



*Hacettepe University*

*Computer Engineering Department*

Wireless and Mobil Networks lab.

**Professor**

**Suat Özdemir**

**Home automation system using an Arduino**

AYOUB OTHMAN

21202955

VUGAR KARIMOV

21305599

## ABSTRACT

The main objective of this project is to develop a home automation system using an Arduino board with Bluetooth being remotely controlled by any Android OS smart phone. As technology is advancing so houses are also getting smarter. Modern houses are gradually shifting from conventional switches to centralized control system, involving remote controlled switches. Presently, conventional wall switches located in different parts of the house makes it difficult for the user to go near them to operate. Even more it becomes more difficult for the elderly or physically handicapped people to do so. Remote controlled home automation system provides a most modern solution with smart phones. In order to achieve this, a Bluetooth module is interfaced to the Arduino board at the receiver end while on the transmitter end, a GUI application on the cell phone sends ON/OFF commands to the receiver where loads are connected. By touching the specified location on the GUI, the loads can be turned ON/OFF remotely through this technology.

# INTRODUCTION

Nowadays, we have remote controls for our television sets and other electronic systems, which have made our lives real easy. Have you ever wondered about home automation which would give the facility of controlling tube lights, fans and other electrical appliances at home using a remote control? Off-course, Yes! But, are the available options cost-effective? If the answer is No, we have found a solution to it. We have come up with a new system called Arduino based home automation using Bluetooth. This system is super-cost effective and can give the user, the ability to control any electronic device without even spending for a remote control. This project helps the user to control all the electronic devices using his/her smartphone. Time is a very valuable thing. Everybody wants to save time as much as they can. New technologies are being introduced to save our time. To save people's time we are introducing Home Automation system using Bluetooth . With the help of this system you can control your home appliances from your mobile phone. You can turn on/off your home appliances within the range of Bluetooth.

## COMPONENTS REQUIRED

- 1) ARDUINO UNO
- 2) CHANNEL RELAY(5v)
- 3) BLUETOOTH MODULE HC05
- 4) POWER SUPPLY
- 5) CONNECTING WIRES
- 6) VERO BOARD
- 7) SMARTPHONE (BLUETOOTH ENABLED)

# DESCRIPTION

## ARDUINO UNO

### Features of the Arduino UNO:

Microcontroller: ATmega328

Operating Voltage: 5V

Input Voltage (recommended): 7-12V

Input Voltage (limits): 6-20V

Digital I/O Pins: 14 (of which 6 provide PWM output)

Analog Input Pins: 6

DC Current per I/O Pin: 40 mA

DC Current for 3.3V Pin: 50 mA

Flash Memory: 32 KB of which 0.5 KB used by bootloader

SRAM: 2 KB (ATmega328)

EEPROM: 1 KB (ATmega328)

