9); // MFRC522 mfr522(SS_PIN, RST_PIN)
Servo sg90;
// Initialize Pins for led's, servo and buzzer
// Blue LED is connected to 5V
constexpr uint8_t greenLed = 7;
constexpr uint8_t redLed = 6;
constexpr uint8_t servoPin = 8;
constexpr uint8_t buzzerPin = 5;
char initial_password[4] = {'1', '4', '7', '*'}; // Variable to store initial password
String tagUID = "11 8E 4F D3"; // String to store UID of tag. Change it with your tag's UID
char password[4]; // Variable to store users password
```

```
boolean RFIDMode = true; // boolean to change modes
char key_pressed = 0; // Variable to store incoming keys
uint8_t i = 0; // Variable used for counter
// defining how many rows and columns our keypad have
const byte rows = 4;
const byte columns = 4;
// Keypad pin map
char hexaKeys[rows][columns] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
// Initializing pins for keypad
byte row_pins[rows] = {A0, A1, A2, A3};
byte column_pins[columns] = {2};
// Create instance for keypad
Keypad keypad = Keypad( makeKeymap(hexaKeys), row_pins, column_pins, rows, columns);
void setup() {
  // Arduino Pin configuration
  pinMode(buzzerPin, OUTPUT);
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  sg90.attach(servoPin); //Declare pin 8 for servo
  sg90.write(0); // Set initial position at 90 degrees
}
```

```

lcd.begin(); // LCD screen
lcd.backlight();
BTserial.begin(9600); // Initiate a serial communication
BTserial.println("Waiting for connections...");
SPI.begin(); // Init SPI bus
mfrc522.PCD_Init(); // Init MFRC522
lcd.clear(); // Clear LCD screen
BTserial.println("Put your card to the reader...");
BTserial.println();

}

void loop() {
  // System will first look for mode
  if (RFIDMode == true) {
    lcd.setCursor(0, 0);
    lcd.print(" Door Lock");
    lcd.setCursor(0, 1);
    lcd.print(" Scan Your Tag ");
    // Look for new cards
    if ( ! mfrc522.PICC_IsNewCardPresent() ) {
      return;
    }
    // Select one of the cards
    if ( ! mfrc522.PICC_ReadCardSerial() ) {
      return;
    }
  }
}

```

```

BTserial.print("UID tag :");
String tag = "";
byte letter;
for (byte j = 0; j < mfrc522.uid.size; j++)
{
  tag.concat(String(mfrc522.uid.uidByte[j] < 0x10 ? " 0" : " "));
  tag.concat(String(mfrc522.uid.uidByte[j], HEX));
  BTserial.print(mfrc522.uid.uidByte[j] < 0x10 ? " 0" : " ");
  BTserial.print(mfrc522.uid.uidByte[j], HEX);
}
BTserial.println();
BTserial.print("Message : ");

tag.toUpperCase();
//Checking the card
if (tag.substring(1) == tagUID)
{
  // If UID of tag is matched.
  lcd.clear();
  lcd.print("Tag Matched");
  digitalWrite(greenLed, HIGH);
  delay(3000);
  digitalWrite(greenLed, LOW);
  lcd.clear();
}

```

```

    lcd.print("Enter Password:");
    lcd.setCursor(0, 1);
    RFIDMode = false; // Make RFID mode false
}
else
{
    // If UID of tag is not matched.
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Wrong Tag Shown");
    lcd.setCursor(0, 1);
    lcd.print("Access Denied");
    digitalWrite(buzzerPin, HIGH);
    digitalWrite(redLed, HIGH);
    delay(3000);
    digitalWrite(buzzerPin, LOW);
    digitalWrite(redLed, LOW);
    lcd.clear();
}
}
// If RFID mode is false, it will look for keys from keypad
if (RFIDMode == false) {
    key_pressed = keypad_key.getKey(); // Storing keys
    if (key_pressed)
    {
        password[i++] = key_pressed; // Storing in password variable
    }
}

```

