CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUTION

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 smart phone. 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.

The recent developments in technology which permit the use of wireless controlling environments like, Bluetooth and Wi-Fi that have enabled different devices to have capabilities of connecting with each other. Using a WIFI shield to act as a Micro web server for the Arduino which eliminates the need for wired connections between the Arduino board and computer which reduces cost and enables it to work as a stand alone device. The Wi-Fi shield needs connection to the internet from a wireless router or wireless hotspot and this would act as the gateway for the Arduino to communicate with the internet.

1.1 Embedded System Implementation

Introduction

An embedded system is one kind of a computer system mainly designed to perform several tasks like to access, process, and store and also control the data in various electronics-based systems. Embedded systems are a combination of hardware and software where software is usually known as firmware that is embedded into the hardware. One of its most important characteristics of these systems is, it gives the o/p within the time limits. Embedded systems support to make the work more perfect and convenient. So, we frequently use embedded systems in simple and complex devices too. The applications of embedded systems mainly involve in our real life for several devices like microwave, calculators, TV remote control, home security and neighbourhood traffic control systems, etc.

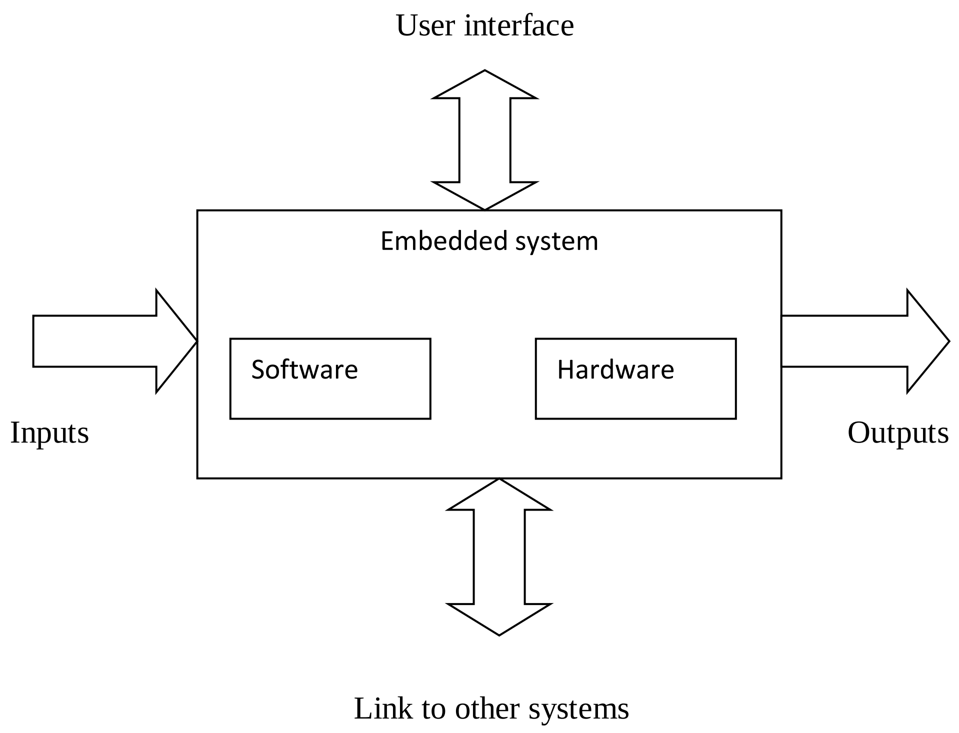


Fig 1.1: Overview of embedded system

Embedded System

Embedded system includes mainly two sections, they are

1. Hardware
2. Software

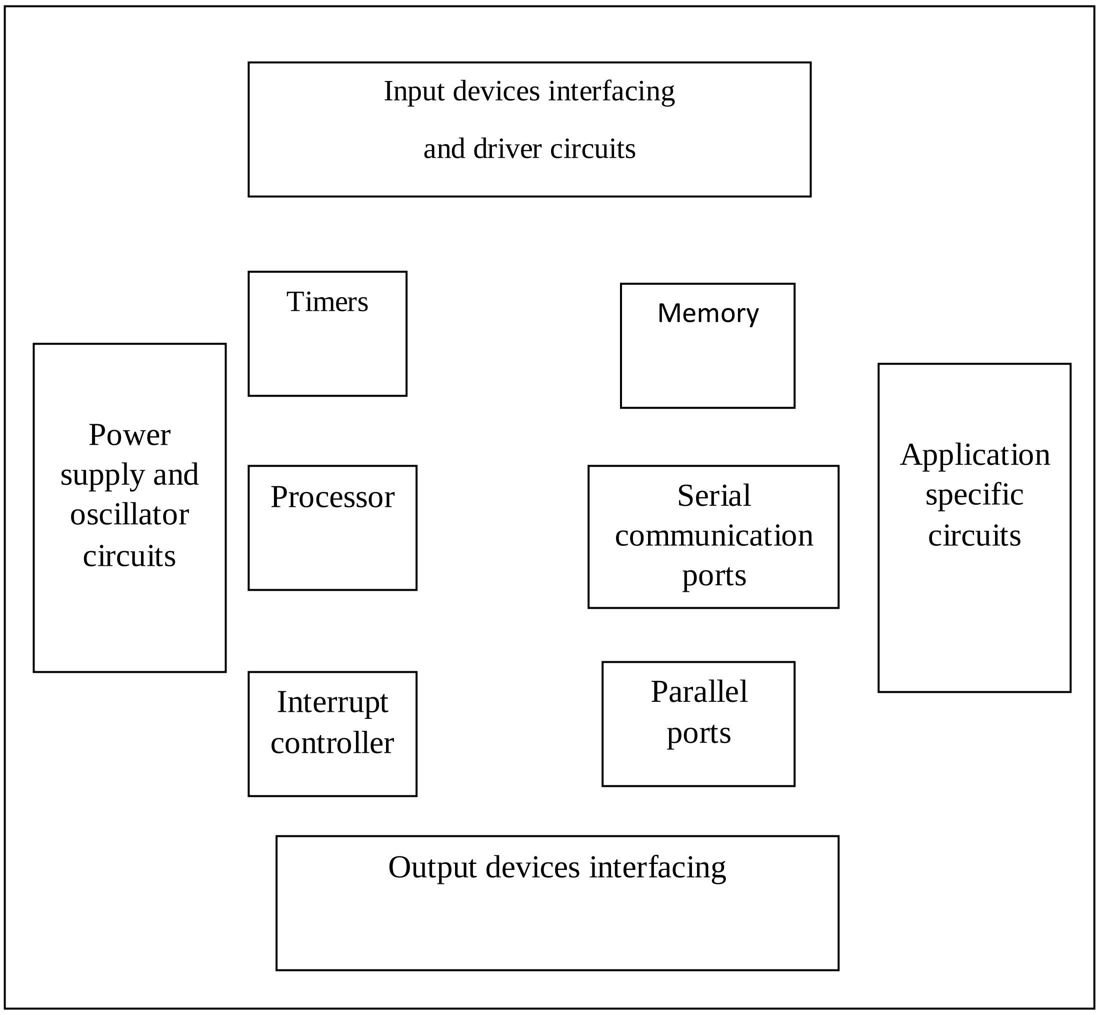

Input devices interfacing and driver circuits
Timers
Memory
Power supply and oscillator circuits
Processor
Serial communication ports
Application specific circuits
Interrupt controller
Parallel ports
Output devices interfacing

Fig 1.2 : Block diagram of embedded system

1.2 Embedded System Hardware

As with any electronic system, an embedded system requires a hardware platform on which it performs the operation. Embedded system hardware is built with a microprocessor or microcontroller. The embedded system hardware has elements like input output (I/O) interfaces, user interface, memory and the display. Usually, an embedded system consists of:

- Power Supply
- Processor
- Memory
- Timers
- Serial communication ports
- Output/Output circuits
- System application specific circuits

Embedded systems use different processors for its desired operation. Some of the processors used are

1. Microprocessor

2. Microcontroller

3. Digital signal processor

Microprocessor vs. Microcontroller

Microprocessor

- CPU on a chip.
- We can attach required amount of ROM, RAM and I/O ports.
- Expensive due to external peripherals.
- Large in size
- general-purpose

Microcontroller

- Computer on a chip
- fixed amount of on-chip ROM, RAM, I/O ports
- Low cost.
- Compact in size.
- Specific –purpose

1.3 Embedded System Software

The embedded system software is written to perform a specific function. It is typically written in a high level format and then compiled down to provide code that can be lodged within a non-volatile memory within the hardware. An embedded system software is designed to keep in view of the three limits:

- Availability of system memory
- Availability of processor's speed
- When the system runs continuously, there is a need to limit power dissipation for events like stop, run and wake up.

1.4 Bringing Software And Hardware Together For Embedded System

To make software to work with embedded systems we need to bring software and hardware together .for this purpose we need to burn our source code into microprocessor or microcontroller which is a hardware component and which takes care of all operations to be done by embedded system according to our code.

Generally we write source codes for embedded systems in assembly language, but the processors run only executable files.The process of converting the source code representation of your embedded software into an executable binary image involves three distinct steps:

1. Each of the source files must be compiled or assembled into an object file.
2. All of the object files that result from the first step must be linked together to produce a single object file, called the re-locatable program.

3. Physical memory addresses must be assigned to the relative offsets within the re-locatable program in a process called relocation.

The result of the final step is a file containing an executable binary image that is ready to run on the embedded system.

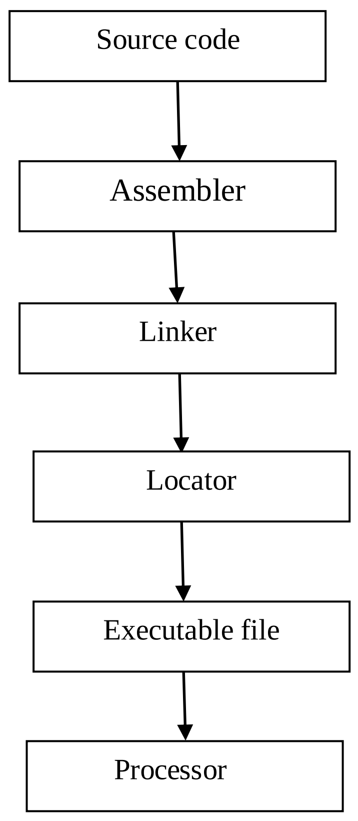


Fig 1.3 :Flow of burning source code to processor

Applications

Embedded systems have different applications. A few select applications of embedded systems are smart cards, telecommunications, satellites, missiles, digital consumer electronics, computer networking, etc.

Embedded Systems in Automobiles

- Motor Control System
- Engine or Body Safety

- Robotics in Assembly Line
- Mobile and E-Com Access

Embedded systems in Telecommunications

- Mobile computing
- Networking
- Wireless Communications

Embedded Systems in Smart Cards

- Banking
- Telephone
- Security Systems

Implementation Flow

Stage 1

Considering the problems of existing methods and giving solution to that problem by considering the basic requirements for our proposed system

Stage 2

Considering the hardware requirement for the proposed system

For this we need to select the below components:

1. Microcontroller

2. Inputs for the proposed system (ex: sensors, drivers etc..,)

3. Outputs (ex: relays, loads)

Stage 3

After considering hardware requirements, now we need to check out the software requirements. Based on the microcontroller we select there exists different software for coding, compiling, debugging. we need to write source code for that proposed system based on our requirements and compile, debug the code in that software .

After completing all the requirements of software and hardware we need to bring both together to work our system.

CHAPTER 2

LITERATURE REVIEW

"Smart Energy Efficient Home Automation System using IOT", by Satyendra K. Vishwakarma, Prashant Upadhyaya, Babita Kumari, Arun Kumar Mishra.

This paper presents a step-by-step procedure of a smart home automation controller. It uses IOT to convert home appliances to smart and intelligent devices, with the help of design control. An energy efficient system is designed that accesses the smart home remotely using IOT connectivity. The proposed system mainly requires, Node MCU as the microcontroller unit, IFTTT to interpret voice commands, Adafruit a library that supports MQTT acts as an MQTT broker and Arduino IDE to code the microcontroller. This multimodal system uses Google Assistant along with a web based application to control the smart home. The smart home is implemented with main controller unit that is connected with the 24-hour available Wi-Fi network. To ensure, that the Wi-Fi connection do not turn off, the main controller is programmed to establish automatic connection with the available network and connected to the auto power backup.

"IOT Based Smart Security and Home Automation", by ShardhaSomani, Parikshit Solunke, ShaunakOke, ParthMedhi, Prof. P. P. Laturkar.

This paper focuses on a system that provides features of Home Automation relying on IOT to operate easily, in addition to that it includes a camera Module and provides home security. The android application basically converts Smartphone into a remote for all home appliances. Security is achieved with motion sensors if movement is sensed at the entrance of the house; a notification is sent that contains a photo of house entrance in real time. This notification will be received by the owner of the house via internet such that app can trigger a notification. So owner can raise an alarm in case of any intrusion or he/she can toggle the appliances like opening the door if the person is a guest. The system uses Raspberry Pi, a small sized computer which acts as server for the system. The smart home consist two modules. Home automation that consists; fan light and door controller, and security module that consists; smoke sensor motion sensor and camera module.

Roshni Bhandari Assistant Professor Computer Engineering Department S.S.Agrawal Institute of Engineering and Technology Navsari, India

In developed and developing countries the more and more technologies are arriving every year. And of the most important is the IOT. IOT can be used in many sectors of the technology. IOT is also used in home automation which helps to make life easy as well as it is time saving. Smartphone Application can be utilize for the IOT communication protocols. Application can be used to monitor and control local switches via mobile phone. Many systems are reported in the literature based on single monitoring and controlling mode utilizing text, voice or gesture commands. The system has two different operation modes first mode make use of a mobile app interface with virtual switches and slider to monitor and control appliances. The second is chat-based that use text or audio command filter with natural language processing to monitor and control the home appliances. The proposed system is scalable in that it is able to add and remove rooms on demand. Most of the technologies which are in the developed are limited as some as feature of turning light and fan on and off while some has control over the gate. There is one technology in which website is use to control the home appliances and in another one there is only mobile application. Different technologies are developed in different way.

2.1 Existing Method

Bluetooth based home automation system

Home automation systems using smartphone, Arduino board and Bluetooth technology are secured and low cost. The Bluetooth system uses a PC or smartphone as receiver device.Bluetooth has a limited range of 10 meters if the smart is out of range then it will not be able to control the home applications, this is one of the main disadvantages of Bluetooth based home automation system.

Voice recognition based home automation

A voice recognition based home automation system proposed by a researcher. The wireless communication between the smartphone and the Arduino UNO is done through Bluetooth technology. The main drawback of this system is that communication betweenuser and voice recognition tool depends on signal to noise ratio, if voice signal is noisy then communication can highly effect and the system will fail to show accuracy.

GSM based home automation system

A smart home automation system implemented by using Global system for Mobile communication (GSM). In GSM based home automation sytems, communication between main module and appliances is done through text messages. The main drawback of GSM based home automation system is that, there is no guarantee text message deliver to the system every time so it is not so reliable system.

2.2 Proposed Method

Proposed system enables secure local IOT server with an IOT App which will control the devices locally in home/ apartment complex. The user can control the lights, fans, AC, security alarms and can read Temperature, humidity based on the need using the BLYNK IOT app installed on their Android Phones. The BLYNK IOT server can be deployed on Raspberry Pie or on a laptop which acts as local cloud for all control / monitoring of the devices. The Home automation was controlled by the wi-fi network in Blynk android application. The wi-fi Home automation can be easily to make on and off by using the command in Blynk application. We can make the Home automation do the various task using wi-fi network technologies.

Block Diagram For Proposed Method

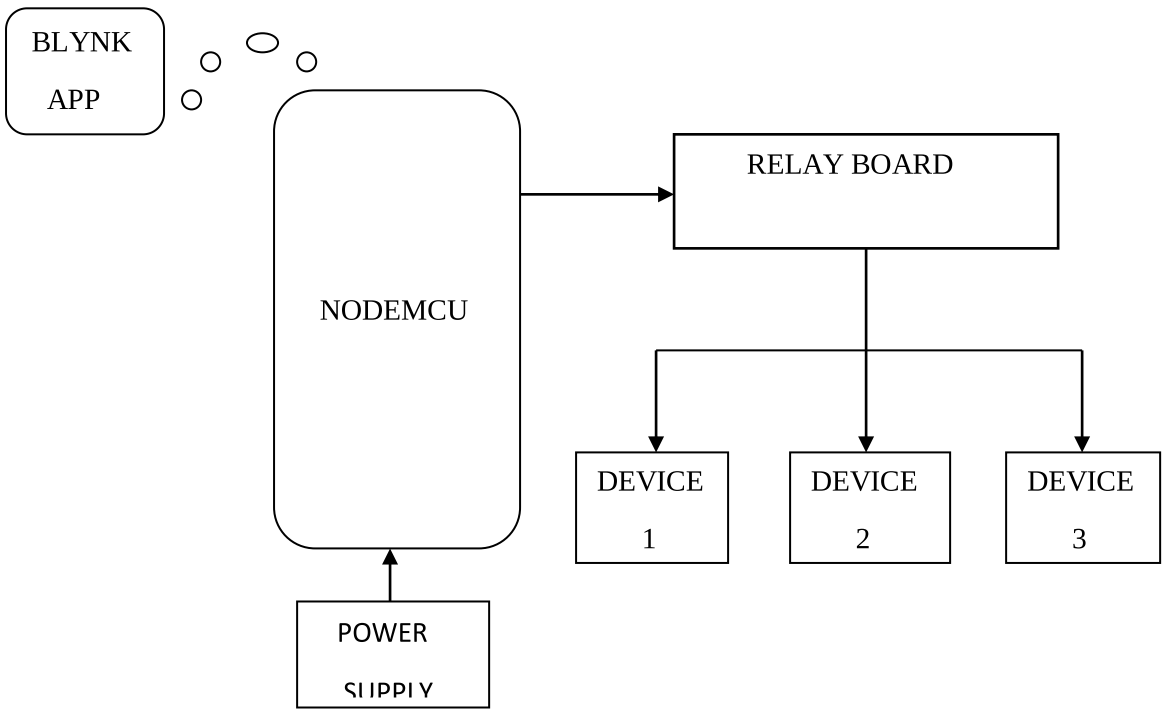


Fig 2.1: Block diagram of proposed method for home automation system

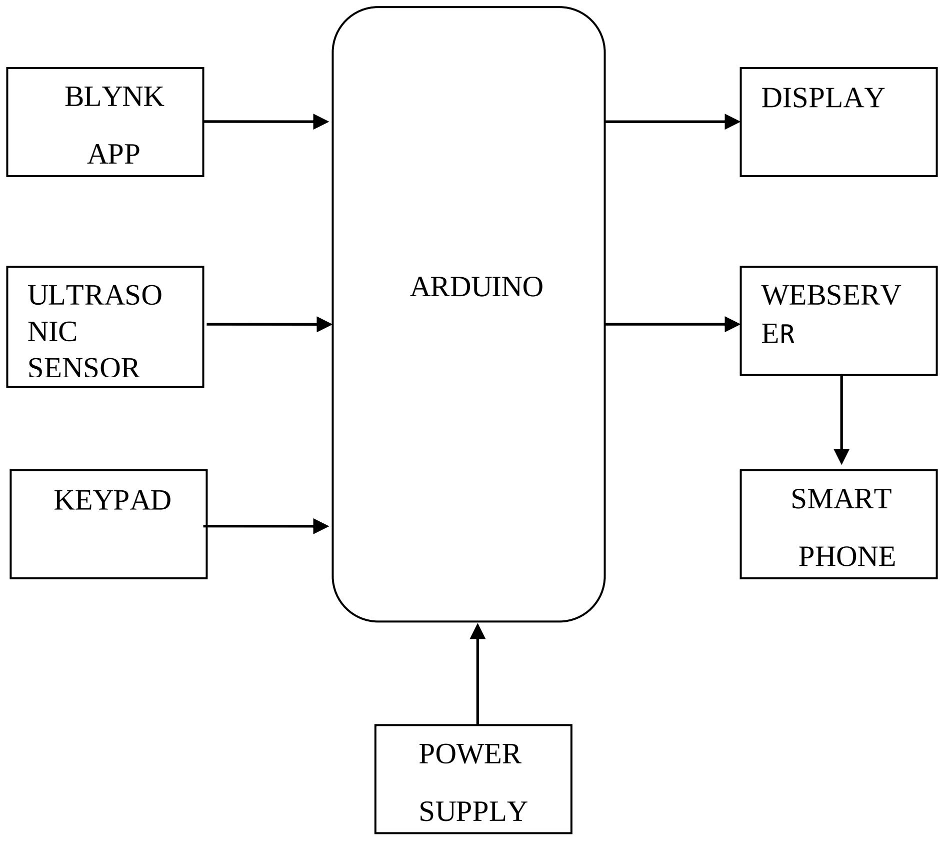


Fig 2.2 : Block diagram of proposed method for home security system

CHAPTER 3
HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements

3.1 Arduino

The Arduino microcontroller is an easy to use yet powerful single board computer that has gained considerable traction in the hobby and professional market. The Arduino is open-source, which means hardware is reasonably priced and development software is free. This guide is for students in ME 2011, or students anywhere who are confronting the Arduino for the first time. For advanced Arduino users, prowl the web; there are lots of resources.