Clock Speed: 16 MHz

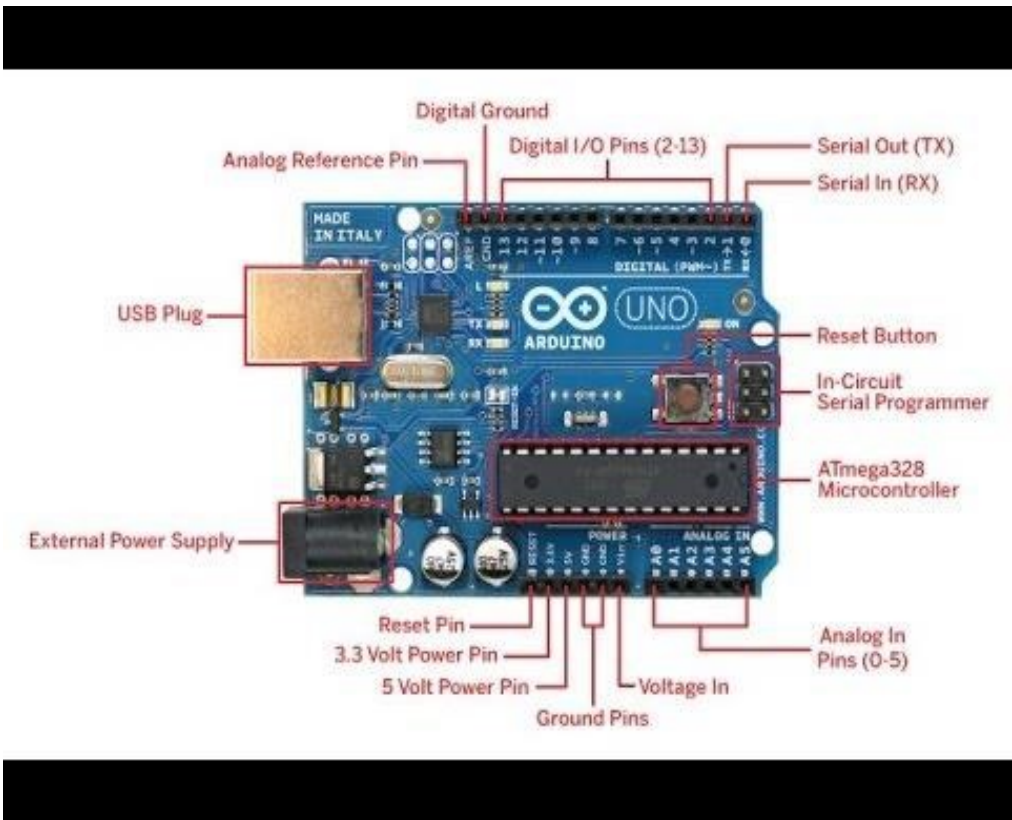


FIG 1 ARDUINO UNO

## ARDUINO HARDWARE PART:-

Arduino is open-source hardware. The hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are available on the Arduino website. Layout and production files for some versions of the hardware are also available.

Although the hardware and software designs are freely available under copy left licenses, the developers have requested the name *Arduino* to be exclusive to the official product and not be used for derived works without permission. The official policy document on use of the Arduino name emphasizes that the project is open to incorporating work by others into the official product. Several Arduino-compatible products commercially released have avoided the project name by using various names ending in *-duino*.

Most Arduino boards consist of an Atmel 8-bit AVR microcontroller (ATmega8, ATmega168, ATmega328, ATmega1280, ATmega2560) with varying amounts of flash memory, pins, and features. The 32-bit Arduino Due, based on the Atmel SAM3X8E was introduced in 2012. The boards use single or double-row pins or female headers that facilitate connections for programming and incorporation into other circuits. These may connect with add-on modules termed *shields*. Multiple and possibly stacked shields may be individually addressable via an I<sup>2</sup>C serial bus. Most boards include a 5 V linear regulator and a 16 MHz crystal oscillator or ceramic resonator. Some designs, such as the LilyPad, run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions.

Arduino microcontrollers are pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory. The default bootloader of the Arduino UNO is the optiboot bootloader. Boards are loaded with program code via a serial connection to another computer. Some serial Arduino boards contain a level shifter circuit to convert between RS232 logic levels and transistor–transistor logic(TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232. Some boards, such as later-model Uno boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods. When used with traditional microcontroller tools, instead of the Arduino IDE, standard AVR in-system programming (ISP) programming is used.

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The *Diecimila*, *Duemilanove*, and current *Uno* provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs, which can also be used as six digital I/O

pins. These pins are on the top of the board, via female 0.1-inch (2.54 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board that can plug into solderless breadboards.

## ARDUINO SOFTWARE PART:-

### IDE

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It originated from the IDE for the languages *Processing* and *Wiring*. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple *one-click* mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.



