PROJECT SYNOPSIS

ON

IOT Automatic Watering System for Plants

Bachelor of Technology

COMPUTER SCIENCE & ENGINEERING

BY

Abhiraj Sharma

18-CSE-006

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
ECHELON INSTITUTE OF TECHNOLOGY, FARIDABAD

CERTIFICATE

I/We hereby certify that the work which is being presented in the B.Tech. Project Report entitled, IOT Automatic Watering System for Plants in partial fulfilment of the requirements for the award of the Bachelor of Technology in Computer Science & Engineering and submitted to the Department of Computer Science & Engineering of Echelon Institute of Technology, Faridabad is an authentic record of my own work carried out during a period from Jan 2022 to May 2022 under the supervision of **Ms. Suman Chandila.**

The matter presented in this thesis has not been submitted by me for the award of any other degree elsewhere.

Signature of Candidates

Abhiraj Sharma 18-CSE-006

Under The Guidance

Ms. Suman Chandila

Assistant Professor

CSE Department

ABSTRACT

This Report is prepared during training. it was life's greatest learning experience, gaining knowledge and practical implementation. This period also provided me a chance to give theoretical knowledge a Practical shape. Most importantly I have been given the exposure to the latest technology in the Programming World . This report is a result of the final project that I did in my college as a student gave me a decent platform in the beginning of my professional career. I wholeheartedly thank the organization for giving me an opportunity to work on the latest technology & for bringing out the hat in me and developing my talents & improving my skills. not only in the technical field but also in my overall personality. The mat important V utile is that. I learned how to work in a team and complete task as a team. Cooperating and assisting each other helped me to exploit my potential & perform much better.

Index

Introduction

This is an IoT based Automatic Watering System for Plants using Arduino microcontroller. With the help of WIFI, the moisture and other data can be monitored from anywhere around the world.

Plants need sufficient water to grow well. Watering plants is a work that needs to be done by farmers and plant lovers in caring for plants. Watering plants with suitable water volume is important because it has a direct impact on plants. Lack of water or excessive water content can make plants dry or rotten

Soil moisture related to water content which a factor that affects the plant growth. The process of watering plants is generally done manually regardless of the volume of water needed by plants. This research discussed about an automated prototype and a system that have the function of watering plants based on the soil moisture level. The method used is prototyping which is suitable with the research purpose. The prototype and systems built with microcontroller, soil moisture sensors, relay and solenoid valve, which integrated with the IoT platform Blynk apps and ThingSpeak. The process starts from the detection of soil moisture by the sensor. If soil moisture value is detected on 30% - 35%, then the device activates the watering function by opening the valve from the solenoid valve to drain water to the pipe. When the soil moisture detected more then 35%, the device stops the watering function. ThingSpeak IoT platform, used to display moisture percentage data in graphical form. Blynk apps provide notification features to the user's smartphone when the watering device is activated or deactivated. Based on the test scenario performed, it was found that the percentage of soil moisture with an initial value of 30% - 35% increased to 68.2%, after the watering process. Each component of the device and system has been tested and functioning according to the purpose, so the system has the potential to be used in the process of watering the plants automatically.

Software and Hardware requirements

Software required:

- Blynk App for demo app creation
- Android Studio for final version
- Arduino IDE

Hardware required:

- NodeMCU ESP8266 microcontroller
- Soil Moisture Sensor
- Relay Module
- Solenoid Water Valve
- Breadboard
- Jumper Wires
- 12V Battery