This is what the Arduino board looks like

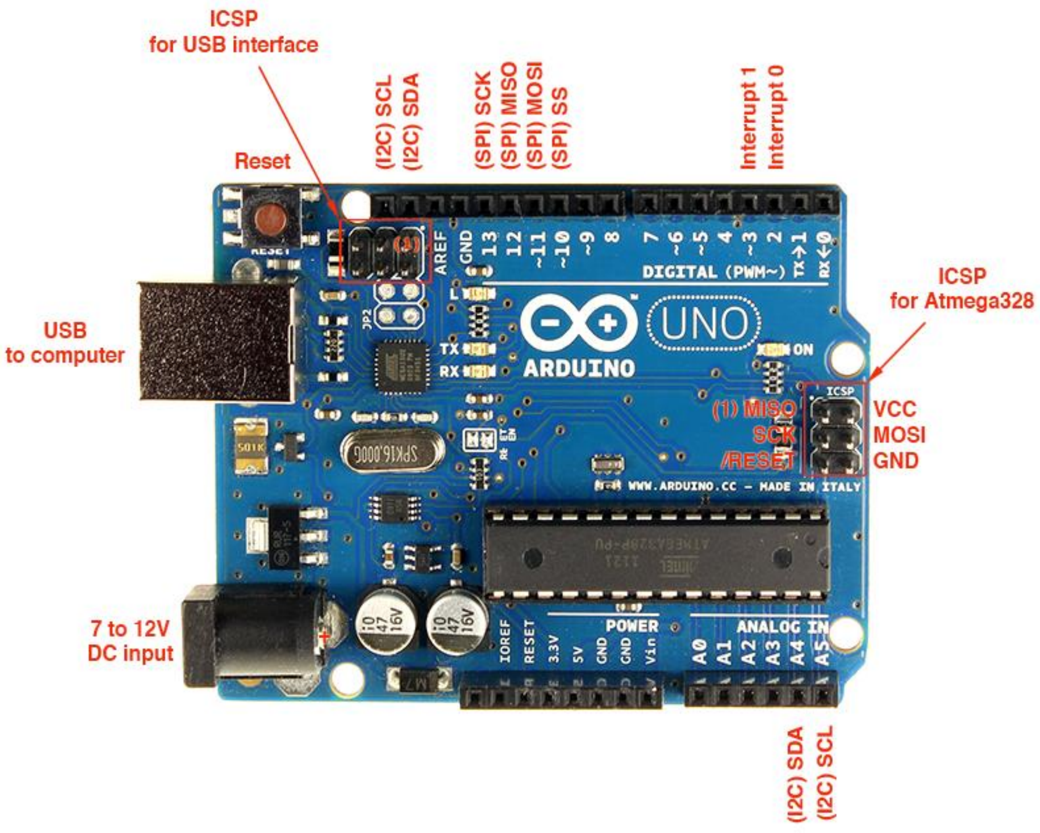


Fig 3.1: Arduino UNO

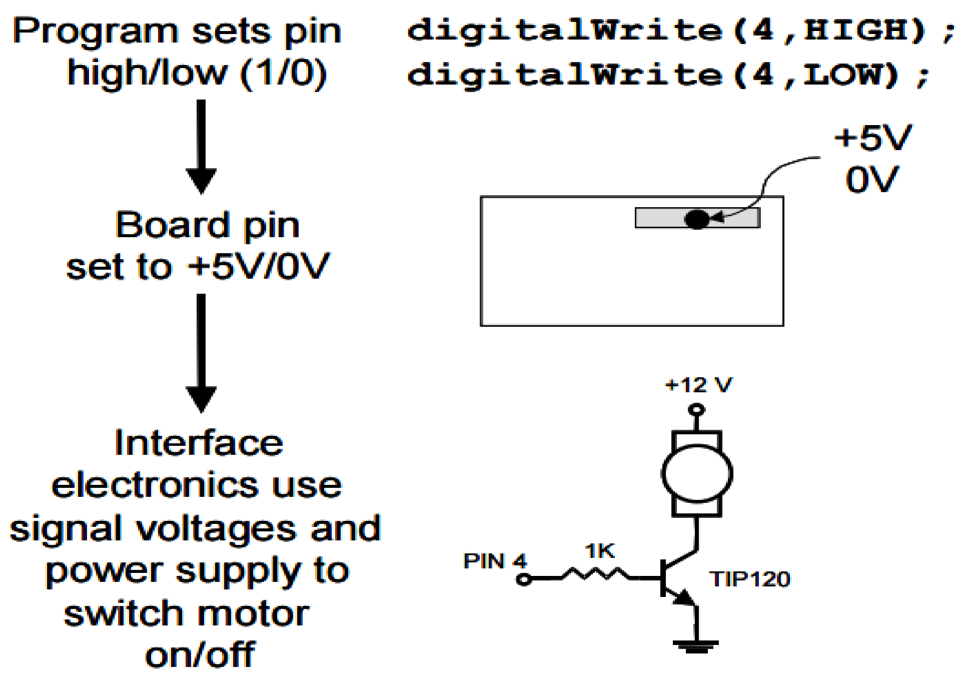
The Arduino programming language is a simplified version of C/C++. If you know C, programming the Arduino will be familiar. If you do not know C, no need to worry as only a few commands are needed to perform useful functions.

Arduino Hardware

The power of the Arduino is not its ability to crunch code, but rather its ability to interact with the outside world through its input-output (I/O) pins. The Arduino has 14 digital I/O pins labeled 0 to 13 that can be used to turn motors and lights on and off and read the state of switches.

Each digital pin can sink or source about 40 mA of current. This is more than adequate for interfacing to most devices, but does mean that interface circuits are needed to control devices other than simple LED's. In other words, you cannot run a motor directly using the current available from an Arduino pin, but rather must have the pin drive an interface circuit that in turn drives the motor. A later section of this document shows how to interface to a small motor.

To interact with the outside world, the program sets digital pins to a high or low value using C code instructions, which corresponds to +5 V or 0 V at the pin. The pin is connected to external interface electronics and then to the device being switched on and off. The sequence of events is shown in this figure.


Program sets pin high/low (1/0)
digitalWrite(4,HIGH);
digitalWrite(4,LOW);
+5V
0V
Board pin set to +5V/0V
Interface electronics use signal voltages and power supply to switch motor on/off
+12 V
PIN 4
1K
TIP120

To determine the state of switches and other sensors, the Arduino is able to read the voltage value applied to its pins as a binary number. The interface circuitry translates the sensor signal into a 0 or +5 V signal applied to the digital I/O pin. Through a program command, the Ardino interrogates the state of the pin. If the pin is at 0 V, the program will read it as a 0 or LOW. If it is at +5 V, the program will read it as a 1 or HIGH. If more than +5 V is applied, you may blow out your board, so be careful. The sequence of events to read a pin is shown in this figure.

Interacting with the world has two sides. First, the designer must create electronic interface circuits that allow motors and other devices to be controlled by a low (1-10 mA) current signal that switches between 0 and 5 V, and other circuits that convert sensor readings into a switched 0 or 5 V signal. Second, the designer must write a program using the set of Arduino commands that set and read the I/O pins. Examples of both can be found in the Arduino resources section of the ME2011 web site.

Atmega328p features:

- High Performance, Low Power AVR® 8-Bit Microcontroller

- Advanced RISC Architecture

  – 131 Powerful Instructions

  – Most Single Clock Cycle Execution

  – 32 x 8 General Purpose Working Registers

  – Fully Static Operation

  – Up to 20 MIPS Throughput at 20 MHz

  – On-chip 2-cycle Multiplier

- High Endurance Non-volatile Memory Segments

  – 4/8/16/32K Bytes of In-System Self-Programmable Flash program Memory (ATmega48PA/88PA/168PA/328P)

  – 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)

  – 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)

  – Write/Erase Cycles: 10,000 Flash/100,000 EEPROM

  – Data retention: 20 years at 85°C/100 years at 25°C(1)

  – Optional Boot Code Section with Independent Lock Bits In-System Programming by On-chip Boot Program True Read-While-Write Operation

  – Programming Lock for Software Security

- Peripheral Features

  – 32 x 8 General Purpose Working Registers

  – Fully Static Operation

  – Up to 20 MIPS Throughput at 20 MHz

  – On-chip 2-cycle Multiplier Peripheral Features

– Two 8-bit Timer/Counters with Separate Prescaler and Compare mode

– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode

– Real Time Counter with Separate Oscillator

– Six PWM Channels – 8-channel 10-bit ADC in TQFP and QFN/ML package Temperature Measurement – 6-channel 10-bit ADC in PDIP package Temperature Measurement

– Programmable Serial USART

– Master/Slave SPI Serial Interface

– Byte-oriented 2-wire Serial Interface (Philips I2 C compatible)

– Programmable Watchdog Timer with Separate On-chip Oscillator

– On-chip Analog Comparator

– Interrupt and Wake-up on Pin Change

- I/O and Packages

– 23 Programmable I/O Lines

– 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF

- Operating Voltage:

– 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P

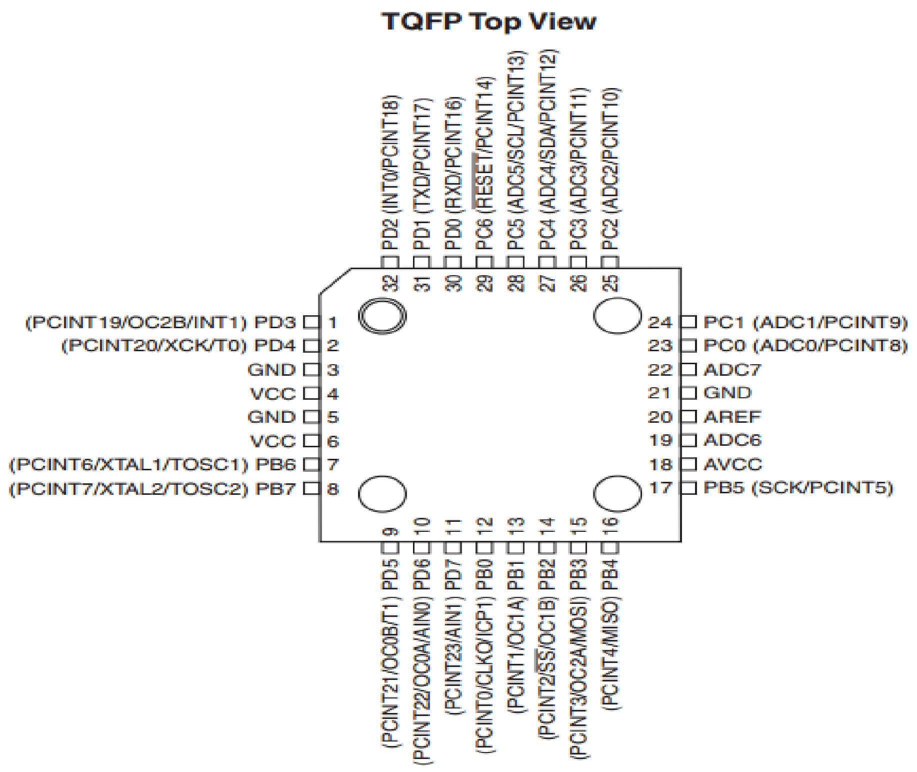
- Temperature Range:

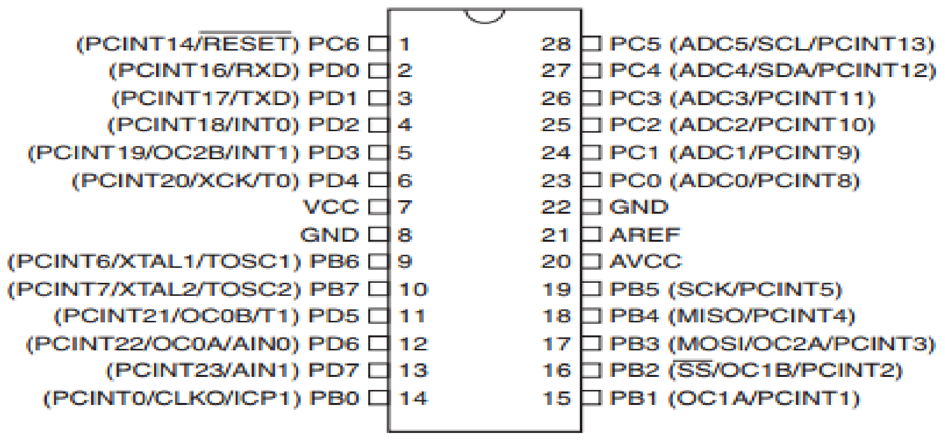
– -40°C to 85°C

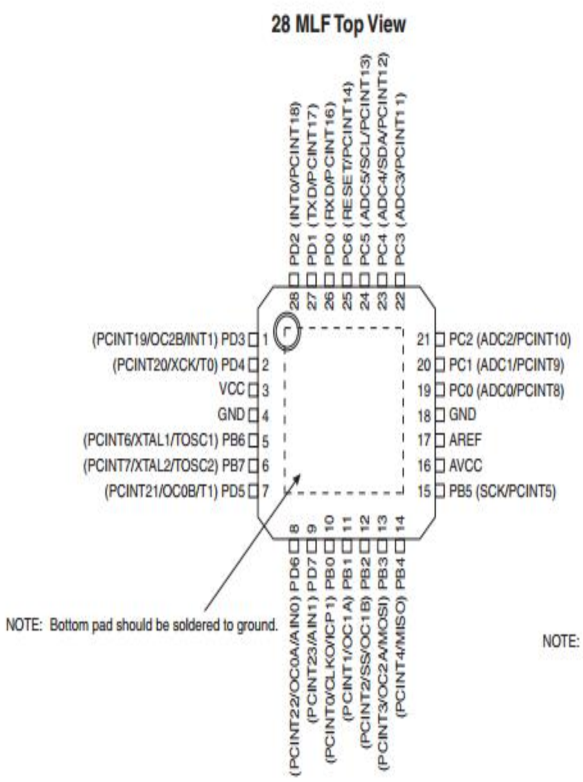
- Speed Grade:

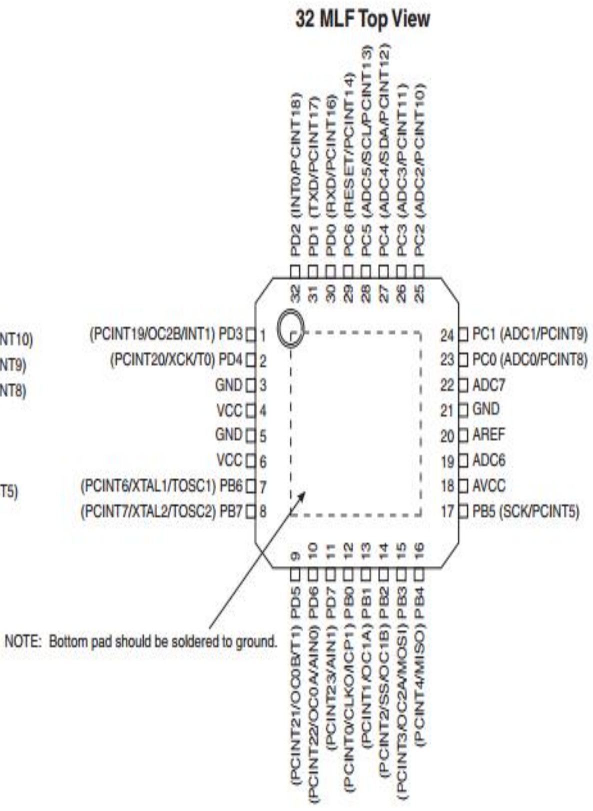
– 0 - 20 MHz @ 1.8 - 5.5V

3.1.1 Pin Configurations









Pin Descriptions

VCC: Digital supply voltage.

GND: Ground.

Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2: Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit).. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Depending on the clock selection fuse settings, PB6 can be used as input to the inverting. Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier. If the Internal Calibrated RC Oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

Port C (PC5:0): Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5..0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated..

PC6/RESET: If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C. If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given in Table 28-3 on page 308. Shorter pulses are not guaranteed to generate a Reset..

Port D (PD7:0): Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

AVCC: AVCC is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. Note that PC6..4 use digital supply voltage, VCC.