**RELAY:**

A relay is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuit by a separate low-power signal, or where several circuits must be controlled by one signal. The first relays were used in long distance telegraph circuits as amplifiers: they repeated the signal coming in from one circuit and re-transmitted it on another circuit. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

A type of relay that can handle the high power required to directly control an electric motor or other loads is called a contactor. Solid-state relays control power circuits with no moving parts, instead using a semiconductor device to perform switching. Relays with calibrated operating characteristics and sometimes multiple operating coils are used to protect electrical circuits from overload or faults; in modern electric power systems these functions are performed by digital instruments still called "protective relays".

Magnetic latching relays require one pulse of coil power to move their contacts in one direction, and another, redirected pulse to move them back. Repeated pulses from the same input have no effect. Magnetic latching relays are useful in applications where interrupted power should not be able to transition the contacts.

Magnetic latching relays can have either single or dual coils. On a single coil device, the relay will operate in one direction when power is applied with one polarity, and will reset when the polarity is reversed. On a dual coil device, when polarized voltage is applied to the reset coil the contacts will transition. AC controlled magnetic latch relays have single coils that employ steering diodes to differentiate between operate and reset commands.

**2 channel relay features:**

Number of Relays: 2

Control signal: TTL level

Rated load: 7A/240VAC 10A/125VAC 10A/28VDC

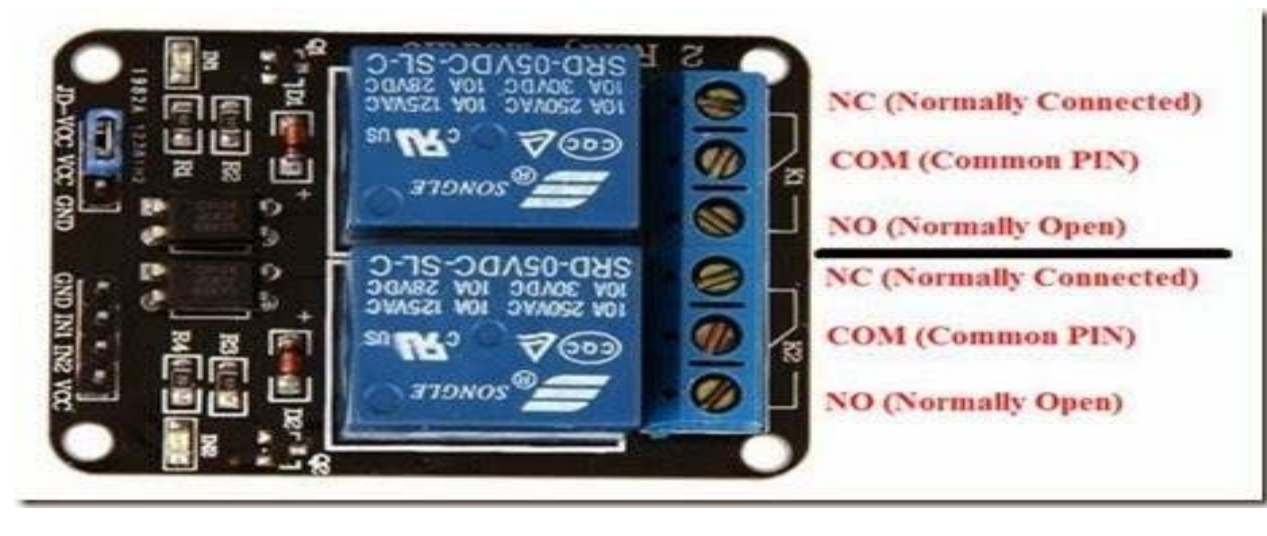
Contact action time: 10ms/5ms

## RELAY H152S MODULE

### Overview

We can control high voltage electronic devices using relays. A Relay is actually a switch which is electrically operated by an electromagnet. The electromagnet is activated with a low voltage, for example 5 volts from a microcontroller and it pulls a contact to make or break a high voltage circuit.