About NodeMCU

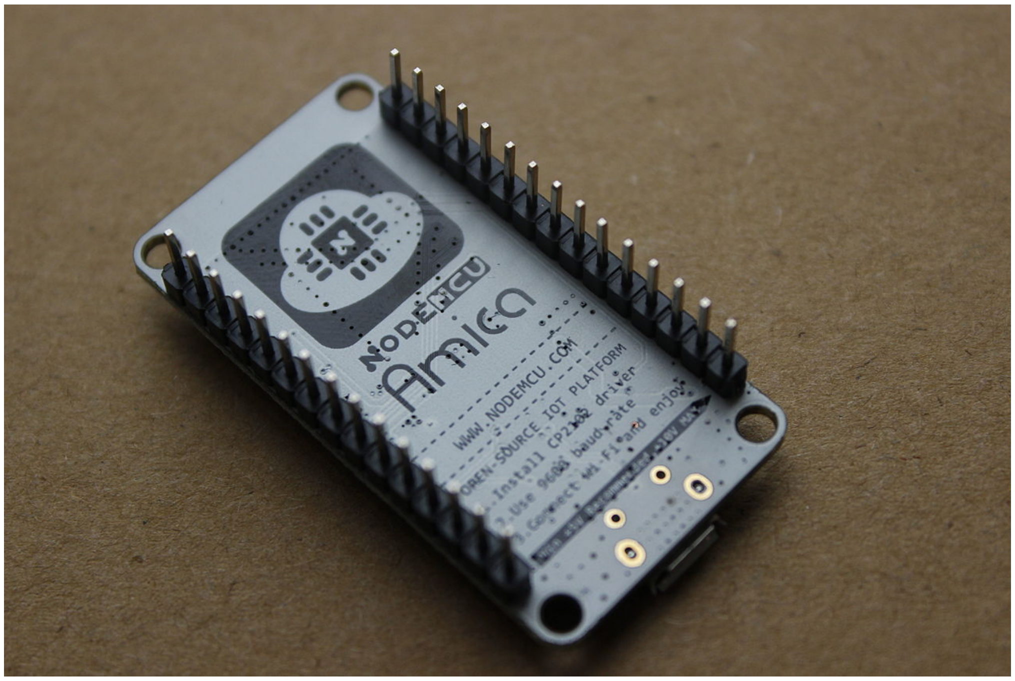
NodeMCU is a low-cost open source IoT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which was based on the ESP-12 module. Later, support for the ESP32 32-bit MCU was added.

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.

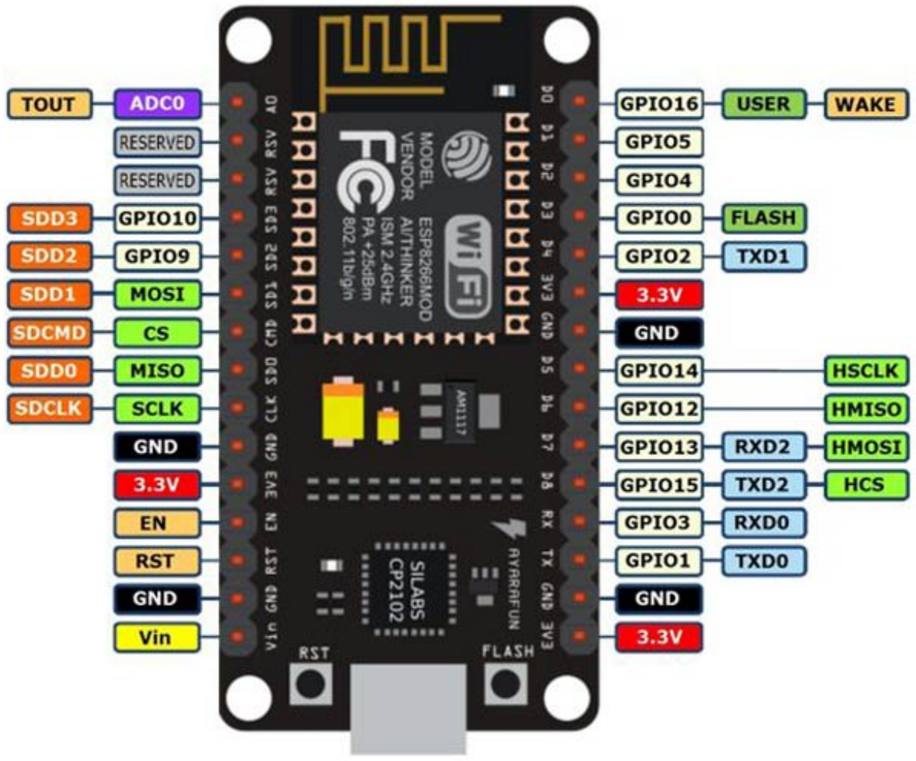
The firmware uses the Lua scripting language. The firmware is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build 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. The design was initially based on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications (see related projects).