AREF: AREF is the analog reference pin for the A/D Converter

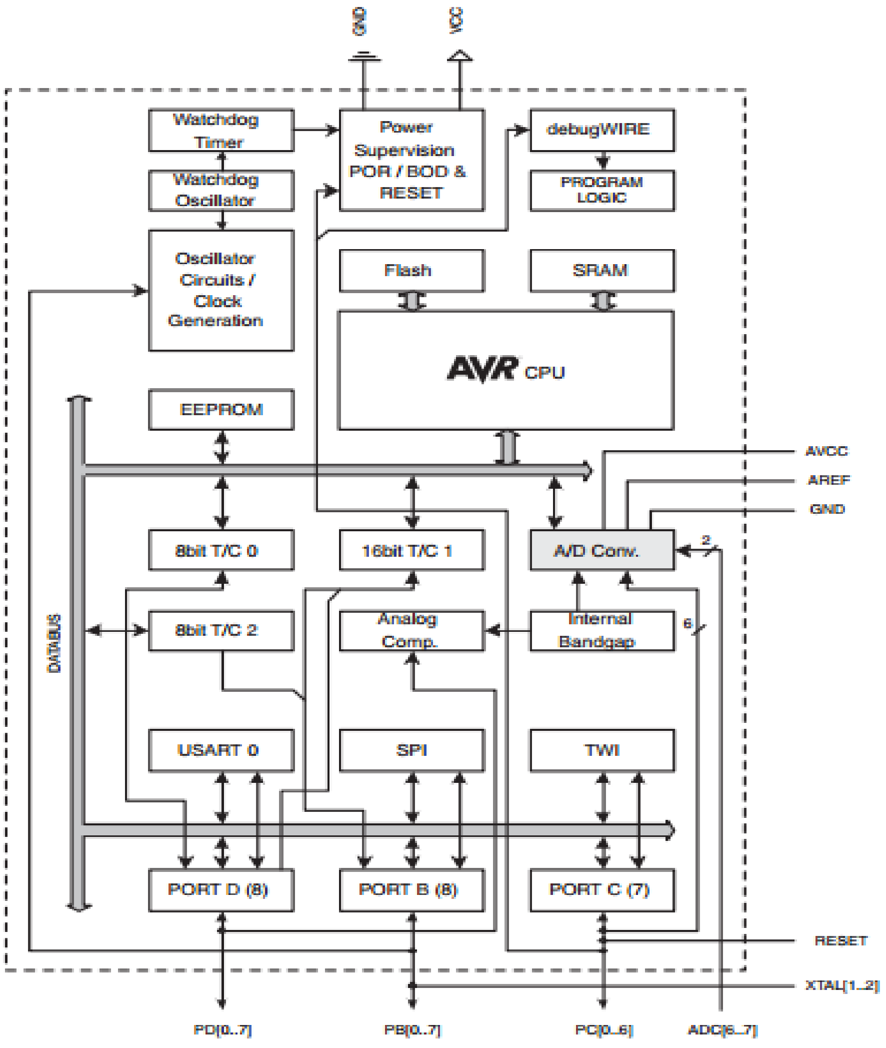
ADC7:6 (TQFP and QFN/MLF Package Only): In the TQFP and QFN/MLF package, ADC7:6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

OVERVIEW

The ATmega48PA/88PA/168PA/328P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega48PA/88PA/168PA/328P achieves throughputs approaching 1 MIPS per MHz allowing the system designed to optimize power consumption versus processing speed.

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

3.1.2 Block Diagram Of Arduino



The ATmega48PA/88PA/168PA/328P provides the following features: 4/8/16/32K bytes of In System Programmable Flash with Read-While-Write capabilities, 256/512/512/1K bytes EEPROM, 512/1K/1K/2K bytes SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte-oriented 2-wire Serial Interface, an SPI serial port, a 6-channel 10-bit ADC (8 channels in TQFP and QFN/MLF packages), a programmable Watchdog Timer with internal Oscillator, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, USART, 2-wire Serial Interface, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions.

The device is manufactured using Atmel's high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip Boot program running on the AVR core. The Boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega48PA/88PA/168PA/328P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

3.2 Power Supply

A power supply is a component that provides at least one electrical charge with power. It typically converts one type of electrical power to another, but it can also convert a different Energy form in electrical energy, such as solar, mechanical, or chemical.

A power supply provides electrical power to components. Usually the term refers to devices built into the powered component.Computer power supplies, for example, convert AC current to DC current and are generally located along with at least one fan at the back of the computer case.

Most computer power supplies also have an input voltage switch that, depending on the geographic location, can be set to 110v/115v or 220v/240v. Due to the different power voltages supplied by power outlets in different countries, this switch position is crucial.

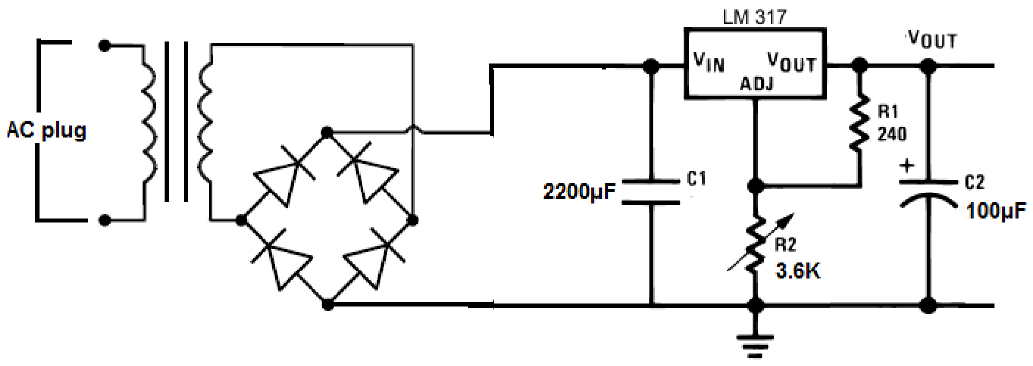


Fig 3.2: Some basic components used in the supply of power

Transformer

A transformer is a static electrical gadget that exchanges control between at least two circuits. A fluctuating current creates a changing attractive motion in one transformer curl, which thus actuates a differing electromotive power over a second loop twisted around a similar center.

Without a metallic association between the two circuits, electrical vitality can be exchanged between the two loops. The enlistment law of Faraday found in 1831 portrayed the impact of prompted voltage in any curl because of the changing attractive flux surrounded by the coil.

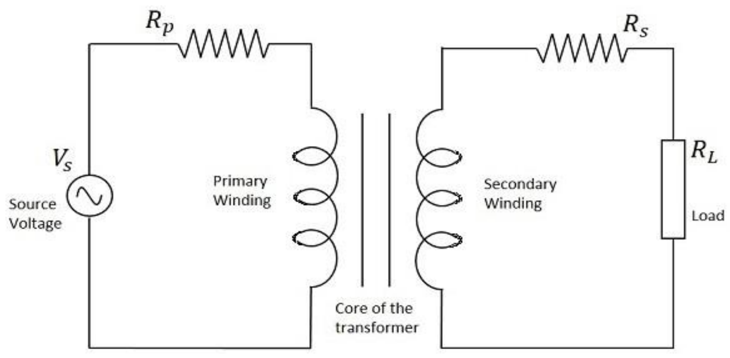


Fig 3.3: Circuit of transformer

Transformer

Rectifier

A rectifier is an electrical device that converts alternating (AC), which periodically reverses direction, to direct current (DC), which flows in only one direction. The process is known as *rectification*, since it "straightens" the direction of current.

Rectifiers have many uses, but are often found to serve as components of DC power supplies and direct power transmission systems with high voltage. Rectification can be used in roles other than direct current generation for use as a power source.

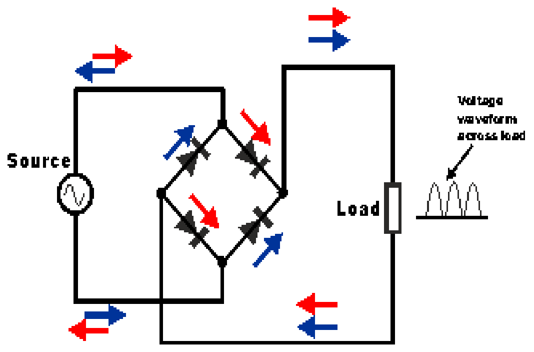
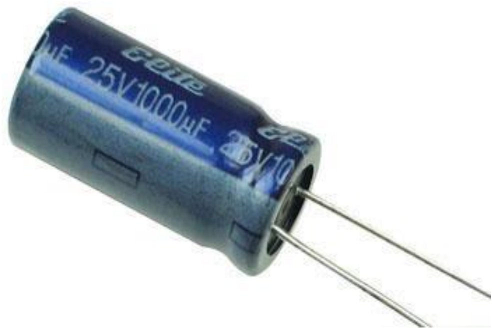


Fig 3.4: Circuit of rectifier **Rectifier**

Capacitors

Capacitors are used to attain from the connector the immaculate and smoothest DC voltage in which the rectifier is used to obtain throbbing DC voltage which is used as part of the light of the present identity. Capacitors are used to acquire square DC from the current AC experience of the current channels so that they can be used as a touch of parallel yield.

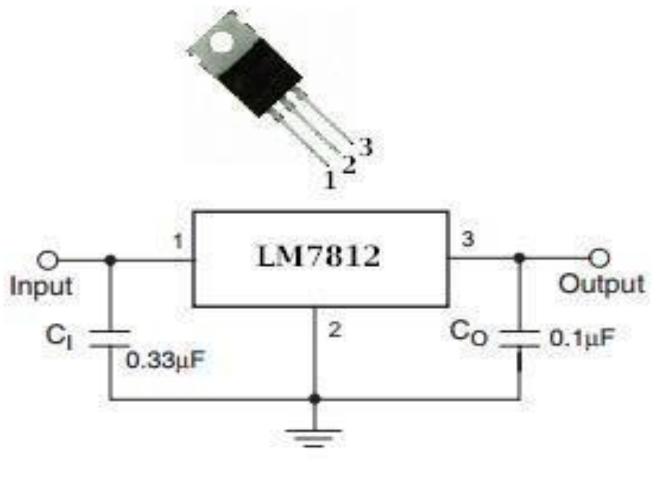


Capacitor

Voltage regulators

The 78XX voltage controller is mainly used for voltage controllers as a whole. The XX speaks to the voltage delivered to the specific gadget by the voltage controller as the yield. 7805 will supply and control 5v yield voltage and 12v yield voltage will be created by 7812.

The voltage controllers are that their yield voltage as information requires no less than 2 volts. For example, 7805 as sources of information will require no less than 7V, and 7812, no less than 14 volts.



7812 voltage regulator with pinout

3.3 Keypad

Keypad is an analog switching device which is generally available in matrix structure. It is used in many embedded system application for allowing the user to perform a necessary task.

It is used in many embedded system application for allowing the user to perform a necessary task. Consider the block diagram representation of interfacing keypad with microcontroller is:-A matrix keypad is consists of an arrangement of switches connected in matrix format in rows and columns.

A matrix keypad is consists of an arrangement of switches connected in matrix format in rows and columns. The rows and columns are connected with a microcontroller such that the rows of switches are connected to one pin and the columns of switches are connected to another pin of a microcontroller.

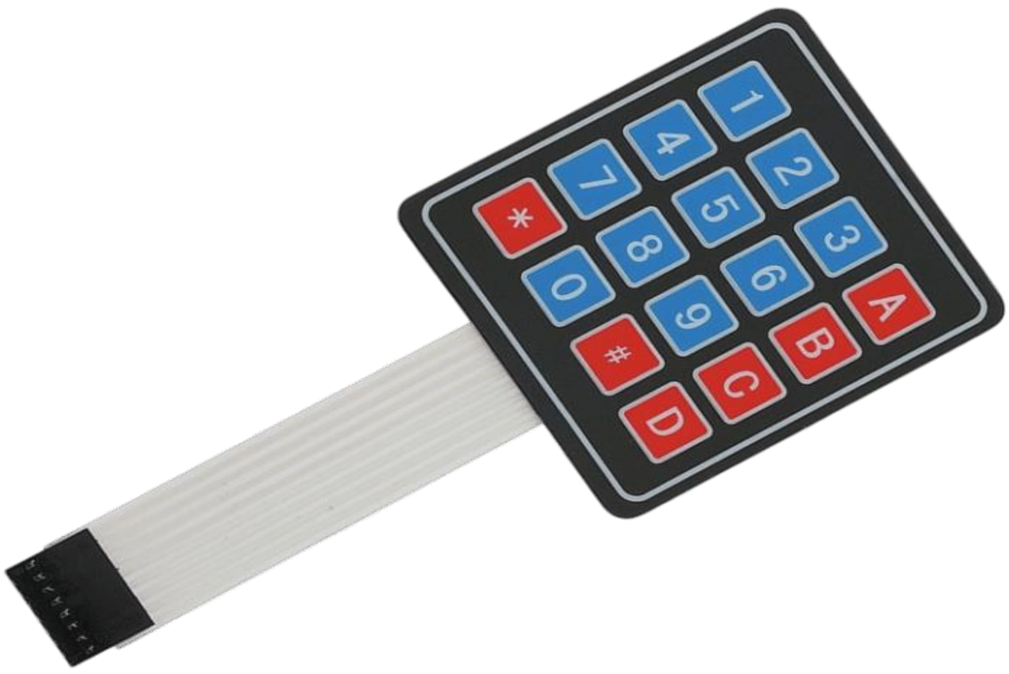


Fig 3.5: Digital keypad

3.4 LCD

LCD (Liquid Crystal Display) is the innovation utilized in scratch pad shows and other littler PCs. Like innovation for light-producing diode (LED) and gas-plasma, LCDs permit presentations to be a lot more slender than innovation for cathode beam tube (CRT). LCDs expend considerably less power than LED shows and gas shows since they work as opposed to emanating it on the guideline of blocking light.

A LCD is either made with an uninvolved lattice or a showcase network for dynamic framework show. Likewise alluded to as a meager film transistor (TFT) show is the dynamic framework LCD. The uninvolved LCD lattice has a matrix of conductors at every crossing point of the network with pixels. Two conductors on the lattice send a current to control the light for any pixel. A functioning framework has a transistor situated at every pixel crossing point, requiring less current to control the luminance of a pixel.

A 16x2 LCD show is an essential module that is generally utilized in various gadgets and circuits. These modules more than seven sections and other multi fragment LEDs are liked. The reasons being: LCDs are affordable; effectively programmable; have no restriction of showing exceptional and even custom characters (not at all like in seven fragments), movements, etc.

A 16x2 LCD implies 16 characters can be shown per line and 2 such lines exist. Each character is shown in a lattice of 5x7 pixels in this LCD. There are two registers in this LCD, in particular Command and Data.

Data/Signals/Execution of LCD

Now that was all about the signals and the hardware. Let us come to data, signals and execution.
Two types of signals are accepted by LCD, one is data and one is control. The LCD module recognizes these signals from the RS pin status. By pulling the R / W pin high, data can now also be read from the LCD display. Once the E pin has been pulsed, the LCD display reads and executes data at the falling edge of the pulse, the same for the transmission case.

There are two RAMs for LCD displays, namely DDRAM and CGRAM. DDRAM registers the position in which the character would be displayed in the ASCII chart. Each DDRAM byte represents every single position on the display of the LCD.

The DDRAM information is read by the LCD controller and displayed on the LCD screen. CGRAM enables users to define their personalized characters. Address space is reserved for users for the first 16 ASCII characters. Users can easily display their custom characters on the LCD screen after CGRAM has been set up to display characters.

Images of LCD Display

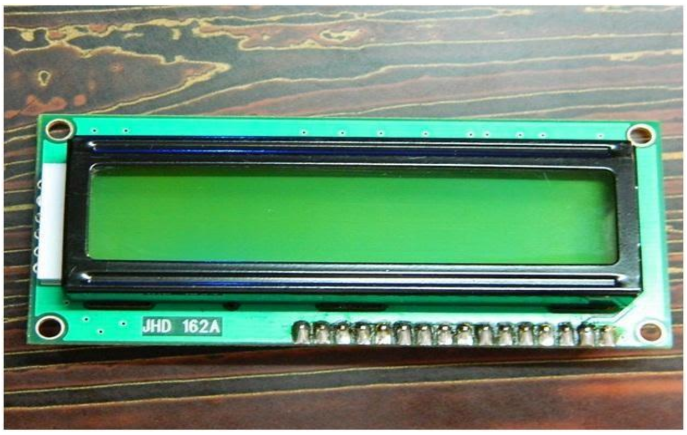
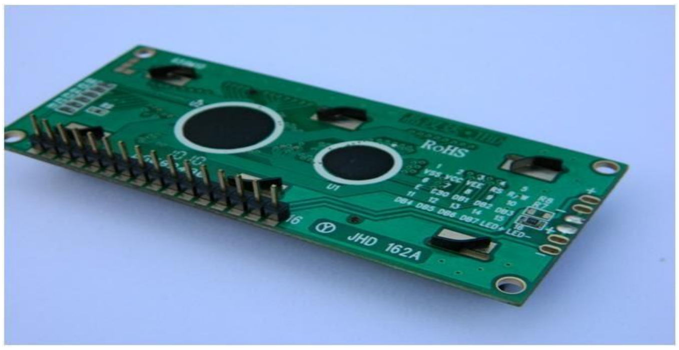


Fig 3.6 : LCD – Front View



LCD – Back View

Pin Diagram

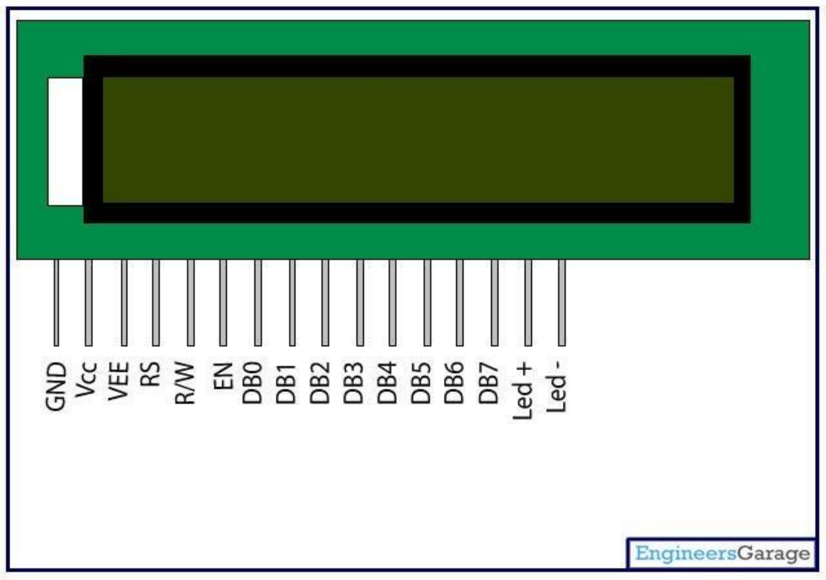


Table 3-1: Pin Description

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2

10	8-bit data pins	DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{cc} (5V)	Led+
16	Backlight Ground (0V)	Led-

RS (Register select)

A 16X2 LCD has two order and information registers. The determination of the register is utilized to change starting with one register then onto the next. RS=0 for the register of directions, while RS=1 for the register of information.

Command Register

The guidelines given to the LCD are put away by the direction register. An order is a direction given to LCD to play out a predefined assignment, for example, instating it, clearing its screen, setting the situation of the cursor, controlling showcase, and so on. Order preparing happens in the direction register.

Data Register

The information register will store the information that will be shown on the LCD. The information is the character's ASCII incentive to show on the LCD. It goes to the information register and is prepared there when we send information to the LCD. While choosing RS=1, the information register.

Read and Write Mode of LCD

As stated, the LCD itself comprises of an interface IC. This interface IC can be perused or composed by the MCU. A large portion of the occasions we're simply going to keep in touch with the IC since perusing will make it increasingly perplexing and situations like that are exceptionally uncommon. Information such as cursor position, status completion interrupts, etc. can be read if necessary.