**FIG 2: 2 CHANNEL RELAY**



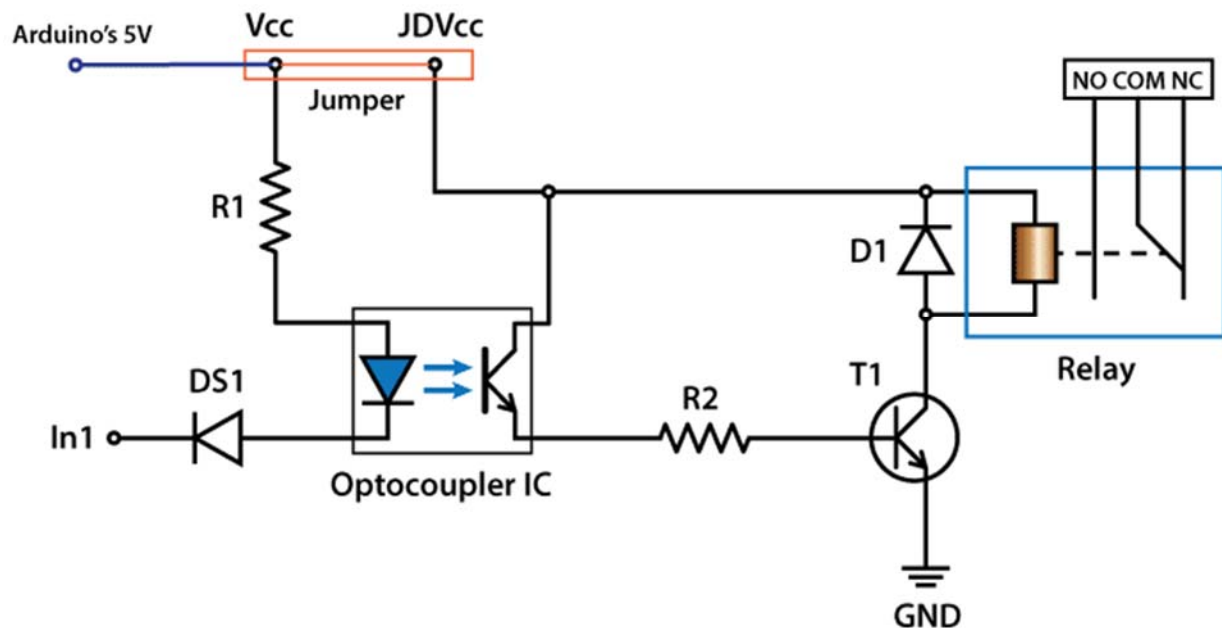
As an example for this Arduino Relay Tutorial we will use the HL-52S 2 channel relay module, which has 2 relays with rating of 10A @ 250 and 125 V AC and 10A @ 30 and 28 V DC. The high voltage output connector has 3 pins, the middle one is the common pin and as we can see from the markings one of the two other pins is for normally open connection and the other one for normally closed connection.

On the other side of the module we have these 2 sets of pins. The first one has 4 pins, a Ground and a VCC pin for powering the module and 2 input pins In1 and In2. The second set of pins has 3 pins with a jumper between the JDVcc and the Vcc pin. With a configuration like this the