```

    lcd.print("Pass Accepted");
    sg90.write(90); // Door Opened
    digitalWrite(greenLed, HIGH);
    delay(3000);
    digitalWrite(greenLed, LOW);
    BTserial.println("Authorized access");
    BTserial.println();
    sg90.write(0); // Door Closed
    lcd.clear();
    i = 0;
    RFIDMode = true; // Make RFID mode true
}
else // If password is not matched
{
    BTserial.println(" Access denied");
    lcd.clear();
    lcd.print("Wrong Password");
    digitalWrite(buzzerPin, HIGH);
    digitalWrite(redLed, HIGH);
    delay(3000);
    digitalWrite(buzzerPin, LOW);
    digitalWrite(redLed, LOW);
    lcd.clear();
    i = 0;
    RFIDMode = true; // Make RFID mode true
}
}

```

# Security for Intelligent, Connected IoT Edge Nodes

## INTRODUCTION

Internet-of-things (IoT) has emerged as a term to aptly describe the end-to-end platform for sustaining a device-to-Internet-to-device communication model. These large set of electronic devices form the core of an IoT infrastructure, to support every day user activity for a plethora of applications. The evolution of the Internet, attributed primarily to advances in communication bandwidth accompanied with rapid development of diverse user-specific applications, has sustained a progressive march towards introduction of efficient electronic devices to facilitate our daily activity. These set of devices constituting an IoT architecture are tiny (can generally fit into a consumer's pocket with ease), and exhibit features that help facilitate smooth and convenient user/business-activity on the go. Some of these devices include RFID tags, wireless sensors and mobile phones. Through the IoT infrastructure, end-users control applications required for: remote transmission of emails, blood pressure monitoring and transmission to a remote healthcare facility and remote transmission of the geographical position of a truck carrying goods to its destination. Moreover, rapid advances in processing and communication capabilities of these small devices over the last decade, has made communication over longer distances with imposed real-time constraints, realisable. The interaction of these IoT devices with legacy systems requires efficient and secure usage of the Internet communication infrastructure. As a result, not only can IoT devices establish and use remote communication to convey a status report of a particular event or activity, but can also provide end-users with a higher level of confidence in the privacy and authenticity of all services provided therein. The level and procedure to provide privacy and security to an IoT infrastructure will vary in diversity, description, as well as complexity, depending upon the IoT devices in use. For instance, a mobile phone connected through a 3G communication channel to the Internet will implement a different set of protocols for securing the channel, as opposed to a resource-constrained RFID tag, that will invariably have a customised version of a resource-demanding secret-key verifier.

# Tips to secure those IoT devices

There are 5.5 million new things getting connected every day in 2016, as we head toward more than 20 billion by 2020, according to [Gartner](#). That's an awful lot of devices. They might bring all sorts of handy new features, but, whether it's the latest cutting-edge baby monitor or a wireless doorbell camera that links to your phone, it's also a network-connected computer and should be treated as such. Here are eight tips to help you secure those IoT devices.

## 1. Don't connect your devices unless you need to

The first step is to consider what functionality you need from the device. Just because your TV or fridge can connect to the internet, doesn't mean you definitely want to hook it up. Take a good look at the features it offers and learn exactly what internet connectivity brings before you connect.

## 2. Create a separate network

Many Wi-Fi routers support guest networking so that visitors can connect to your network without gaining access to shared files or networked devices. This kind of separation also works well for IoT devices that have questionable security.

## 3. Pick good passwords and a different password for every device


It's very important to pick strong passwords, but you must also make sure that you pick a different password for every device. If a hacker manages to get one of your passwords, they will typically try it with other services and devices. Reusing passwords is not a good idea. Use a password manager to keep track of all your passwords.

## 4. Turn off Universal Plug and Play (UPnP)

Sadly, UPnP can make routers, printers, cameras and other devices vulnerable to attack. It's designed to make it easier to network devices without configuration by helping them automatically discover each other. The problem is that hackers can also potentially discover them from beyond your local network because of vulnerabilities in the UPnP protocol. Is best to turn UPnP off completely.