Table 3-2 : LCD Commands

There are some preset commands in the LCD that we need to send to the LCD via some microcontroller. The following are some important command instructions:

Sr. No.	Hex Code	Command to LCD instruction Register
1	01	Clear display screen
2	02	Return home
3	04	Decrement cursor (shift cursor to left)
4	06	Increment cursor (shift cursor to right)
5	05	Shift display right
6	07	Shift display left
7	08	Display off, cursor off
8	0A	Display off, cursor on
9	0C	Display on, cursor off
10	0E	Display on, cursor blinking
11	0F	Display on, cursor blinking
12	10	Shift cursor position to left
13	14	Shift cursor position to right

14	18	Shift the entire display to the left
15	1C	Shift the entire display to the right
16	80	Force cursor to beginning (1st line)
17	C0	Force cursor to beginning (2nd line)
18	38	2 lines and 5×7 matrix

Command codes for LCD

Block Diagram of LCD Display

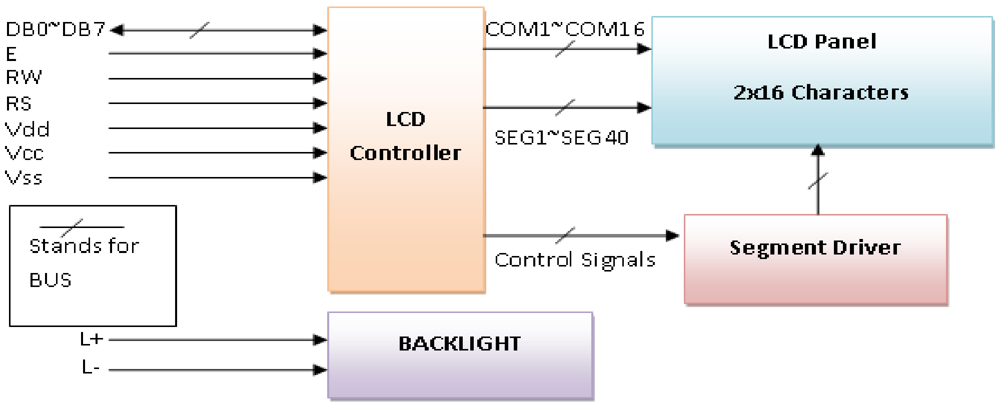


Fig 3.7 :Block diagram of LCD display

4-bit and 8-bit Mode of LCD

The LCD can work in two striking modes, the 4-bit mode and the 8-bit mode. We send the information snack through snack in 4 bit mode, first upper chomp, by then lower snack. For those of you who don't have the foggiest idea what a goody is: a chomp is a four-piece gathering, so a byte's lower four bits (D0-D3) are the lower snack, while a byte's upper four bits (D4-D7) are the higher snack.

3.5 Ultrasonic Sensor

An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves. An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity.

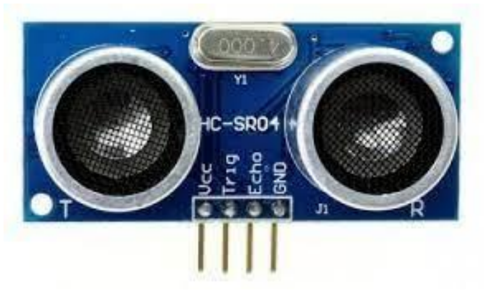


Fig 3.8: Ultrasonic Sensor

HC-SR04 Ultrasonic (US) sensor is used in this project which is a 4 pin module with pin names as Vcc, Echo, Ground and Trigger respectively. This sensor is widely used in applications where object sensing and distance measurement is required. This sensor comprises of two eyes like projections in the front which represents the Ultrasonic Receiver and Transmitter. Using these transmitters and receivers this sensor measures the distances depending on the transmitted and received ultrasonic signals. This sensor works based on the simple formula Distance = Time × Speed. An ultrasonic wave is transmitted by the Ultrasonic transmitter, this transmitted wave travels in air and when it gets objected by any obstacle or a material, the wave gets reflected back towards the ultrasonic sensor. This reflected wave is detected by the arranged Ultrasonic receiver module as observed.

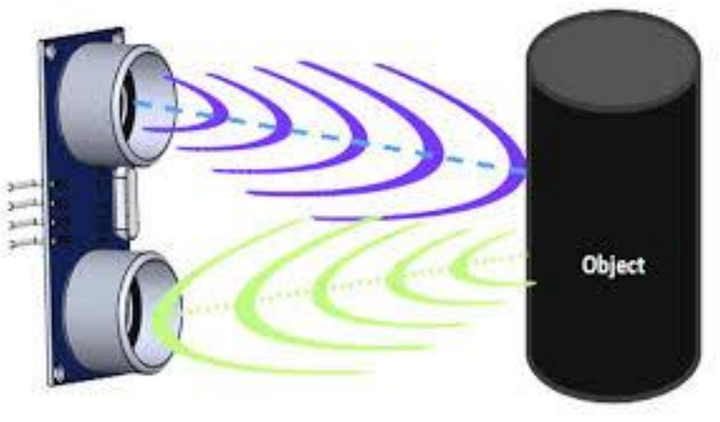


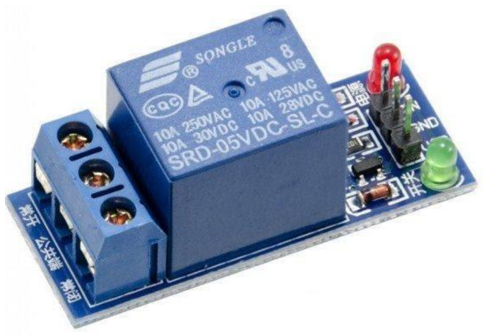
Fig. 3.9:Ultrasonic sensor transmission and reception of the signal

3.6 Relay

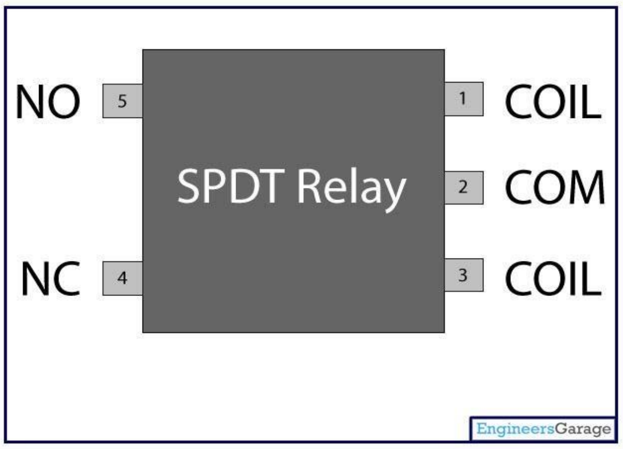
What is a relay?

A relay is an electromagnetic switch that is used to turn on and turn off a circuit by a low power signal, or where several circuits must be controlled by one signal.

Most of the high end industrial application devices have relays for their effective working. Relays are simple switches which are operated both electrically and mechanically. Relays consist of an electromagnet and also a set of contacts. The switching mechanism is carried out with the help of the electromagnet. There are also other operating principles for its working. But they differ according to their applications. Most of the devices have the application of relays.



Pin Diagram:



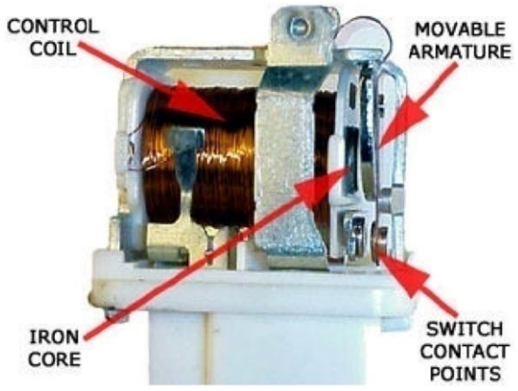
Why is a relay used?

The main operation of a relay comes in places where only a low-power signal can be used to control a circuit. It is also used in places where only one signal can be used to control a lot of circuits. The application of relays started during the invention of telephones. They played an important role in switching calls in telephone exchanges. They were also used in long distance telegraphy. They were used to switch the signal coming from one source to another destination. After the invention of computers they were also used to perform Boolean and other logical operations. The high end applications of relays require high power to be driven by electric motors and so on. Such relays are called contactors.

Relay Design

- There are only four main parts in a relay. They are
- Electromagnet
- Movable Armature
- Switch point contacts
- Spring

The figures given below show the actual design of a simple relay.



Relay Construction

It is an electro-magnetic relay with a wire coil, surrounded by an iron core. A path of very low reluctance for the magnetic flux is provided for the movable armature and also the switch point contacts.

The movable armature is connected to the yoke which is mechanically connected to the switch point contacts. These parts are safely held with the help of a spring. The spring is used so as to produce an air gap in the circuit when the relay becomes de-energized.

How relay works?

The relay function can be better understood by explaining the following diagram given below.

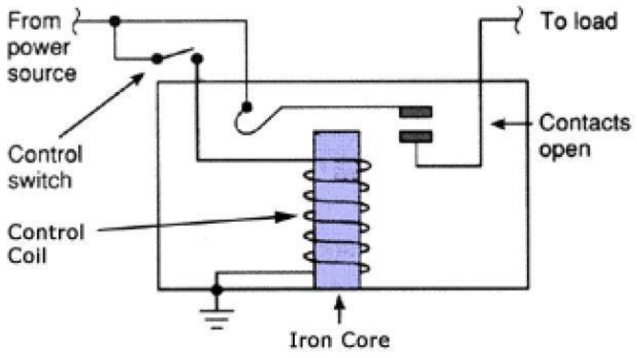


Fig 3.10:*Relay Design*

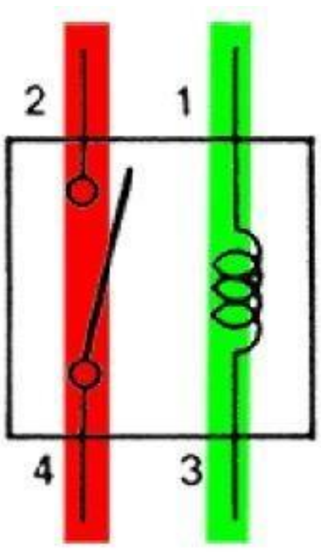
The diagram shows an inner section diagram of a relay. An iron core is surrounded by a control coil. As shown, the power source is given to the electromagnet through a control switch and through contacts to the load. When current starts flowing through the control coil, the electromagnet starts energizing and thus intensifies the magnetic field. Thus the upper contact arm starts to be attracted to the lower fixed arm and thus closes the contacts causing a short circuit for the power to the load. On the other hand, if the relay was already de-energized when the contacts were closed, then the contact move oppositely and make an open circuit.

As soon as the coil current is off, the movable armature will be returned by a force back to its initial position. This force will be almost equal to half the strength of the magnetic force. This force is mainly provided by two factors. They are the spring and also gravity.

Relays are mainly made for two basic operations. One is low voltage application and the other is high voltage. For low voltage applications, more preference will be given to reduce the noise of the whole circuit. For high voltage applications, they are mainly designed to reduce a phenomenon called arcing.

Relay Basics

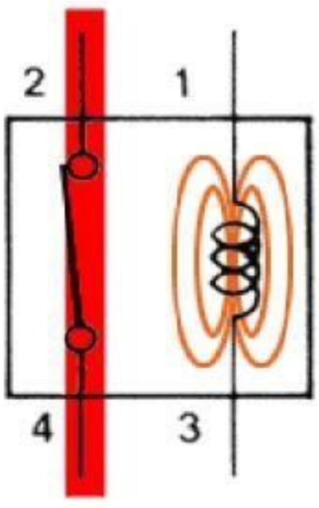
The basics for all the relays are the same. Take a look at a 4 pin relay shown below. There are two colors shown. The green color represents the control circuit and the red color represents the load circuit. A small control coil is connected onto the control circuit. A switch is connected to the load. This switch is controlled by the coil in the control circuit. Now let us take the different steps that occur in a relay.



Relay operation

- **Energized Relay (ON)**

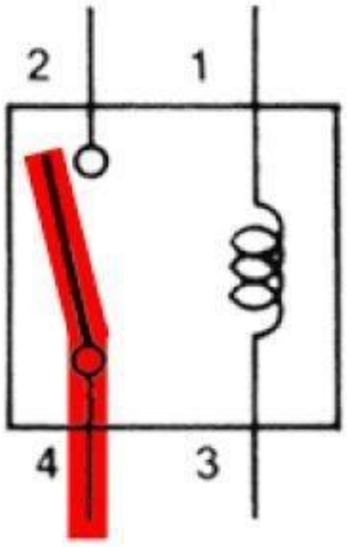
As shown in the circuit, the current flowing through the coils represented by pins 1 and 3 causes a magnetic field to be aroused. This magnetic field causes the closing of the pins 2 and 4. Thus the switch plays an important role in the relay working. As it is a part of the load circuit, it is used to control an electrical circuit that is connected to it. Thus, when the electrical relay in energized the current flow will be through the pins 2 and 4.



Energized Relay (ON)

- **De – Energized Relay (OFF)**

As soon as the current flow stops through pins 1 and 3, the relay switch opens and thus the open circuit prevents the current flow through pins 2 and 4. Thus the relay becomes de-energized and thus in off position.



De-Energized Relay (OFF)

In simple, when a voltage is applied to pin 1, the electromagnet activates, causing a magnetic field to be developed, which goes on to close the pins 2 and 4 causing a closed circuit. When there is no voltage on pin 1, there will be no electromagnetic force and thus no magnetic field. Thus the switches remain open.

Pole and Throw

Relays have the exact working of a switch. So, the same concept is also applied. A relay is said to switch one or more poles. Each pole has contacts that can be thrown in mainly three ways. They are

- **Normally Open Contact (NO):** NO contact is also called a make contact. It closes the circuit when the relay is activated. It disconnects the circuit when the relay is inactive.
- **Normally Closed Contact (NC):** NC contact is also known as break contact. This is opposite to the NO contact. When the relay is activated, the circuit disconnects. When the relay is deactivated, the circuit connects.

- **Change-over (CO) / Double-throw (DT) Contacts:** This type of contacts are used to control two types of circuits. They are used to control a NO contact and also a NC contact with a common terminal. According to their type they are called by the names **break before make** and **make before break** contacts.

  Relays can be used to control several circuits by just one signal. A relay switches one or more poles, each of whose contacts can be thrown by energizing the coil.
  Relays are also named with designations like

- **Single Pole Single Throw (SPST)**: The SPST relay has a total of four terminals. Out of these two terminals can be connected or disconnected. The other two terminals are needed for the coil to be connected.
- **Single Pole Double Throw (SPDT):** The SPDT relay has a total of five terminals. Out of these two are the coil terminals. A common terminal is also included which connects to either of two others.
- **Double Pole Single Throw (DPST):** The DPST relay has a total of six terminals. These terminals are further divided into two pairs. Thus they can act as two SPST which are actuated by a single coil. Out of the six terminals two of them are coil terminals.
- **Double Pole Double Throw (DPDT)**: The DPDT relay is the biggest of all. It has mainly eight relay terminals. Out of these two rows are designed to be change over terminals. They are designed to act as two SPDT relays which are actuated by a single coil.

Relay Applications

- A relay circuit is used to realize logic functions. They play a very important role in providing safety critical logic.
- Relays are used to provide time delay functions. They are used to time the delay open and delay close of contacts.
- Relays are used to control high voltage circuits with the help of low voltage signals. Similarly they are used to control high current circuits with the help of low current signals.

- They are also used as protective relays. By this function all the faults during transmission and reception can be detected and isolated.

Relays available to buy comes in different specifications, the most important of them being the current ranges and response time. Most of them are designed to automatically reset to work after the motor is turned back on.

3.7 Nodemcu

NodeMCU (Node Microcontroller Unit) is a low-cost open source IOT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espress if Systems, and hardware which was based on the ESP-12 module. Later, support for the ESP32 32-bit MCU was added.

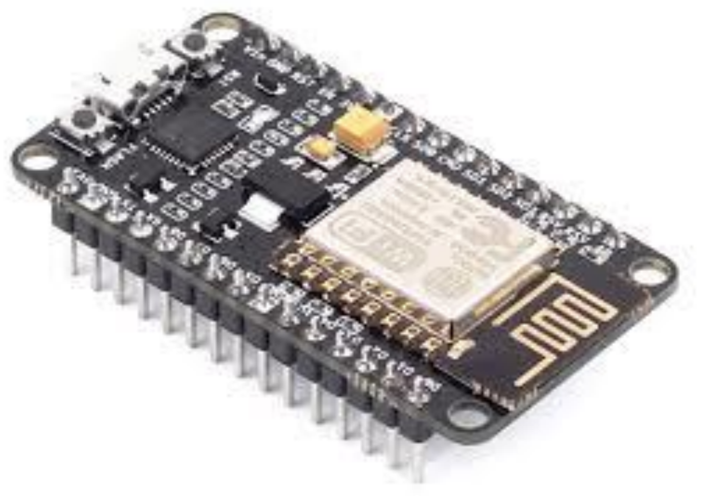


Figure 3.11: Node MCU Development Board

NodeMCU is an open source firmware for which open source prototyping board designs are available. The name “NodeMCU” combines “node” and “MCU” (micro-controller unit). The term “NodeMCU” strictly speaking refers to the firmware rather than the associated development kits.

Both the firmware and prototyping board designs are open source.

The firmware uses the Lua scripting language. The firmware is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266.

It uses many open source projects, such as luacjson and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.

The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards.

3.7.1 Pin Configuration of Node MCU Development Board

This module provides an access to the GPIO subsystem. All the access is based on I/O index number of Node MCU kits, not the internal GPIO pins. For example, the D0 pin on the development kit is mapped to GPIO pin 16. Node MCU provides access to the GPIO pins and the following pin mapping table is a part of the API documentation.

The ESP8266 Node MCU has total 30 pins that interface it to the outside world. The pins are grouped by their functionality as:

Power pins: There are four power pins viz. one VIN pin & three 3.3V pins. The VIN pin can be used to directly supply the ESP8266 and its peripherals, if you have a regulated 5V voltage source.

GND: is a ground pin of ESP8266 Node MCU development board.

12 IC Pins: are used to hook up all sorts of I2C sensors and peripherals in your project. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum.

GPIO Pins: ESP8266 Node MCU has 17 GPIO pins which can be assigned to various functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance

ADC Channel: The Node MCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC viz. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.

UART Pins: ESP8266 Node MCU has 2 UART interfaces, i.e. UART0 and UART1, which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication.

SPI Pins: ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:

•4 timing modes of the SPI format transfer

•Up to 80 MHz and the divided clocks of 80 MHz

•Up to 64-Byte FIFO

SDIO Pins: ESP8266 features Secure Digital Input/output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.

PWM Pins: The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μs to 10000 μs, i.e., between 100 Hz and 1 kHz.

Control Pins: are used to control ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.

•EN pin – The ESP8266 chip is enabled when EN pin is pulled HIGH.

When pulled LOW.

- The chip works at minimum power. RST pin – RST pin is used to reset

the ESP8266 chip.