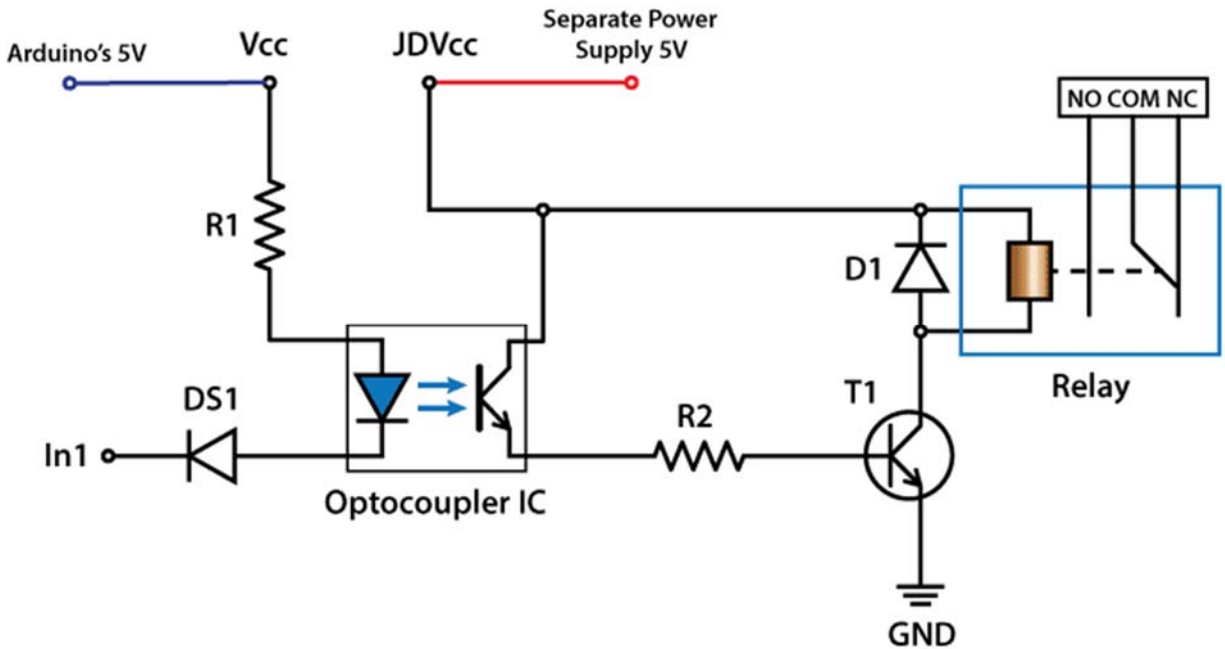
electromagnet of the relay is directly powered from the Arduino Board and if something goes wrong with the relay the microcontroller could get damaged.

### Circuit Schematic

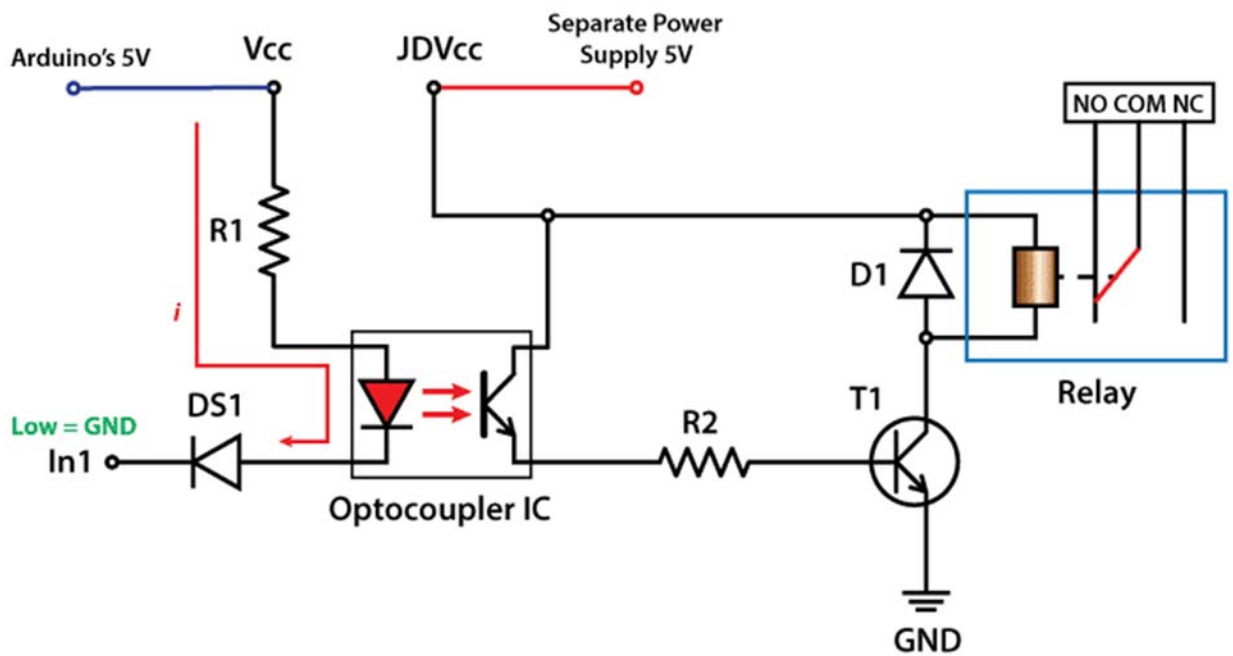
For better understanding let's see the circuit schematics of the relay module in this configuration. So we can see that the 5 volts from our microcontroller connected to the Vcc pin for activating the relay through the Optocoupler IC are also connected to the JDVcc pin which powers the electromagnet of the relay. So in this case we got no isolation between the relay and the microcontroller.