NodeMCU provides access to the GPIO (General Purpose Input/Output) and a pin mapping table is part of the API documentation.

I/O index	ESP8266 pin
0 [*]	GPIO16
1	GPIO5
2	GPIO4
3	GPIO0
4	GPIO2
5	GPIO14
6	GPIO12
7	GPIO13
8	GPIO15
9	GPIO3
10	GPIO1
11	GPIO9
12	GPIO10



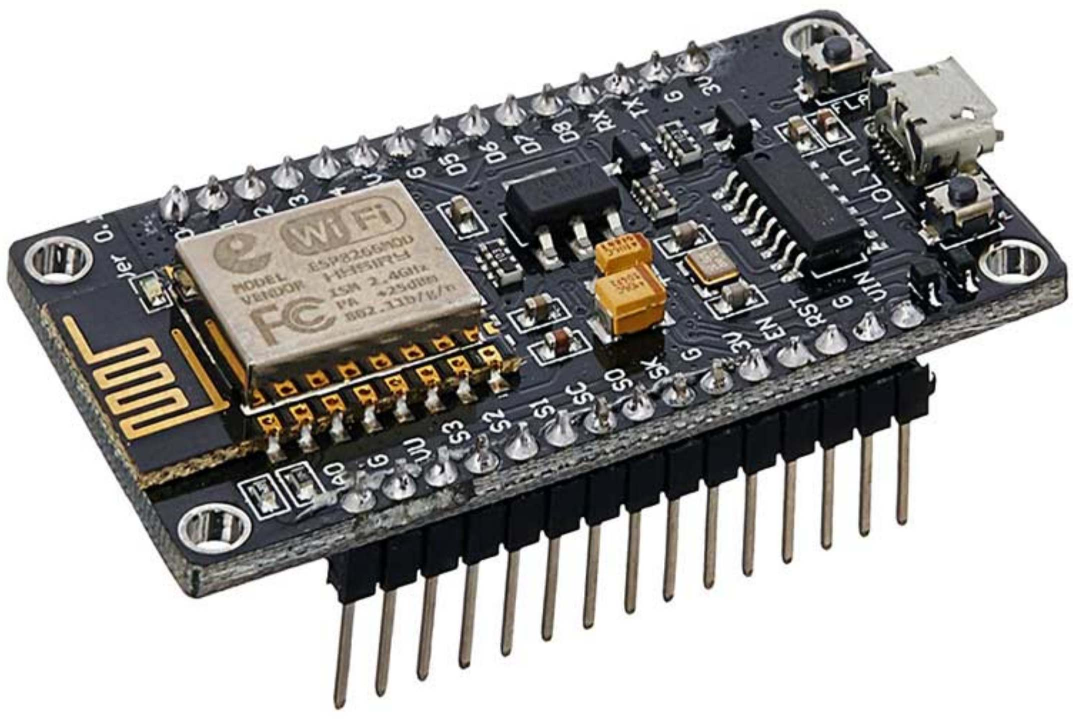
The **NodeMCU ESP8266 development board** comes with the ESP-12E module containing the ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects.

NodeMCU can be powered using a Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.

Programming NodeMCU ESP8266 with Arduino IDE

The NodeMCU Development Board can be easily programmed with Arduino IDE since it is easy to use.

Programming NodeMCU with the Arduino IDE will hardly take 5-10 minutes. All you need is the Arduino IDE, a USB cable and the NodeMCU board itself. You can check this Getting Started Tutorial for NodeMCU to prepare your Arduino IDE for NodeMCU.



NodeMCU Development Board Pinout Configuration

Pin Category	Name	Description
Power	Micro-USB, 3.3V, GND, Vin	**Micro-USB:** NodeMCU can be powered through the USB port **3.3V:** Regulated 3.3V can be supplied to this pin to power the board **GND:** Ground pins **Vin:** External Power Supply
Control Pins	EN, RST	The pin and the button resets the microcontroller
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but

		due to the internal functionality of these pins, you have to find which pin is I2C.