•WAKE pin – Wake pin is used to wake the chip from deep-sleep

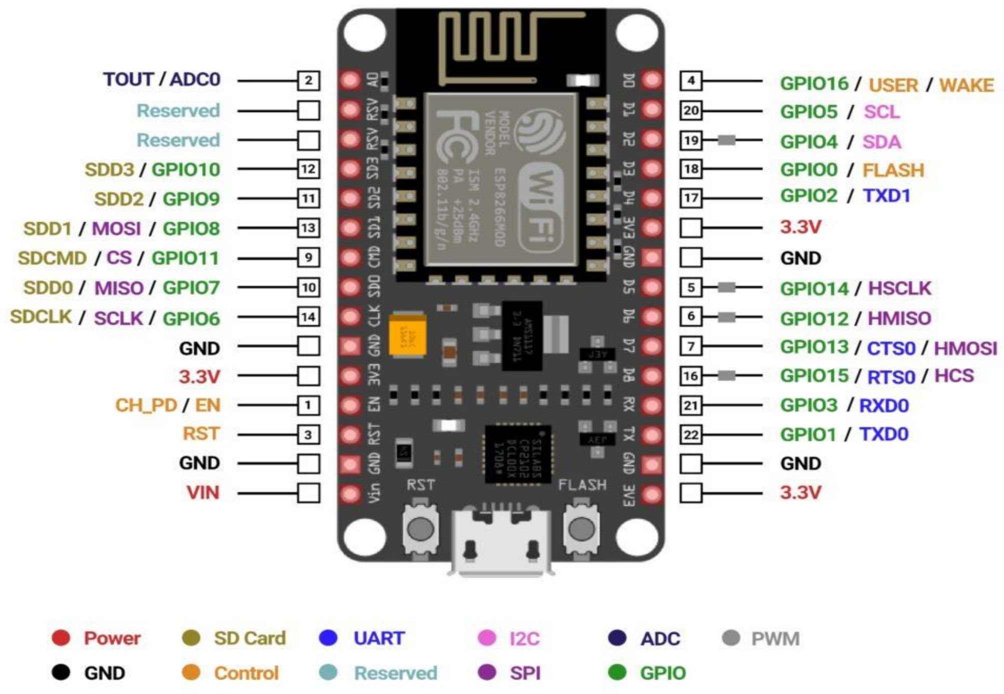


Figure 3.12: ESP8266 Node MCU pinout

3.7.2 Parts of Node MCU Development Board

ESP 12-E Module

The development board equips the ESP-12E module containing ESP8266 chip having TensilicaXtensa® 32-bit LX106 RISC microprocessor which operates at 80 to 160 MHz adjustable clock frequency and supports RTOS.

There’s also 128 KB RAM and 4MB of Flash memory (for program and data storage) just enough to cope with the large strings that make up web pages, JSON/XML data, and everything we throw at IOT devices nowadays.

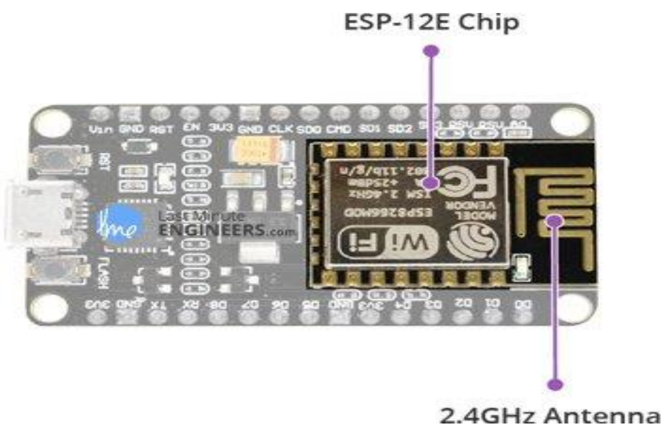


Figure 3.13: ESP 12E module in Node MCU Development board.

•TensilicaXtensa® 32-bit LX106

•80 to 160 MHz clock frequency

•128 kb internal RAM

•4 MB external flash

•802.11b/g/n HT40 Wi-Fi transceiver.

Power Requirements

As the operating voltage range of ESP8266 is 3V to 3.6V, the board comes with a LDO voltage regulator to keep the voltage steady at 3.3V. It can reliably supply up to 600mA, which should be more than enough when ESP8266 pulls as much as 80mA during RF transmissions. The output of the regulator is also broken out to one of the sides of the board and labelled as 3V3.

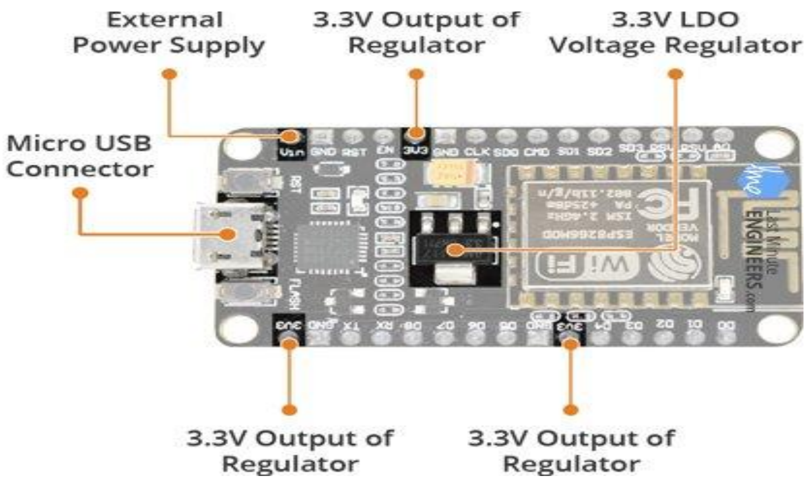


Figure 3.14: Power module on a Node MCU development board

•Operating voltage 2.5V to 3.6V

•On-board 3.6V 600mA regulator

•80 mA operating current

•20 µA during sleep mode

Peripheral I/O

The ESP8266 Node MCU has total 17 GPIO pins broken out to the pinheaders on both sides of the development board. These pins can be assigned to all sorts of peripheral duties, including:

•ADC channel – A 10-bit ADC channel.

• UART interface – UART interface is used to load code serially

.• PWM outputs – PWM pins for dimming LEDs or controlling motors

.•SPI, I2C

• I2S interface – SPI and I2C interface to hook up all sorts of sensors and peripherals. I2S interface – I2S interface if you want to add sound to your project

.• As a result of the pin multiplexing feature (Multiple peripherals multiplexed on a single GPIO pin), a single GPIO pin can act as PWM/UART/SPI.

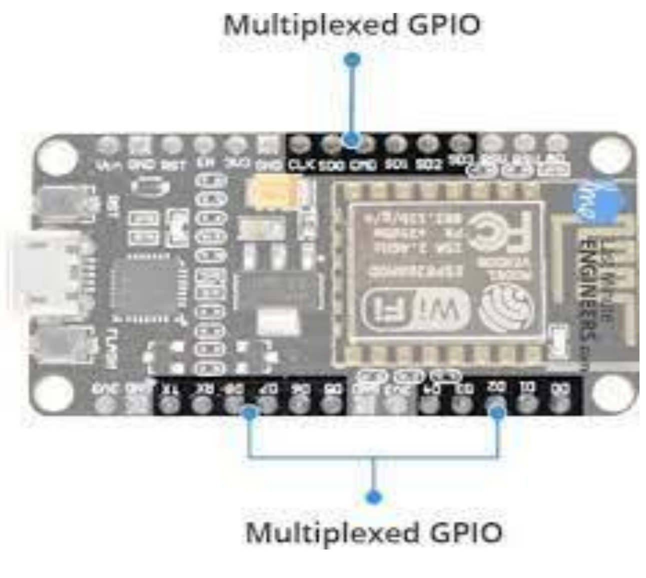


Figure 3.15 GPIO pins on Node MCU development board.

On Board Switches and LED Indicators

The ESP8266 Node MCU features two buttons. One marked as RST located on the top left corner is the Reset button, used of course to reset the ESP8266 chip. The other FLASH button on the bottom left corner is the download button used while upgrading firmware. The board also has a LED indicator which is user programmable and is connected to the D0 pin of the board.

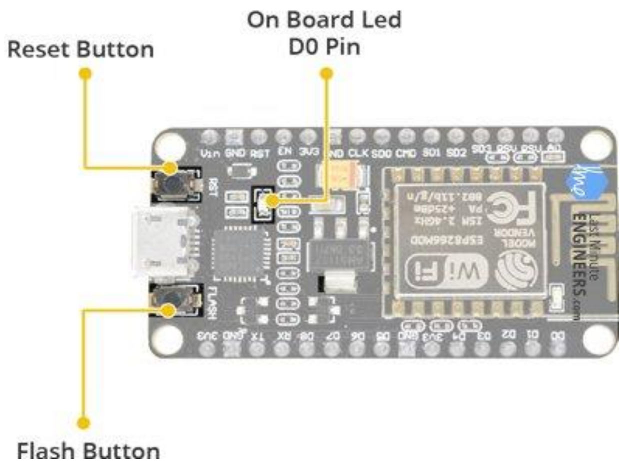


Figure 3.16: ON board switches and LED indicators on Node MCU development board.

Switches and indicators

- RST: Reset the ESP8266 chip
- FLASH: Download new programs
- Blue LED: User programmable.

Serial Communication

The board includes CP2102 USB-to-UART Bridge Controller from Silicon Labs, which converts USB signal to serial and allows your computer to program and communicate with the ESP8266 chip.

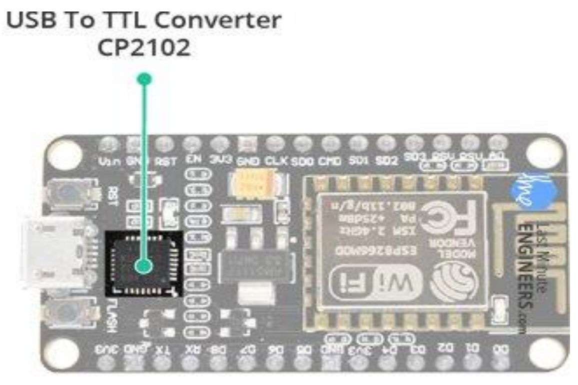


Figure 3.17: CP2120 on Node MCU development board

- CP2120 USB-to-UART converter
- 4.5 Mbps communication speed

SOFTWARE REQUIREMENTS

3.8 Blynk App

Blynk App provides a platform to user for designing own app that is connected to the Blynk Server that provides a path for transmission and reception between the developed project (kit) and user.

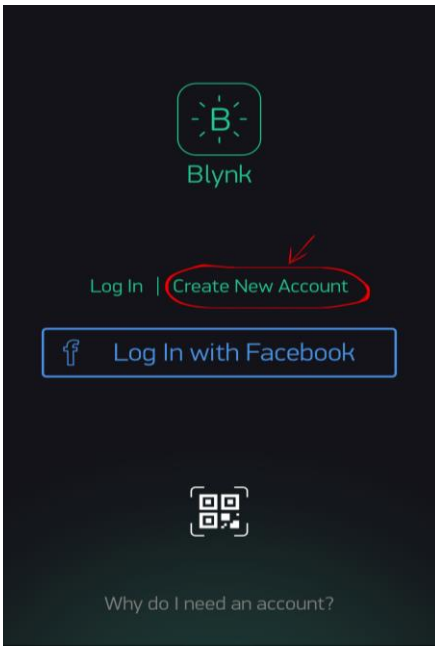
3.8.1 How To Install Blynk App

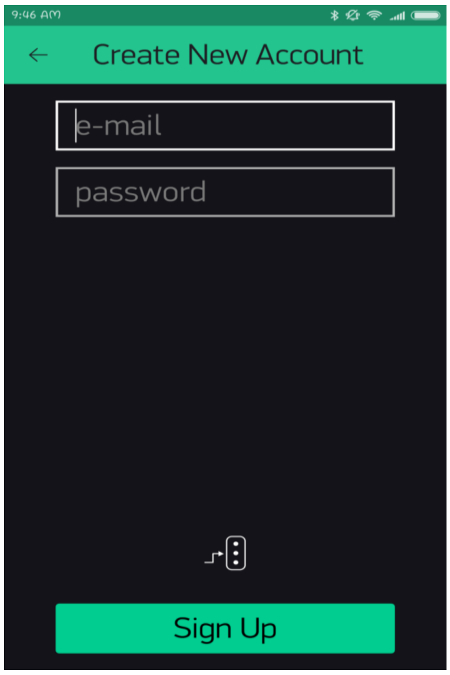
Firstly, download and install this Blynk app on your Smartphone.

Secondly, we need to create a new account, then click on the "Create New Account" button and enter the Gmail address and password of your choice. Afterward, click the "Sign Up" button. Then a mail will go to the email you entered.There is a code called "Auth Token" in that mail and that code is what we need for our future work.

Auth Token

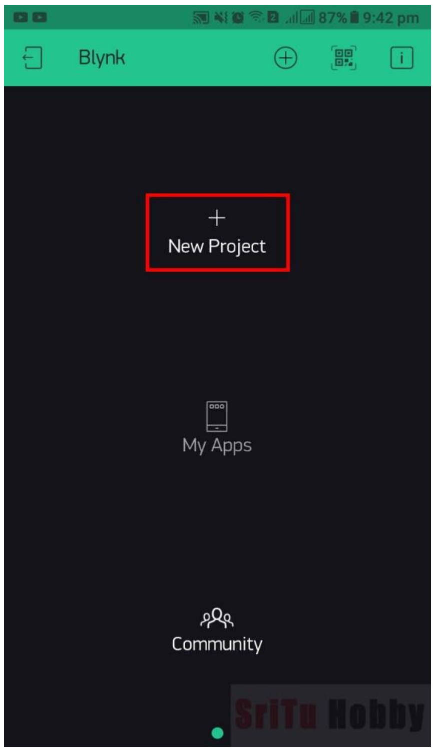
Auth Token is a unique identifier which is needed to connect your hardware to your smartphone. Every new project you create will have its own Auth Token. You'll get Auth Token automatically on your email after project creation. You can also copy it manually. Click on devices section and selected required device**.**

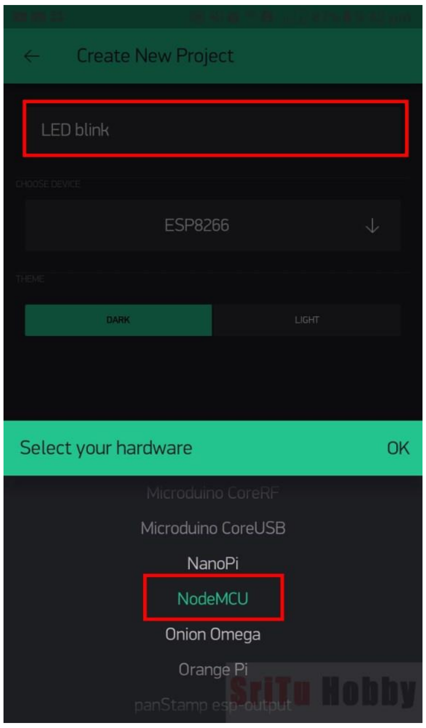


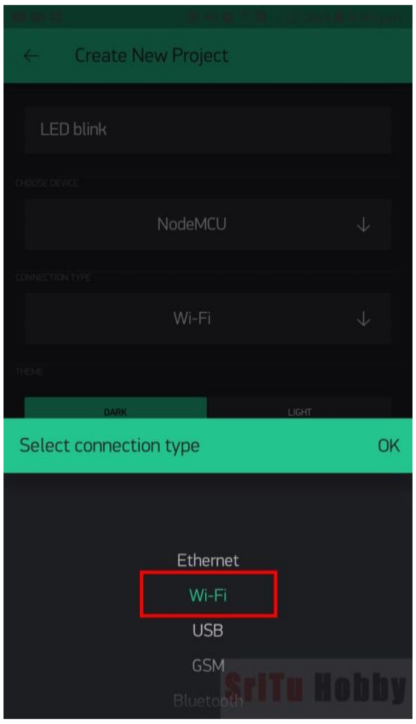


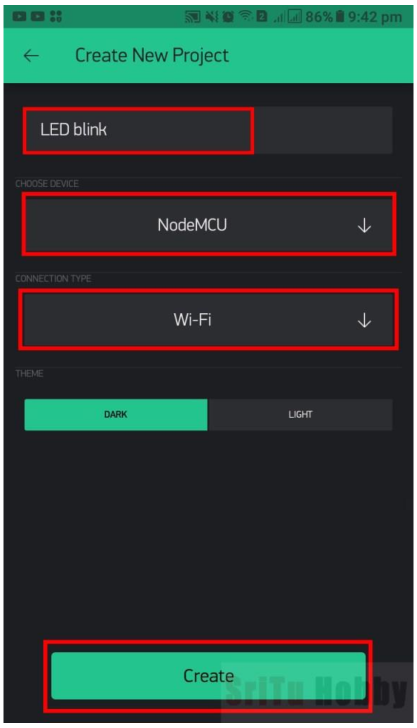
Write your email and a password to create an account on the Blynk server

Thirdly, click the "New Project" button and enter your project name here. Then select Nodemcu instead of "CHOOSE DEVICE". Because we are using a Nodemcu here. Afterward, select the wifi in the "Connection Type" field.Lastly, click the "Confirm" button. Ok, now let's look at the project interface.

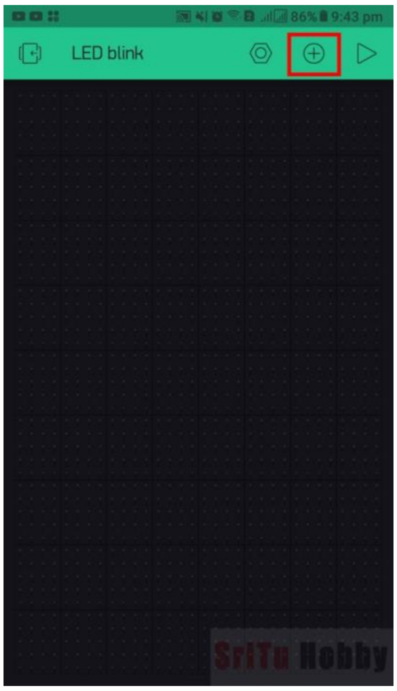


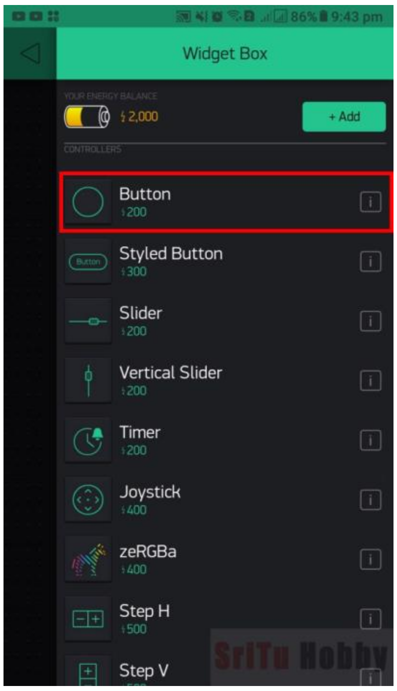




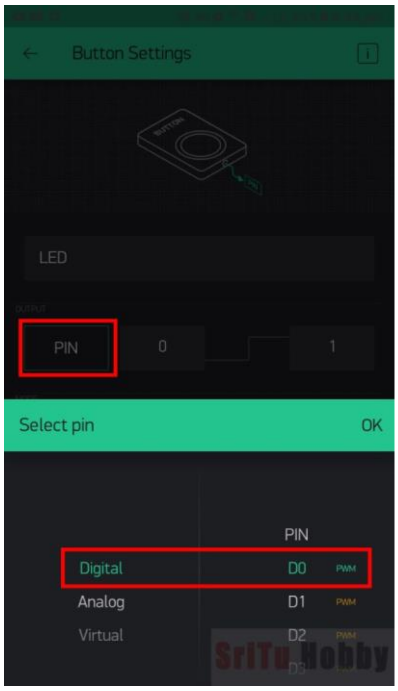


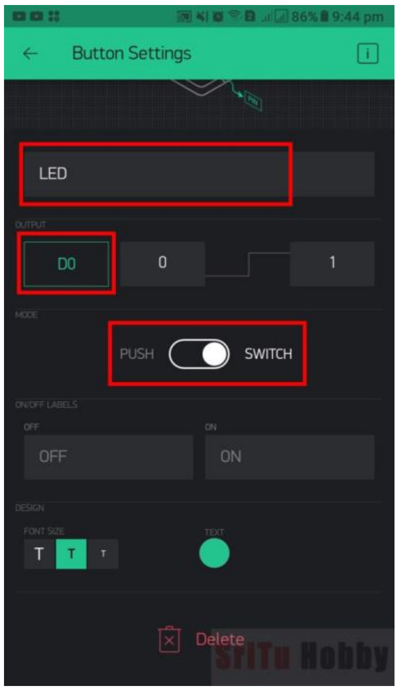
Now we can add widgets to our LED ON and OFF project. For that, click the "+" sign in the upper right corner and add a "Button" widget.