In order to isolate the microcontroller from the relay, we need to remove the jumper and connect separate power supply for the electromagnet to the JDVcc and the Ground pin. Now with this configuration the microcontroller doesn't have any physical connection with the relay, it just uses the LED light of the Optocoupler IC to activate the relay.



There is one more thing to be noticed from this circuit schematic. The input pins of the module work inversely. As we can see the relay will be activated when the input pin will be LOW because in that way the current will be able to flow from the VCC to the input pin which is low or ground, and the LED will light up and active the relay. When the input pin will be HIGH there will be no current flow, so the LED will not light up and the relay will not be activated.



First let's take a look at the circuit diagram. As previously described we will use a 5V Adapter as a separate power supply for the electromagnet connected to the JDVcc and the Ground pin. The Arduino's 5V pin will be connected to the Vcc pin of the module and the pin number 7 to the In1 input pin for controlling the relay. Now for the HIGH Voltage part we need a power plug, a socket and a cable with two wires. One of the two wires will be cut and connected to the common and the normally open pin of the module output connector. So with this configuration when we will activate the relay we will get the high voltage circuit closed and working.

Here's how made the cable. So I bought a plug, a socket and a cable. Then I carefully cut the cable and cut one of the wires as shown in the picture below and connect them to the normally open connection pins of the relay module. Also connected the ends of the cable to the plug and the socket.



## **BLUETOOTH MODULE(HC-05 Bluetooth Module)**

HC-05 Specification:

Bluetooth protocol: Bluetooth Specification v2.0+EDR

Frequency: 2.4GHz ISM band

Modulation: GFSK (Gaussian Frequency Shift Keying)

Emission power:  $\leq 4\text{dBm}$ , Class 2

Sensitivity:  $\leq -84\text{dBm}$  at 0.1% BER

Speed: Asynchronous: 2.1Mbps(Max) / 160 kbps, Synchronous: 1Mbps/1Mbps

Security: Authentication and encryption

Profiles: Bluetooth serial port

Power supply: +3.3VDC 50mA

Working temperature:  $-20 \sim +75\text{Centigrade}$

Dimension: 26.9mm x 13mm x 2.2 mm

### **Overview**

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. The HC-05 Bluetooth Module can be used in a Master or Slave configuration, making it a great solution for wireless communication. This serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04 External single chip Bluetooth system with CMOS technology and with AFH (Adaptive Frequency Hopping Feature).