NodeMCU ESP8266 Specifications & Features

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects

About soil moisture sensor

The Soil Moisture Sensor measures soil moisture grace to the changes in electrical conductivity of the earth (soil resistance increases with drought).The electrical resistance is measured between the two electrodes of the sensor. A comparator activates a digital output when a adjustable threshold is exceeded.

At the time we start writing the code, we define 3 variables :

```
int sensorPin = A0;
```

- The first one defines the analog pin of the Arduino

```
int sensorValue;
```

- The second defines the analog value of the sensor read by the Arduino

```
int limit = 300;
```

- The third defines a limit (in this case if the sensorValue is larger than the limit, then a LED will light up)

```
void setup() {
Serial.begin(9600);
pinMode(13, OUTPUT);
}
```

After, we initialize the serial monitor by indicating the number of baud (here 9600) and also the thirteenth pin of the Arduino (LED pin) by indicating a current output.

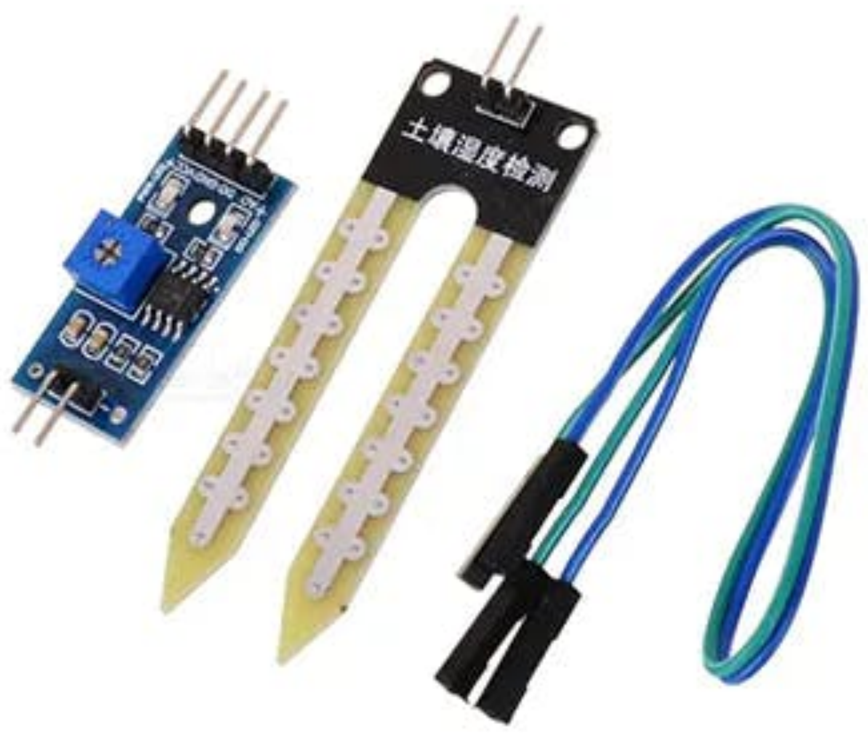
```
void loop() {
sensorValue = analogRead(sensorPin);
```

We define the sensorValue as being the value read by the Arduino.

```
Serial.println("Analog Value : ");
Serial.println(sensorValue);
```

We display the sensorValue on the serial monitor.

```
if (sensorValue<limit) {
digitalWrite(13, HIGH);
}
else {
digitalWrite(13, LOW);
 }
delay(1000);
}
```