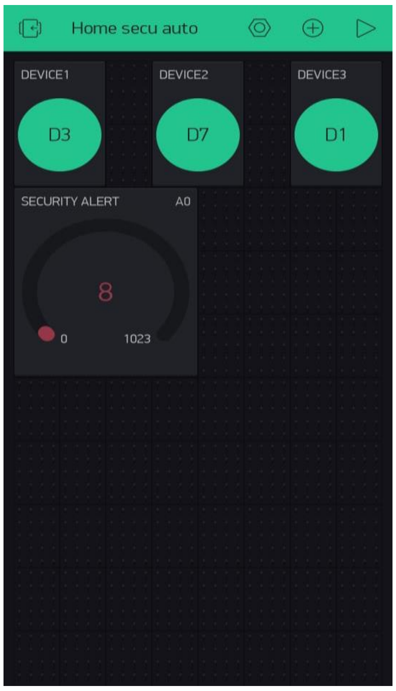




Afterward, we have to make this button as we want. First, give it a name you like. I have put it as LED. Then click on Pin and select Digital and select D0. Next, pull the button under the mode to the switch side. If you want to change the color you can change it. When you have done all this, click the Back button.

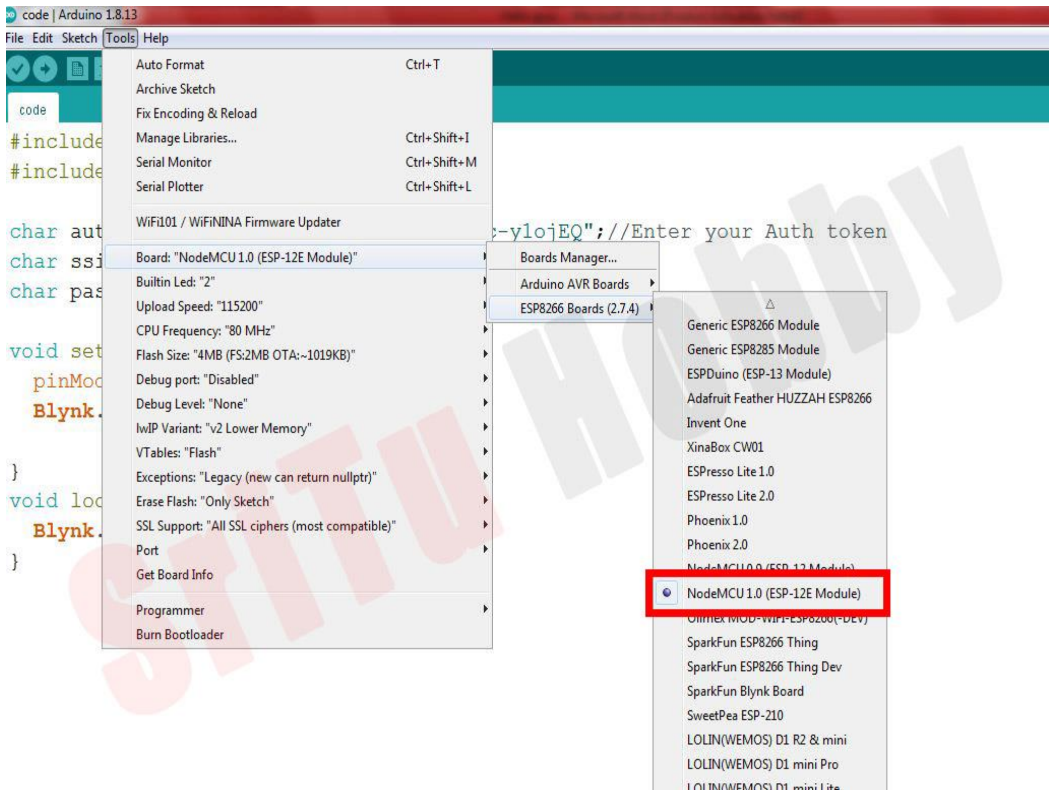






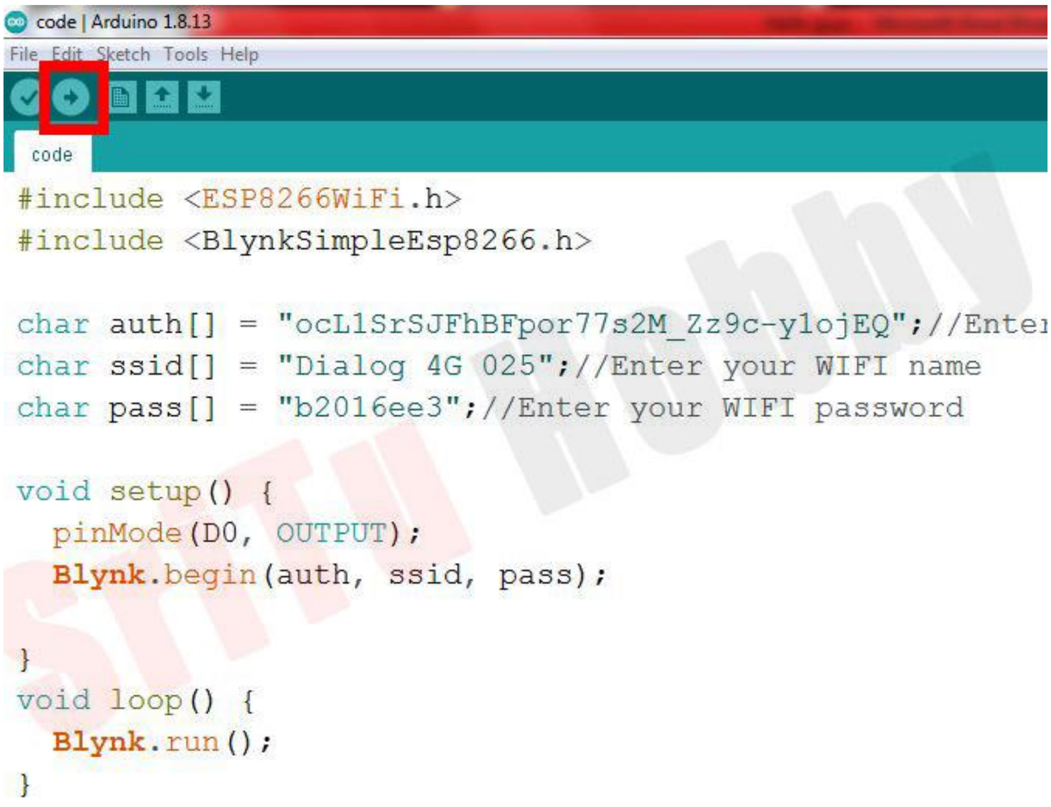
So, now let’s create a program. For this, we first need two libraries. That is the library required for WIFI and the library required for the Blynk application.

Now we are all done. So, now all you have to do is connect your NodeMCU Board to your Laptop or Desktop computer, select NodeMCU 1.0 for the Arduino IDE Board, select the relevant port, and upload the code. Once the code is uploaded, all you have to do is run the project we created in the Blynk app. You can do that by clicking on the triangle on the right side of the Blynk app.

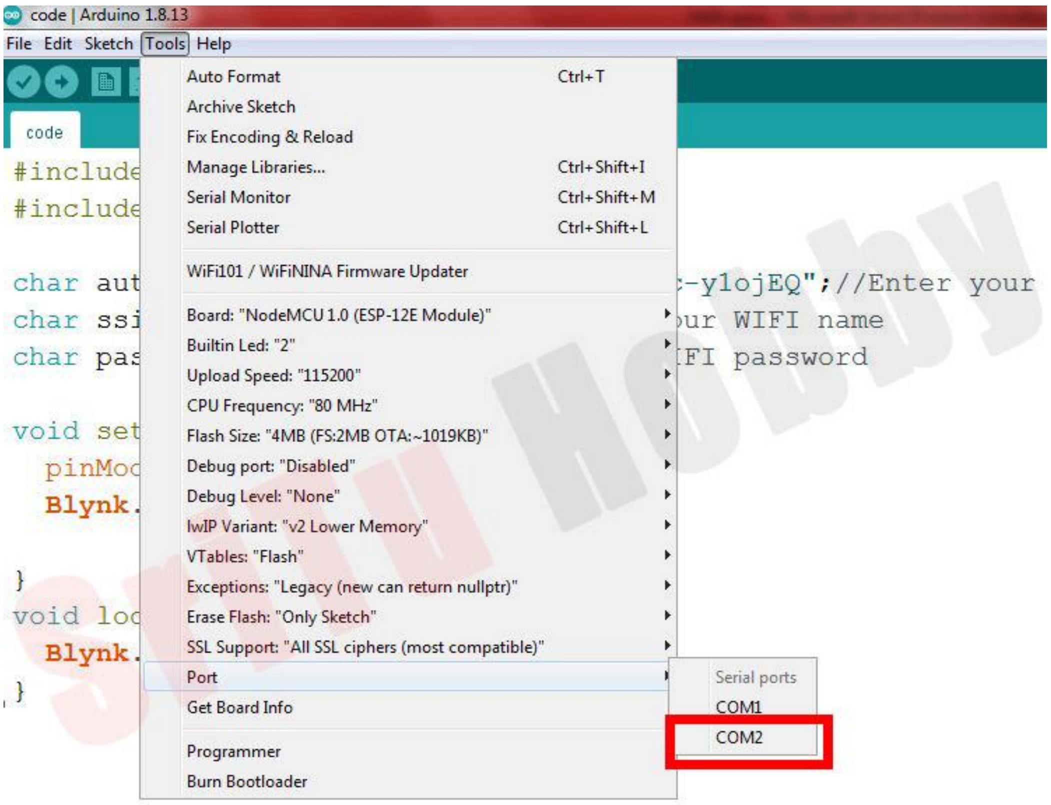


```
#include
#include

char aut                              -y1ojEQ";//Enter your Auth token
char ssi
char pas

void set
  pinMod
  Blynk.

}
void loo
  Blynk.
}
```

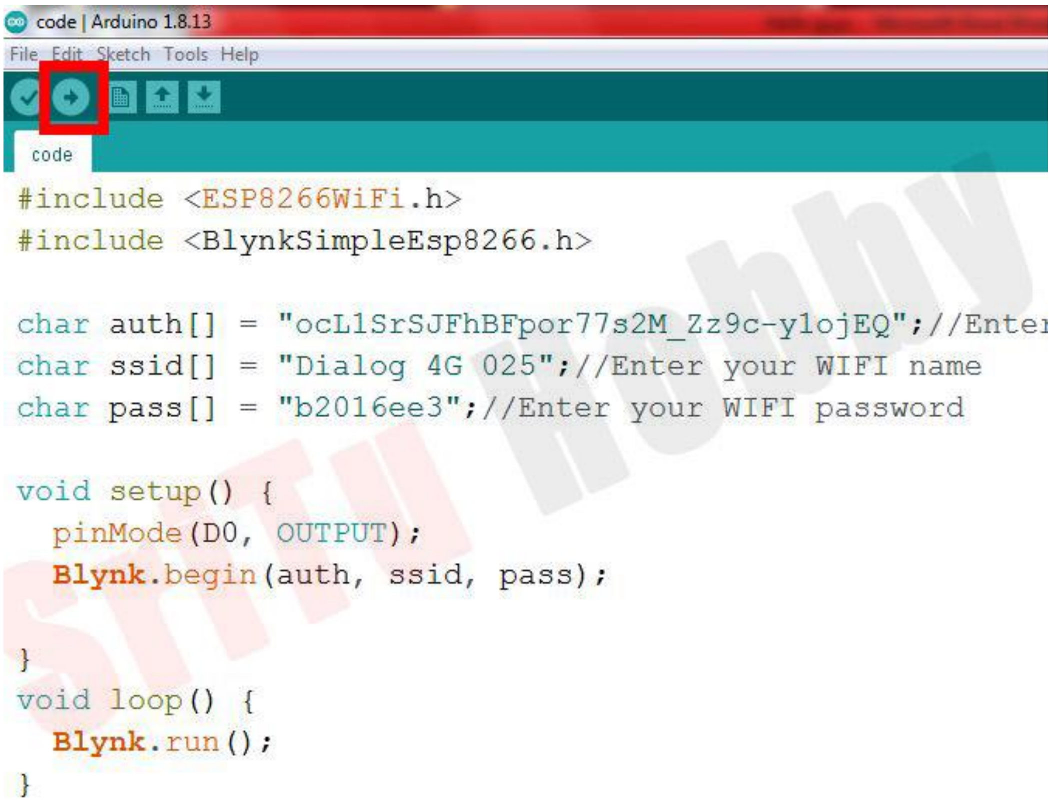
Board: "NodeMCU 1.0 (ESP-12E Module)"
Builtin Led: "2"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Flash Size: "4MB (FS:2MB OTA:~1019KB)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
Exceptions: "Legacy (new can return nullptr)"
Erase Flash: "Only Sketch"
SSL Support: "All SSL ciphers (most compatible)"



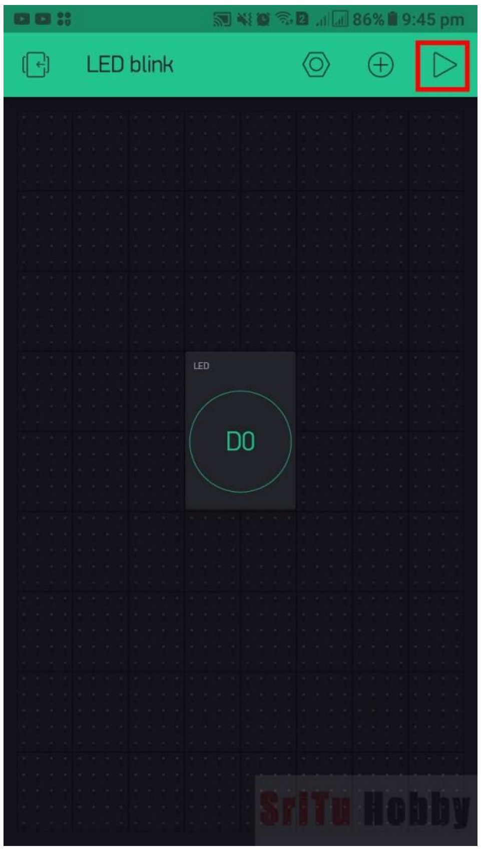
```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = "ocL1SrSJFhBFpor77s2M_Zz9c-y1ojEQ";//Enter
char ssid[] = "Dialog 4G 025";//Enter your WIFI name
char pass[] = "b2016ee3";//Enter your WIFI password

void setup() {
  pinMode(D0, OUTPUT);
  Blynk.begin(auth, ssid, pass);

}
void loop() {
  Blynk.run();
}
```



```
Auto Format                                   Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries...                           Ctrl+Shift+I
Serial Monitor                                Ctrl+Shift+M
Serial Plotter                                Ctrl+Shift+L

WiFi101 / WiFiNINA Firmware Updater

Board: "NodeMCU 1.0 (ESP-12E Module)"
Builtin Led: "2"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Flash Size: "4MB (FS:2MB OTA:~1019KB)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
Exceptions: "Legacy (new can return nullptr)"
Erase Flash: "Only Sketch"
SSL Support: "All SSL ciphers (most compatible)"
Port                                          Serial ports
Get Board Info                                COM1
                                              COM2
Programmer
Burn Bootloader
```



```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = "ocL1SrSJFhBFpor77s2M_Zz9c-y1ojEQ";//Enter
char ssid[] = "Dialog 4G 025";//Enter your WIFI name
char pass[] = "b2016ee3";//Enter your WIFI password

void setup() {
  pinMode(D0, OUTPUT);
  Blynk.begin(auth, ssid, pass);

}
void loop() {
  Blynk.run();
}
```



Then with the button, we created you can turn the LED bulb on / off. Also, remember to activate your internet connection at this time. You can watch this in the video below. In this tutorial, we have created a simple project using Nodemcu. If you understand this well, it will be very useful for the projects of our future articles.

3.9 Arduino IDE

Arduino IDE where IDE stands for Integrated Development Environment – An official software introduced by Arduino.cc, that is mainly used for writing, compiling and uploading the code in the Arduino Device. Almost all Arduino modules are compatible with this software that is an open source and is readily available to install and start compiling the code on the go.

Introduction to Arduino IDE

- Arduino IDE is an open source software that is mainly used for writing and compiling the code into the Arduino Module.
- It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process.
- It is easily available for operating systems like MAC, Windows, and Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment.
- A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more.
- Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.
- The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.
- The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module.
- This environment supports both C and C++ languages.

How to install Arduino IDE

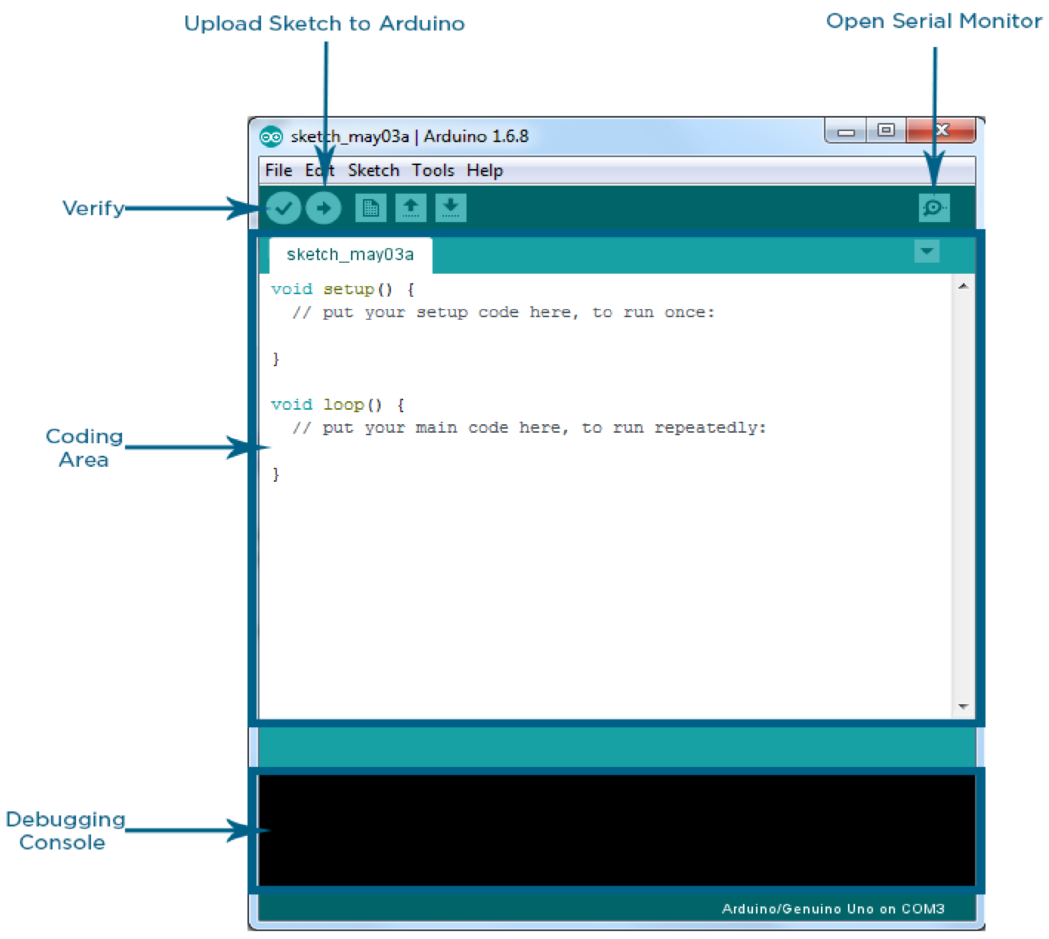
You can download the Software from Arduino main website. As I said earlier, the software is available for common operating systems like Linux, Windows, and MAX, so make sure you are downloading the correct software version that is easily compatible with your operating system.