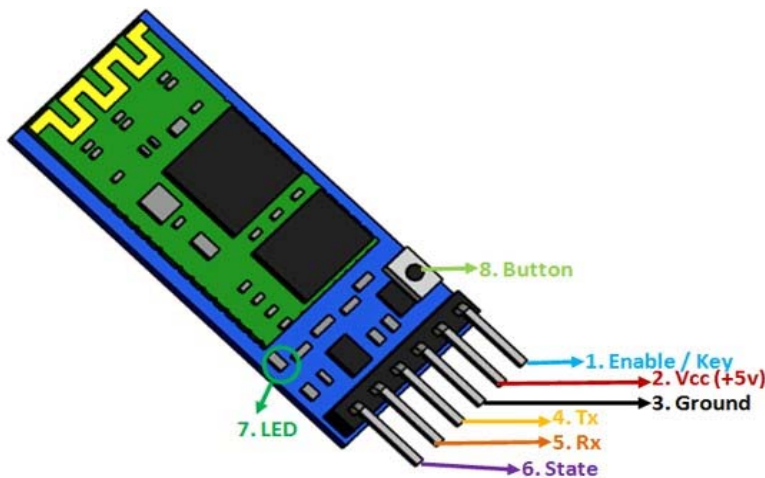
Bluetooth Module HC-05

The Bluetooth module HC-05 is a MASTER/SLAVE module. By default, the factory setting is SLAVE. The Role of the module (Master or Slave) can be configured only by AT

COMMANDS. The slave modules cannot initiate a connection to another Bluetooth device, but can accept connections. Master module can initiate a connection to other devices. The user can use it simply for a serial port replacement to establish connection between MCU and GPS, PC to your embedded project, etc.



## BLUETOOTH MODULE



## PIN DESCRIPTION OF BLUETOOTH MODULE

## Pin Description: -

The HC-05 Bluetooth Module has 6pins. They are as follows:

### ENABLE:

When enable is pulled LOW, the module is disabled which means the module will not turn on and it fails to communicate. When enable is left open or connected to 3.3V, the module is enabled i.e. the module remains on and communication also takes place.

### Vcc:

Supply Voltage 3.3V to 5V

### GND:

Ground pin

### TXD & RXD:

These two pins acts as an UART interface for communication

### STATE:

It acts as a status indicator. When the module is not connected to paired with any other Bluetooth device, signal goes Low. At this low state, the led flashes continuously which denotes that the module is not paired with other device. When this module is connected to/paired with any other Bluetooth device, the signal goes High. At this high state, the led blinks with a constant delay say for example 2s delay which indicates that the module is paired.

### BUTTON SWITCH:

This is used to switch the module into AT command mode. To enable AT command mode, press the button switch for a second. With the help of AT commands, the user can change the



parameters of this module but only when the module is not paired with any other BT device. If the module is connected to any other Bluetooth device, it starts to communicate with that device and fails to work in AT command mode.