Connections

- *Arduino --> Comparator*

  3V --> VCC

  GND --> GND

  A0 --> A0

- *Comparator --> Sensor*

  + --> +

  - --> -

- *Arduino --> LED*

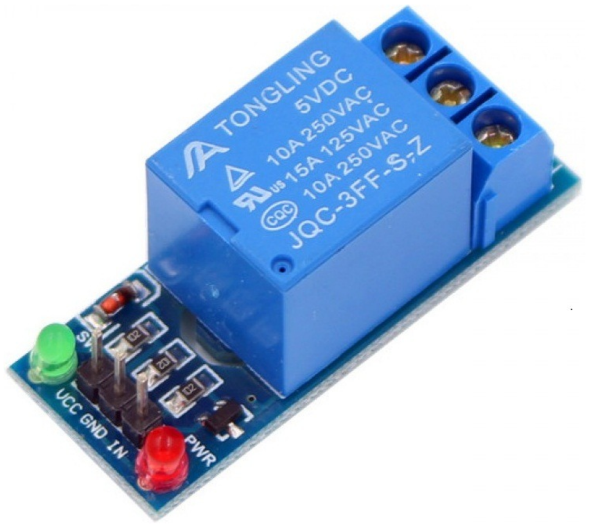
D13 --> +

GND --> -

Connect a resistance between + and - of the LED.

About Relay Module

The relay is the device that open or closes the contacts to cause the operation of the other electric control. It detects the undesirable condition with an assigned area and gives the commands to the circuit breaker to disconnect the affected area through ON or OFF.
Every electromechanical relay consists of
1. Electromagnet
3. Mechanically movable contact
3. Switching points and
4. Spring



COM**:** common pin
NO**:** Normally open – there is no contact between the common pin and the normally open pin. So, when you trigger the relay, it connects to the COM pin and power is provided to the load.
NC**:** Normally closed – there is contact between the common pin and the normally closed pin. There is always connection between the COM and NC pins, even when the relay is turned off. When you trigger the relay, the circuit is opened and there is no supply provided to the load.

Working principle of relay:
It works on the principle of an electromagnetic attraction. When the circuit of the relay senses the fault current, it energises the electromagnetic field which produces the temporary magnetic field. This magnetic field moves the relay armature for opening or closing the connections. The small power relay has only one contacts, and the high power relay has two contacts for opening the switch.
The inner section of the relay is shown in the figure below. It has an iron core which is wound by a control coil. The power supply is given to the coil through the contacts of the load and the control switch.

The current flows through the coil produces the magnetic field around it.
Due to this magnetic field, the upper arm of the magnet attracts the lower arm. Hence close the circuit, which makes the current flow through the load. If the contact is already closed, then it moves oppositely and hence open the contacts.
Types of Relay Based on the principle of operation
1. Electrothermal relay:
2. Electromechanical relay:
3. Solid State relay:
4. Hybrid relay:

Applications of relay:
A. They can be used for both ac and dc systems for protection of ac and dc equipment's
B. Electromagnetic relays operating speeds which has the ability to operate in milliseconds are also can be possible
C. They have the properties such as simple, robust, compact and most reliable
D. These relays are almost instantaneous. Though instantaneous the operating time of the relay varies with the current. With extra arrangements like dashpot, copper rings.
E. Electromagnetic relays have fast operation and fast reset

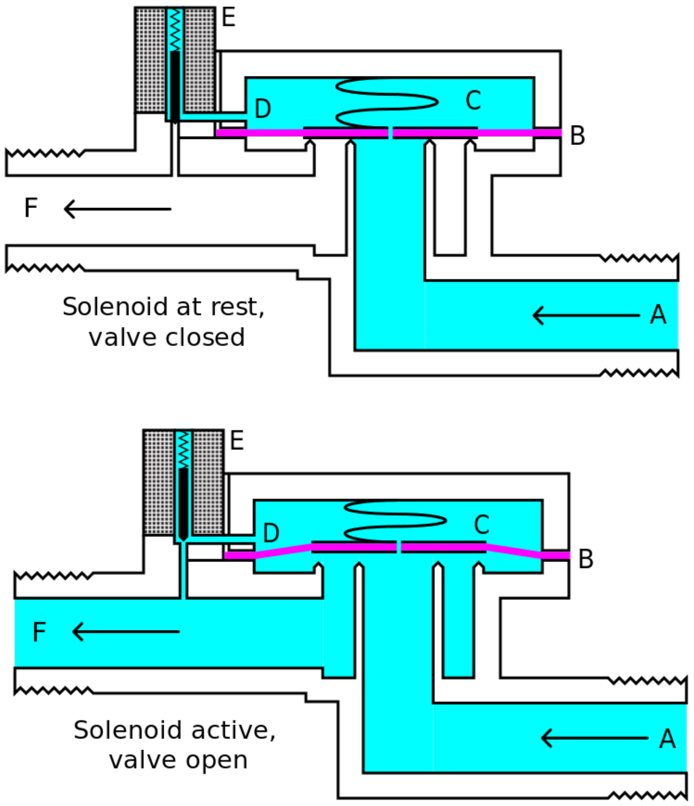
Disadvantages:
a. High burden level instrument transformers are required (CTs and PTs of high burden is required for operating the electromagnetic relays compared to static relays)
b. The directional feature is absent in electromagnetic relays
c. Requires periodic maintenance and testing unlike static relays
d. Relay operation can be affected due to ageing of the components and dust, pollution resulting in spurious trips
e. Operation speed for an electromagnetic relays is limited by the mechanical inertia of the component.