## 5. Make sure you have the latest firmware

If you want to make sure you have the latest security patches and reduce the chances of a successful attack, then you need to keep your firmware fully updated. Vulnerabilities and exploits will be fixed as



they emerge, so your IoT devices and your router need to be regularly updated. Automate this wherever possible or set a schedule to check for updates every three months or so.

## **6. Be wary of cloud services**

A lot of IoT devices rely on cloud services, but the requirement for an internet connection in order for something to function can be a real problem. Not only will it not work when the network is down, but it may also be syncing sensitive data or offering another potential route into your home. Make sure you read up on the provider's privacy policy and look for reassurances about encryption and data protection.

## **7. Keep personal devices out of the workplace**

Don't take your personal IoT devices to work. There are lots of [potential security concerns for wearables](#). Every enterprise should have a clear BYOD policy, and it's often a good idea to prohibit personal IoT devices from connecting to the network, or at least limit them to a guest network.

## **8. Track and assess devices**

Businesses need to track everything connected to the network and monitor the flow of traffic. Devices need to be assessed to determine the level of access they should have, to keep them fully patched and up to date, and to protect data end-to-end to preserve its integrity. Unknown devices should flag an alert. Understanding which devices are connected and what they're doing is a prerequisite for proper security.

If you're dealing with sensitive data or you're concerned about privacy, then make sure you have a long hard look at the IoT devices you're considering. What security protocols do they support? How easy are they to patch? Do the providers have a proper privacy policy? It's not safe to assume they're secure because all too often they simply aren't.



## ***Conclusion and future scope***

The project has proposed the idea of smart homes that can support a lot of home automation systems. A smart home contains a connection between wireless communication, sensors, monitoring and tracking. Smart homes are a huge system that includes multiple technologies and applications that can be used to provide security and control of the home easily.

This project discussed the designed modules like sensors' circuits, monitoring and tracking of the home through IP camera, mobile notifications and home navigator.

A series of experiments have been carried out on the proposed smart home. These experiments show how to detect the fire, water leaking, smoke. Also how to detect any intruder to the home, detect and control the weather of the any room and how to secure the home through an access code. In addition this project illustrate the way to monitoring and tracking the home through an IP camera, and the way to send notifications to the homeowner about the actions in the home. Also this project showed the idea of making a navigator in the home to measuring the temperature in all rooms and detect any fire happens and to detect any motion in the home by using ultrasonic sensors.

## ***Future work***

There are a variety of enhancements that could be made to this system to achieve greater accuracy in sensing and detection.

1. a) There are a lot of other sensors that can be used to increase the security and control of the home like pressure sensor that can be put outside the home to detect that someone will enter the home.
2. b) Changing the way of the automated notifications by using the GSM module to make this system more professional.
3. c) A smart garage that can measure the length of the car and choose which block to put the car into it and it will navigate the car through the garage to make the parking easy for the homeowner in his garage.

## REFERENCES

- [1] [http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8994-Security-for-Intelligent-Connected-IoT-Edge-Nodes\\_Whitepaper.pdf](http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8994-Security-for-Intelligent-Connected-IoT-Edge-Nodes_Whitepaper.pdf)
  
- [2] <https://core.ac.uk/download/pdf/41537076.pdf>
  
- [3] <https://electronics hobbyists.com/rfid-basics-and-rfid-module-interfacing-with-arduino/?fbclid=IwAR0dKg85QeIDhem23eaSjPPcaW0NUMo7iU61rHHLnOI2Cu7RzvHRTpLaIXA>
  
- [4] <https://www.hackster.io/sakshambhutani2001/octopod-smart-iot-home-industry-automation-project-fa939b?fbclid=IwAR1C6sDL DobT5rm--X0ULX8INZz64qeP9uvZLO-sB9aUgLf0i04ICVO3a0M>
  
- [5] <http://smarthomes-technology.blogspot.com/p/conclusion-and-additional-work.html>
  
- [6] <https://www.the-ambient.com/features/visions-through-the-ages-history-of-home-automation-178>