- If you aim to download Windows app version, make sure you have Windows 8.1 or Windows 10, as app version is not compatible with Windows 7 or older version of this operating system.

The IDE environment is mainly distributed into three sections

- **1. Menu Bar**
- **2. Text Editor**
- **3. Output Pane**

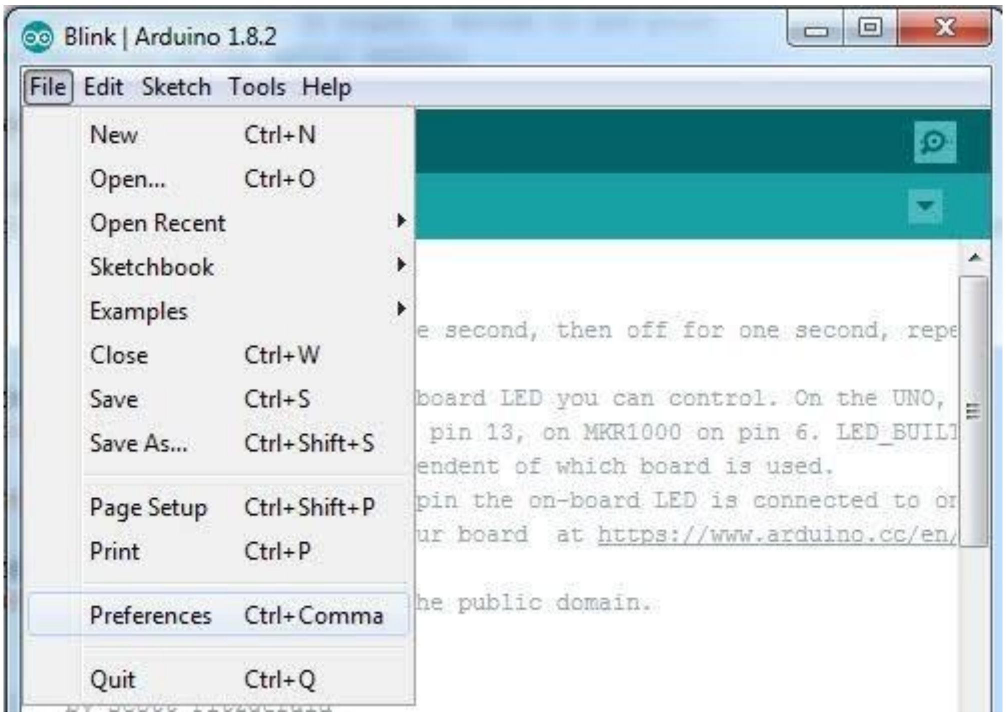
As you download and open the IDE software, it will appear like an image below.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

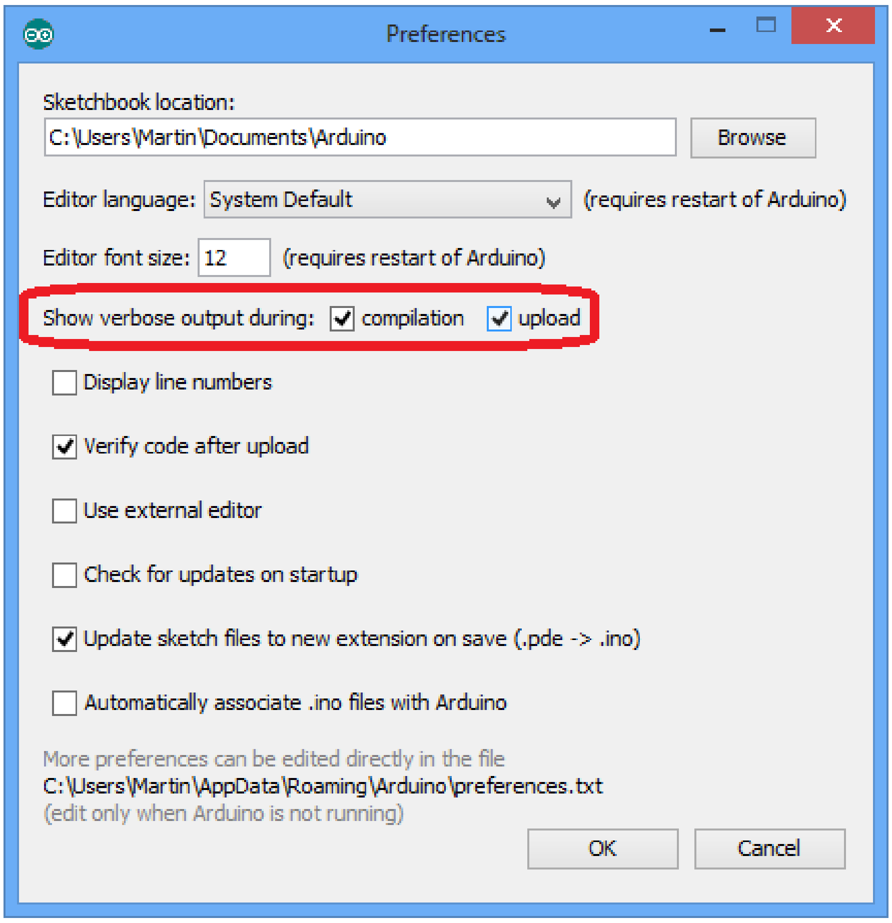
The bar appearing on the top is called **Menu Bar** that comes with five different options as follow

- **File** – You can open a new window for writing the code or open an existing one. Following table shows the number of further subdivisions the file option is categorized into.



```
e second, then off for one second, repe

board LED you can control. On the UNO,
 pin 13, on MKR1000 on pin 6. LED_BUILI
endent of which board is used.
pin the on-board LED is connected to or
ur board  at https://www.arduino.cc/en/

he public domain.
```

As you go to the preference section and check the compilation section, the Output Pane will show the code compilation as you click the upload button.



Sketchbook location: C:\Users\Martin\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12 (requires restart of Arduino)

Show verbose output during: ✓ compilation ✓ upload

☐ Display line numbers

✓ Verify code after upload

☐ Use external editor

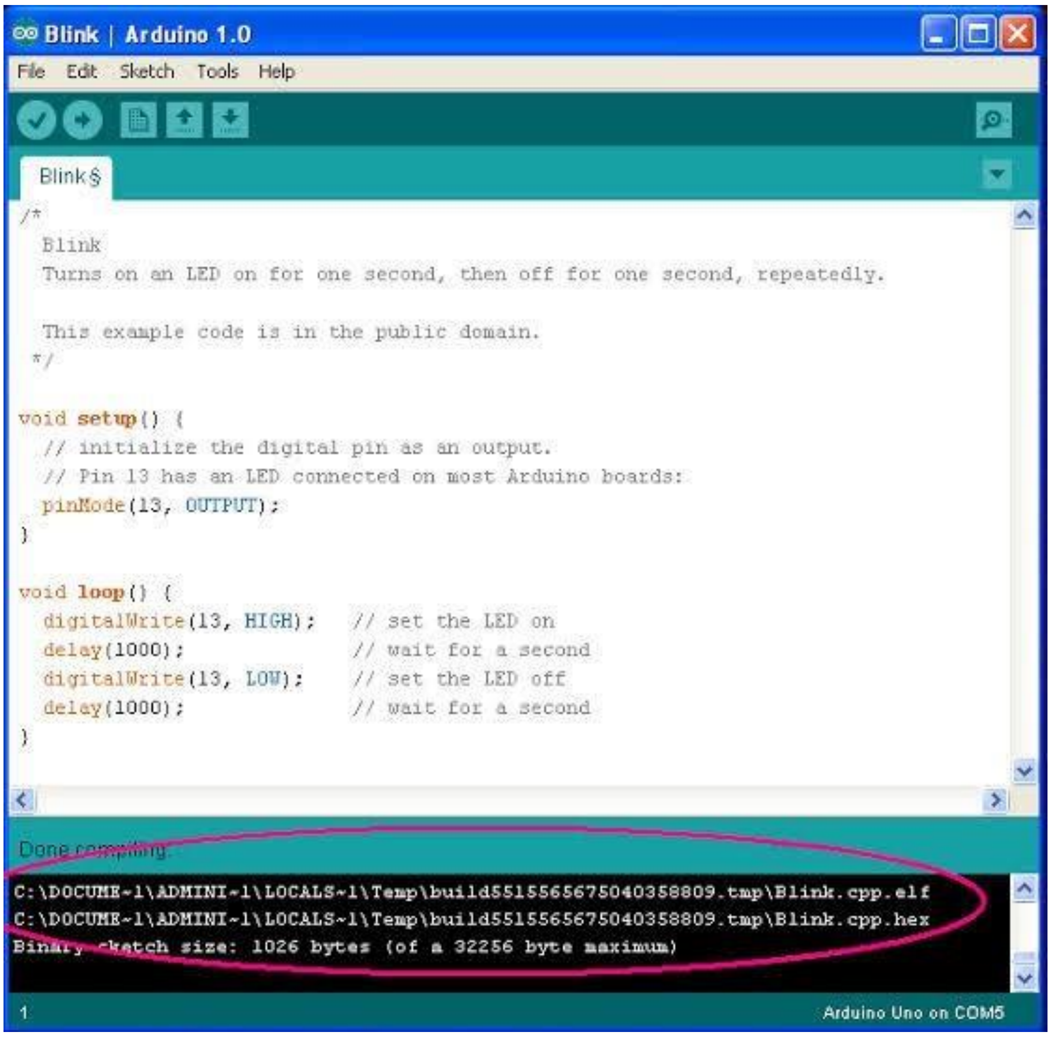
☐ Check for updates on startup

✓ Update sketch files to new extension on save (.pde -> .ino)

☐ Automatically associate .ino files with Arduino

More preferences can be edited directly in the file
C:\Users\Martin\AppData\Roaming\Arduino\preferences.txt
(edit only when Arduino is not running)

And at the end of compilation, it will show you the hex file it has generated for the recent sketch that will send to the Arduino Board for the specific task you aim to achieve.

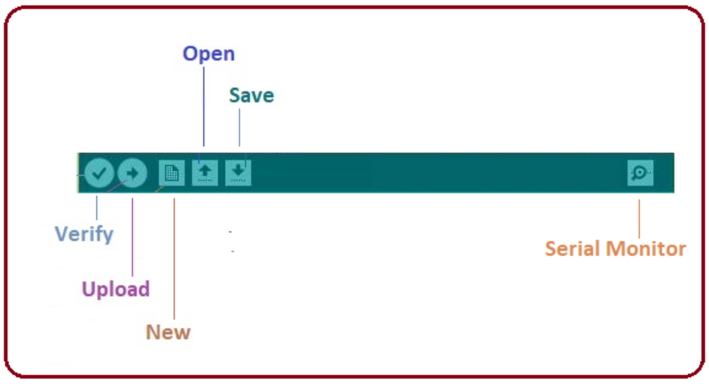


```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

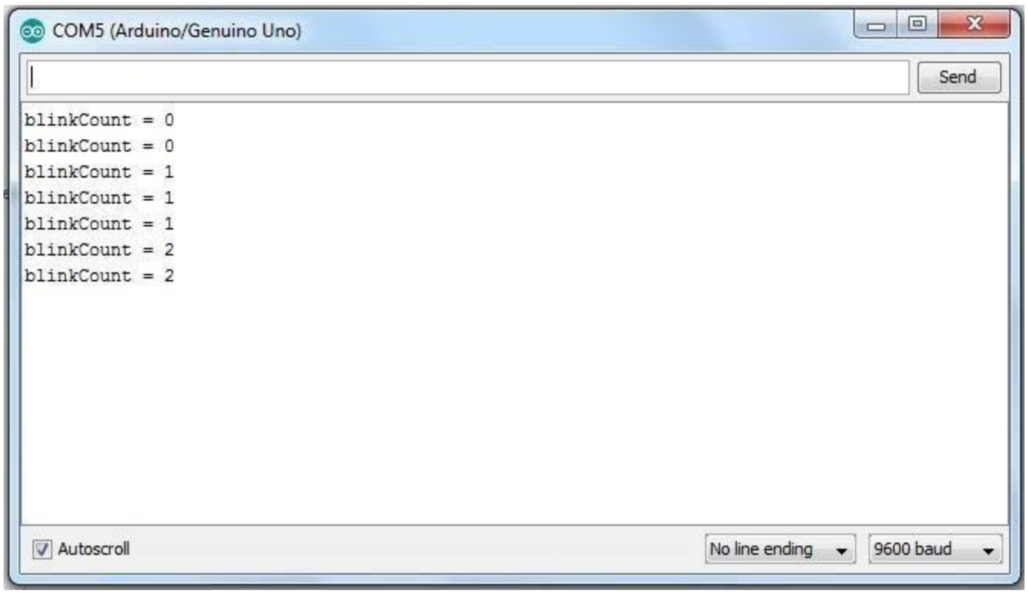
```
Done compiling.
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\build5515565675040358809.tmp\Blink.cpp.elf
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\build5515565675040358809.tmp\Blink.cpp.hex
Binary sketch size: 1026 bytes (of a 32256 byte maximum)
```

- **Edit** – Used for copying and pasting the code with further modification for font
- **Sketch** – For compiling and programming
- **Tools** – Mainly used for testing projects. The Programmer section in this panel is used for burning a bootloader to the new microcontroller.
- **Help** – In case you are feeling skeptical about software, complete help is available from getting started to troubleshooting.

The **Six Buttons** appearing under the Menu tab are connected with the running program as follow.

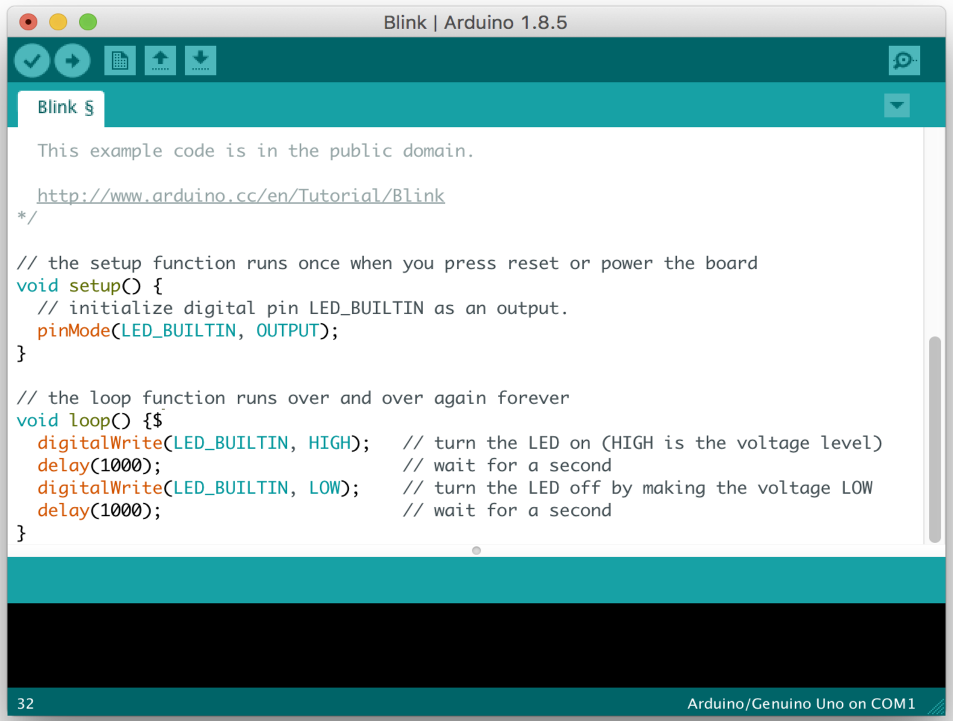


- The check mark appearing in the circular button is used to verify the code. Click this once you have written your code.
- The arrow key will upload and transfer the required code to the Arduino board.
- The dotted paper is used for creating a new file.
- The upward arrow is reserved for opening an existing Arduino project.
- The downward arrow is used to save the current running code.
- The button appearing on the top right corner is a **Serial Monitor** – A separate pop-up window that acts as an independent terminal and plays a vital role for sending and receiving the Serial Data. You can also go to the Tools panel and select Serial Monitor, or pressing Ctrl+Shift+M all at once will open it instantly. The Serial Monitor will actually help to debug the written Sketches where you can get a hold of how your program is operating. Your Arduino Module should be connected to your computer by USB cable in order to activate the Serial Monitor.
- You need to select the baud rate of the Arduino Board you are using right now. For my Arduino Uno Baud Rate is 9600, as you write the following code and click the Serial Monitor, the output will show as the image below.



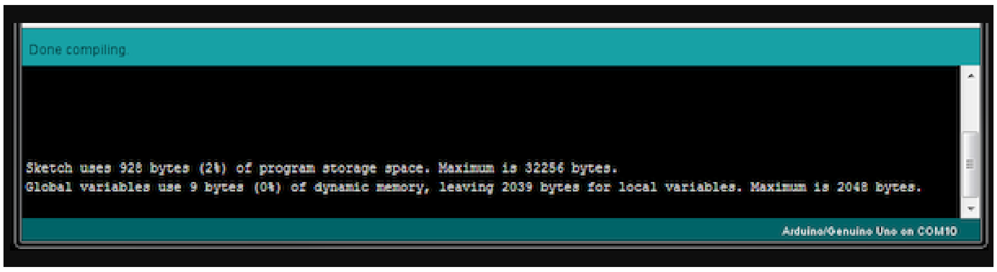
```
blinkCount = 0
blinkCount = 0
blinkCount = 1
blinkCount = 1
blinkCount = 1
blinkCount = 2
blinkCount = 2
```

The main screen below the Menu bard is known as a simple text editor used for writing the required code.



```
  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {$
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

The bottom of the main screen is described as an Output Pane that mainly highlights the compilation status of the running code: the memory used by the code, and errors occurred in the program. You need to fix those errors before you intend to upload the hex file into your Arduino Module.