About Solenoid Water Valve

A solenoid valve is an electromechanically operated valve.

Solenoid valves differ in the characteristics of the electric current they use, the strength of the magnetic field they generate, the mechanism they use to regulate the fluid, and the type and characteristics of fluid they control. The mechanism varies from linear action, plunger-type actuators to pivoted-armature actuators and rocker actuators. The valve can use a two-port design to regulate a flow or use a three or more port design to switch flows between ports. Multiple solenoid valves can be placed together on a manifold.

Solenoid valves are the most frequently used control elements in fluidics. Their tasks are to shut off, release, dose, distribute or mix fluids. They are found in many application areas. Solenoids offer fast and safe switching, high-reliability, long service life, good medium compatibility of the materials used, low control power and compact design.



While there are multiple design variants, the following is a detailed breakdown of a typical pilot-operated solenoid valve. They may use metal seals or rubber seals, and may also have electrical interfaces to allow for easy control.

The diagram on the top shows the design of a basic valve, controlling the flow of water in this example. The top half shows the valve in its closed state. An inlet stream of pressurized water enters at A. B is an elastic diaphragm and above it is a spring pushing it down. The diaphragm has a pinhole through its centre which allows a very small amount of water to flow through. This water fills cavity C so that pressure is roughly equal on both sides of the diaphragm. However, the pressurized water in cavity C acts across a much greater area of the diaphragm than the water in inlet A. From the equation $F = P*A$, the force from cavity C pushing downward is greater than the force from inlet A pushing upward, and the diaphragm remains closed.

Diaphragm B will stay closed as long as small drain passage D remains blocked by a pin, which is controlled by solenoid E. In a normally closed valve, supplying an electric current to the solenoid will raise the pin via magnetic force, and the water in cavity C drains out through passage D faster than the pinhole can refill it. Less water in cavity C means the pressure on that side of the diaphragm drops, proportionately dropping the force too. With the downward force of cavity C now less than the upward force of inlet A, the diaphragm is pushed upward, thus opening the valve. Water now flows freely from A to F. When the solenoid is deactivated and passage D is closed, water once again accumulates in cavity C, closing the diaphragm once the downward force exerted is great enough.

This process is the opposite for a normally open pilot-operated valve. In that case, the pin is naturally held open by a spring, passage D is open, and cavity C is never able to fill up enough, pushing open diaphragm B and allowing unobstructed flow. Supplying an electric current to the solenoid pushes the pin into a closed position, blocking passage D, allowing water to accumulate in cavity C, and ultimately closing diaphragm B.

In this way, a pilot-operated solenoid valve can be conceptualized as two valves working together: a direct-acting solenoid valve which functions as the "brain" to direct the "muscle" of a much more powerful main valve which gets actuated pneumatically or hydraulically. This is why pilot-operated valves will not work without a sufficient pressure differential between input and output, the "muscle" needs to be strong enough to push back against the diaphragm and open it. Should the pressure at the output rise above that of the input, the valve would open regardless of the state of the solenoid and pilot valve.