#### HC-05 Default Settings: -

Default Bluetooth Name: „HC-05“

Default Password: 1234 or 0000

Default Communication: Slave

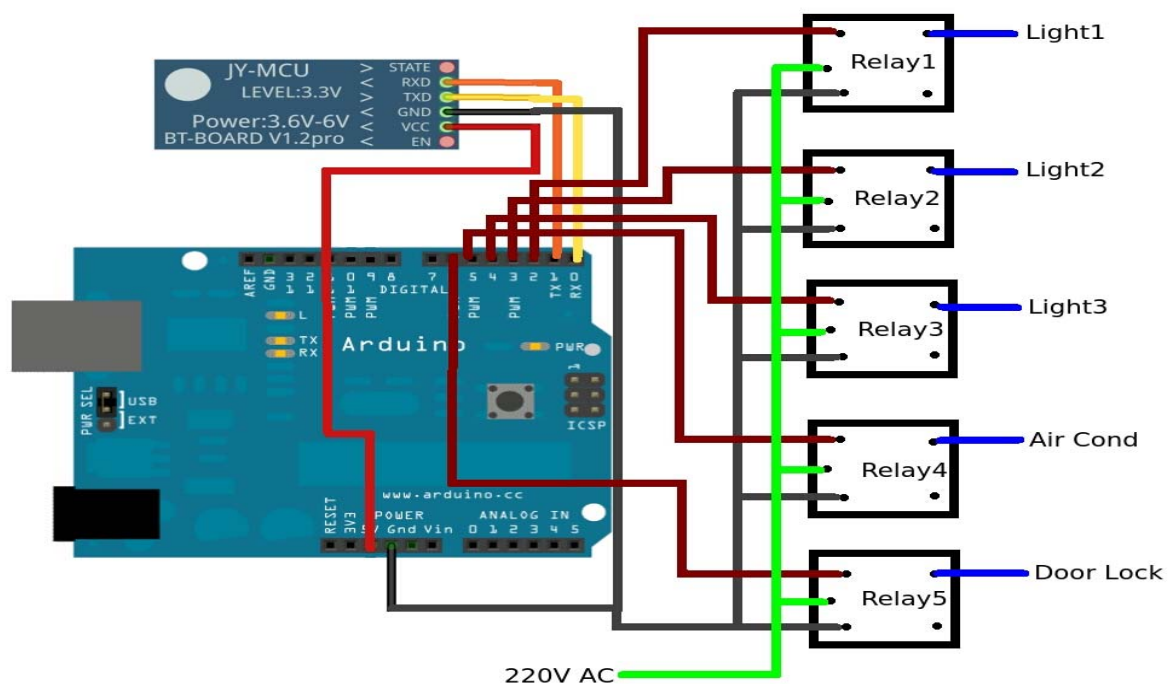
Default Mode: Data Mode

Data Mode Baud Rate: 9600, 8, N, 1

Command Mode Baud Rate: 38400, 8, N, 1

Default firmware: LINVOR

## CIRCUIT DIAGRAM



# CODE

```
#include <SoftwareSerial.h>
SoftwareSerial BT(10, 11); //TX, RX pins of arduino respectively
String command;
void setup()
{
  BT.begin(9600);
  Serial.begin(9600);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
}

void loop() {
  while (BT.available()){ //Check if there is an available byte to read
    delay(10); //Delay added to make thing stable
    char c = BT.read(); //Conduct a serial read
    command += c; //build the string.
  }
  if (command.length() > 0) {
    Serial.println(command);
    if(command == "light on") //this command will be given as an input to switch on light1
    {
      digitalWrite(2, HIGH);
    }
    else if(command == "light off") //this command will be given as an input to switch off light1 similarly other commands work
    {
      digitalWrite(2, LOW);
    }
    else if (command == "lamp on")
    {
      digitalWrite (3, HIGH);
    }
    else if ( command == "lamp off")
    {
      digitalWrite (3, LOW);
    }
    else if (command == "fan on")
    {
      digitalWrite (4, HIGH);
    }
    else if (command == "fan of")
```

```

{
  digitalWrite (4, LOW);
}
else if (command == "all on") //using this command you can switch on all devices
{
  digitalWrite (2, HIGH);
  digitalWrite (3, HIGH);
  digitalWrite (4, HIGH);
}
else if (command == "off")//using this command you can switch off all devices
{
  digitalWrite (2, LOW);
  digitalWrite (3, LOW);
  digitalWrite (4, LOW);
}
command=""; //Reset the variable

```

## APPLICATIONS

### **Lighting Control**

Smart lighting allows you to control wall switches, blinds, and lamps, but how intuitive is a lighting control system? It turns out, quite; its capabilities are extensive. You're able to schedule the times lights should turn on and off, decide which specific rooms should be illuminated at certain times, select the level of light which should be emitted, and choose how particular lights react through motion sensitivity, as seen with Belkin's WeMo Switch + Motion, which is both affordable and easy to use with its plug-and-play simplicity. But in the project that we present to you, the lighting is controlled through an application that works on the Android smartphone.

## REFERENCE

1. Wikipedia
2. Wireless Sensor Networks: Concepts, Applications, Experimentation and Analysis. 2016.
3. Home Automation & Wiring (1 ed.).