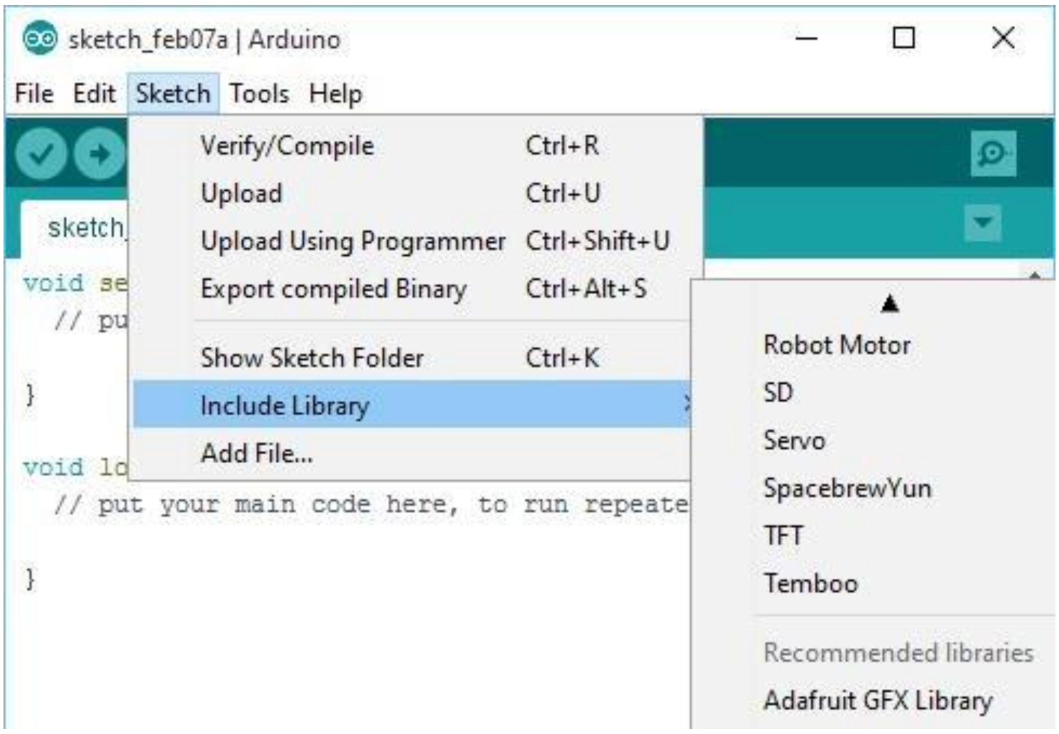


```
Done compiling.

Sketch uses 928 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.
```

More or less, Arduino C language works similar to the regular C language used for any embedded system microcontroller, however, there are some dedicated libraries used for calling and executing specific functions on the board.

Libraries:

Libraries are very useful for adding the extra functionality into the Arduino Module. There is a list of libraries you can add by clicking the Sketch button in the menu bar and going to Include Library.



```
void se
  // pu
}

void lo
  // put your main code here, to run repeate
}
```

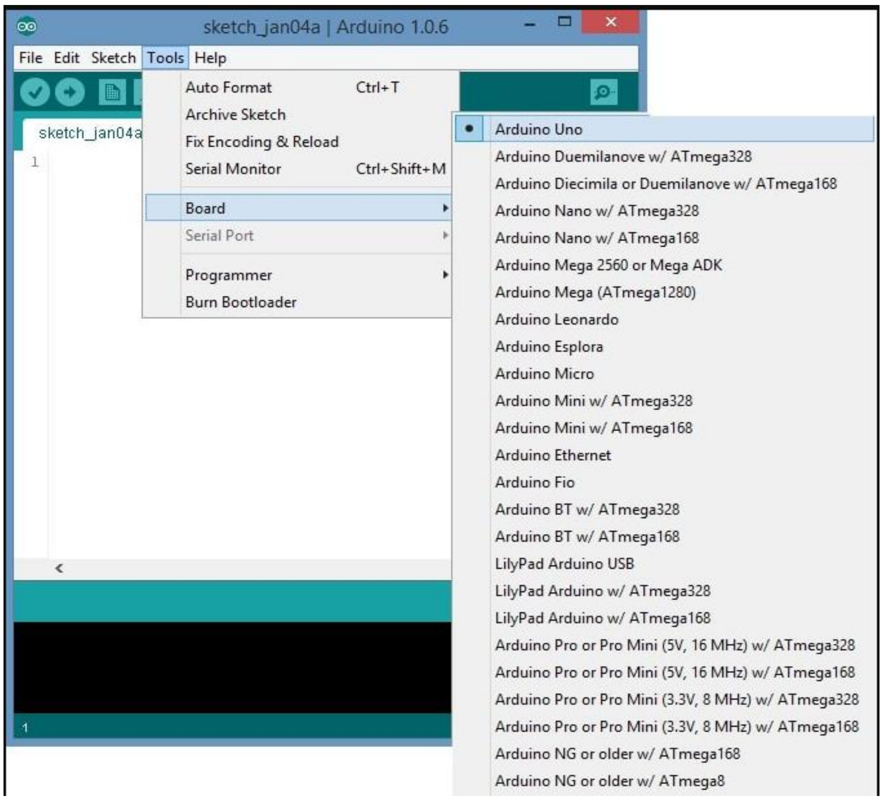
As you click the Include Library and Add the respective library it will on the top of the sketch with a #include sign. Suppose, I Include the EEPROM library, it will appear on the text editor as
#include <EEPROM.h>.
Most of the libraries are preinstalled and come with the Arduino software. However, you can also download them from the external sources.

Making pins Input and output:

The digitalRead and digitalWrite commands are used for addressing and making the Arduino pins as an input and output respectively.
These commands are text sensitive i.e. you need to write them down the exact way they are given like digitalWrite starting with small "d" and write with capital "W". Writing it down with Digitalwrite or digitalwrite won't be calling or addressing any function.

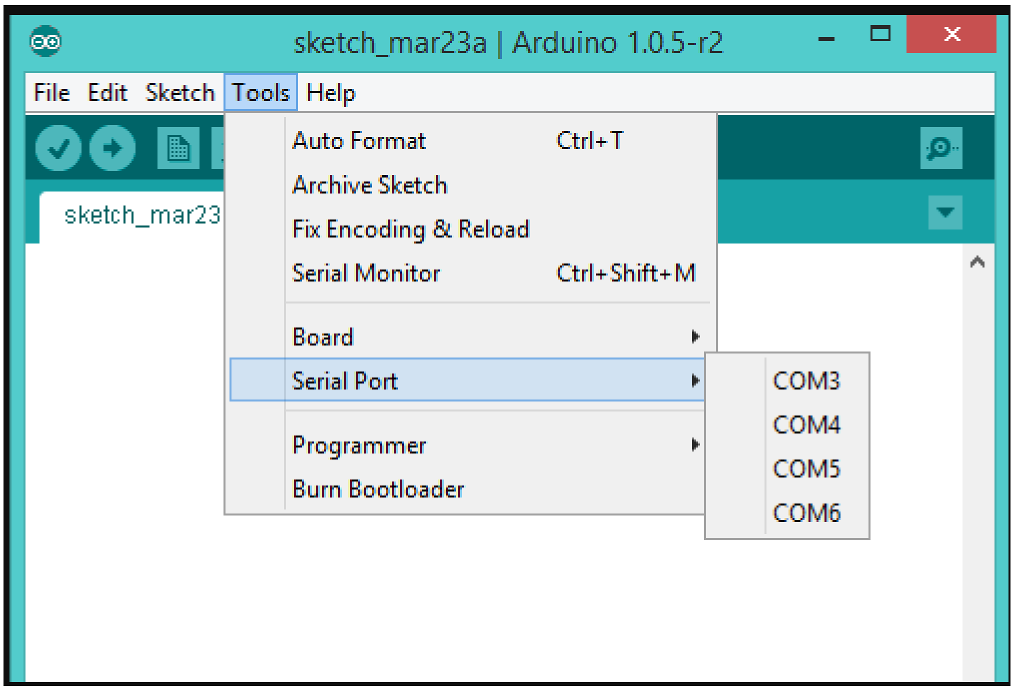
How to select the board:

In order to upload the sketch, you need to select the relevant board you are using and the ports for that operating system. As you click the Tools on the Menu, it will open like the figure below.


sketch_jan04a | Arduino 1.0.6
File Edit Sketch Tools Help
sketch_jan04a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
● Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8

- Just go to the “Board” section and select the board you aim to work on. Similarly, COM1, COM2, COM4, COM5, COM7 or higher are reserved for the serial and USB board. You can look for the USB serial device in the ports section of the Windows Device Manager.

Following figure shows the COM4 that I have used for my project, indicating the Arduino Uno with COM4 port at the right bottom corner of the screen.



- After correct selection of both Board and Serial Port, click the verify and then upload button appearing in the upper left corner of the six button section or you can go to the Sketch section and press verify/compile and then upload.
- The sketch is written in the text editor and is then saved with the file extension .ino.

  It is important to note that the recent Arduino Modules will reset automatically as you compile and press the upload button the IDE software, however, older version may require the physical reset on the board.

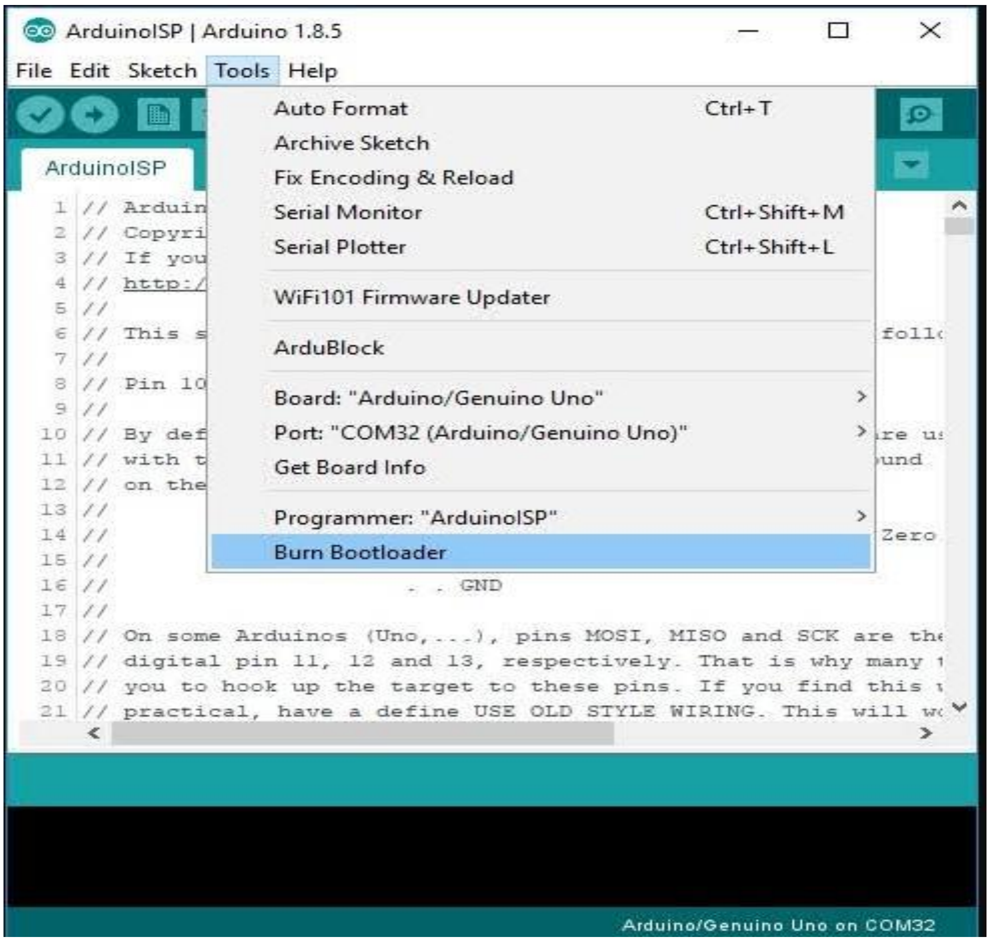
- Once you upload the code, TX and RX LEDs will blink on the board, indicating the desired program is running successfully.

Note: The port selection criteria mentioned above is dedicated for Windows operating system only, you can check this Guide if you are using MAC or Linux.

- The amazing thing about this software is that no prior arrangement or bulk of mess is required to install this software, you will be writing your first program within 2 minutes after the installation of the IDE environment.

BootLoader:

As you go to the Tools section, you will find a bootloader at the end. It is very helpful to burn the code directly into the controller, setting you free from buying the external burner to burn the required code.


ArduinoISP | Arduino 1.8.5
File Edit Sketch Tools Help
ArduinoISP
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
ArduBlock
Board: "Arduino/Genuino Uno"
Port: "COM32 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "ArduinoISP"
Burn Bootloader
Arduino/Genuino Uno on COM32

```
// Arduin
// Copyri
// If you
// http:/
//
// This s                                           follo
//
// Pin 10
//
// By def                                           re us
// with t                                           ound
// on the
//
//                                                  Zero
//
//                             . . GND
//
// On some Arduinos (Uno,...), pins MOSI, MISO and SCK are the
// digital pin 11, 12 and 13, respectively. That is why many t
// you to hook up the target to these pins. If you find this w
// practical, have a define USE OLD STYLE WIRING. This will w
```

When you buy the new Arduino Module, the bootloader is already installed inside the controller. However, if you intend to buy a controller and put in the Arduino module, you need to burn the bootloader again inside the controller by going to the Tools section and selecting the burn bootloader.

CHAPTER 4

RESULTS

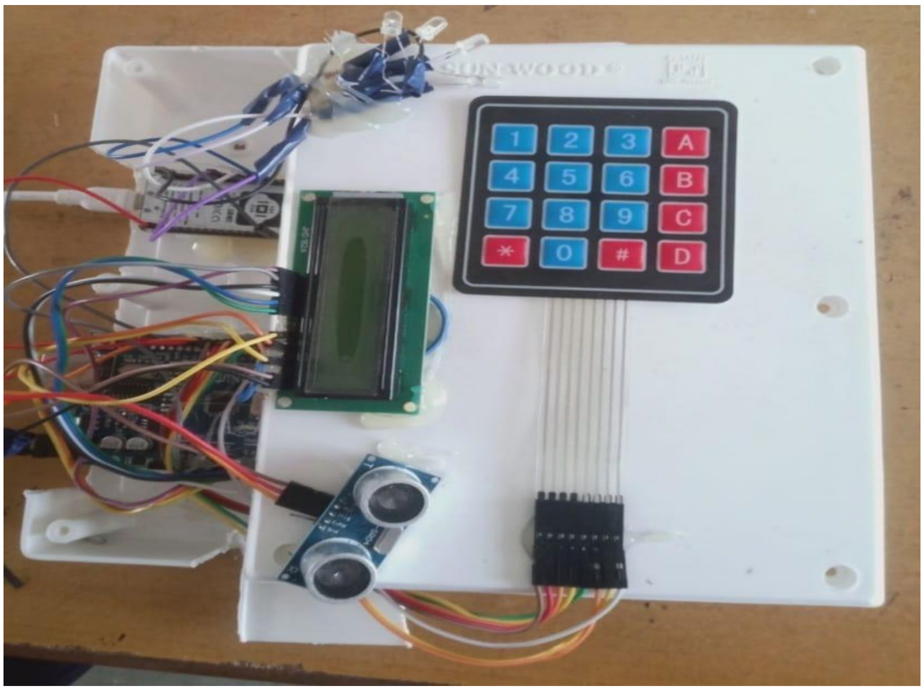


Fig 4.1: Connection of components

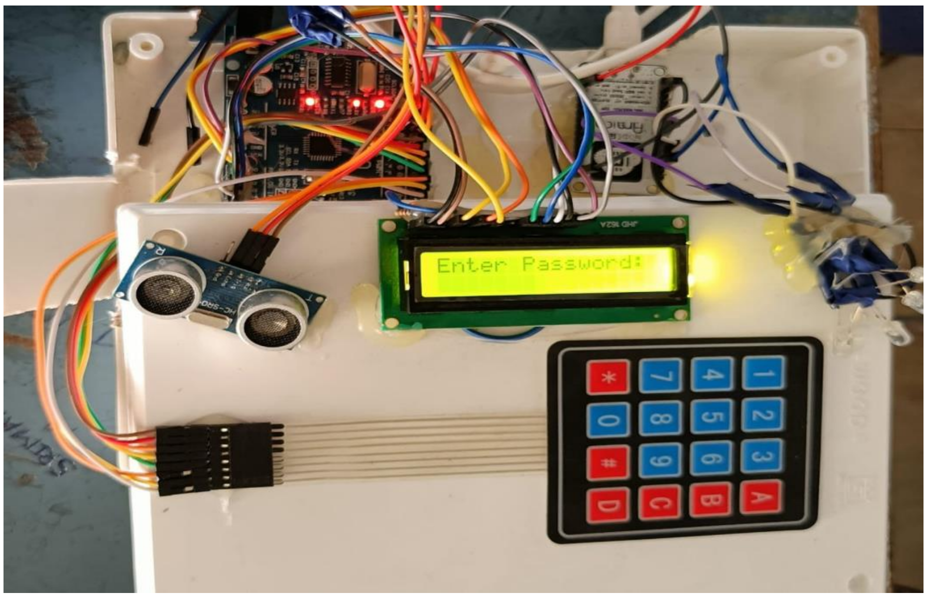


Fig 4.2: Entering of password

CHAPTER 5

CONCLUSION

This project presented is a low cost and flexible home control and monitoring system using Node MCU Board with internet and various sensors remotely controlled by Android OS smart phone. In this, Node MCU micro controller is used as an interface between user and hardware components. It is programmed and connected to several components according to the requirements. A micro web server is used as an application layer for communication between remote users and home devices, security systems. This entire system communication is enabled through internet. Notifications are sent to user through the app BLYNK installed in smart phone. User can operate wirelessly or home appliances can be automated by using several sensors like temperature sensor, LDR etc. All these together forms a complete capable, flexible smart home control and monitoring system, based on IOT technology.

CHAPTER 6

FUTURE SCOPE

More smartness can be added to this proposed project for making this smart home highly automated by using artificial intelligence. A camera can also be connected to micro controller so that suspect photograph can be taken and can be forwarded to the police if needed. Also voice call feature can be included to this system through which user can control the home appliances.

REFERENCES

- Vishwajeet Hari Bhide, Dr.SanjeevWagh," i-Learning loT: An Intelligent Self Learning System for Home Automation Using loT", International Conference on Communication and Signal Processing,, 2015.

- Ian G smith, "The Internet of things" NewHorizons, IERC-Internet of things European Research cluster, 2012.

- Vikram.N, Harish.k, Nihaal.M, Raksha umesh ,ShettyAashik Ashok kumar, "A Low Cost Home Automation System Using Wi-Fi Based Wireless Sensor Network Incorporating Internet of Things (IoT)", IEEE 7th International Advance Computing Conference, pp. 174-178, 2017.

- Yong Tae Park, PraneshSthapit, Jae-Young Pyun, "Smart Digital Door Lock for the Home Automation", IEEE, pp. 1-6, TENCON 2009.

- Himanshu Singh, Vishal Pallagani†, Vedant Khandelwal, Venkanna U. "IoT based Smart Home Automation System using Sensor Node", 2018.

- Smart Energy Efficient Home Automation System using IOT by Satyendra K. Vishwakarma, Prashant Upadhyaya, Babita Kumari, Arun Kumar Mishra