12V DC Solenoid Water Air Valve Switch (Normally Closed) – 1/2″ controls the flow of fluid (liquid or air) and acts as a valve between high-pressure fluid! This liquid valve would make a great addition to your robotic gardening project. There are two ½" (Nominal NPT) outlets. Normally, the valve is closed. When a 12V DC supply is applied to the two terminals, the valve opens and water can push through.

The valve works with the solenoid coil which operates electronically with DC 12 volt supply. As it is a normally closed assembly, it opens the flow of fluids as soon as it is powered ON and stops/blocks the flow when the supply voltage removed.

Features:

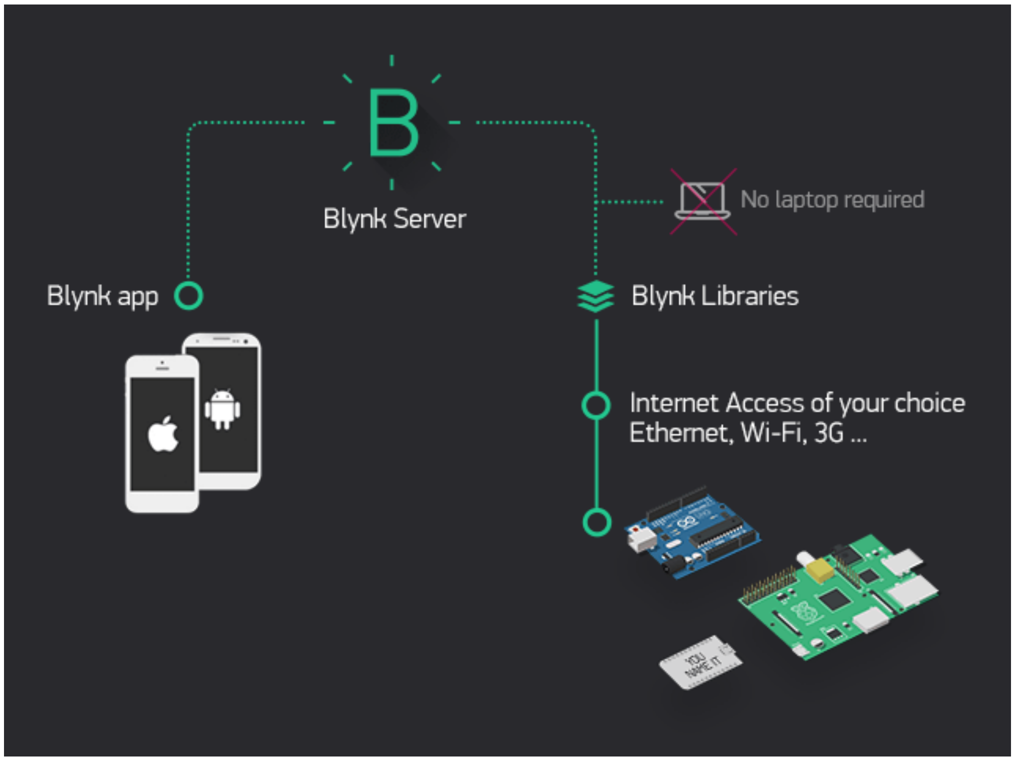
1. Compact and convenient.
2. Easily installed and serviced
3. Precise and reliable.
4. The installation direction can be arbitrary Angle
5. Pressure-regulating valve (steady flow valve) function similar to the water flow switch
6. Not only can it provide a steady flow, but it also prevents the dry burning.
7. Suitable for fluids like water, oil, air.

About Blynk application

Blynk is an Internet of things (IoT) company which provides a platform for building mobile (IOS and Android) applications that can connect electronic devices to the Internet and remotely monitor and control these devices. **Blynk** was founded by Pavel Bayborodin, a user experience (UX) expert in mobile and automotive space. The IoT platform was launched in 2014.

Blynk platform is used by engineers to connect MCUs and prototyping development boards like Arduino, ESP8266 or SBCs like Raspberry Pi over Wi-Fi, Ethernet or the cellular to the Internet and build custom mobile applications to remotely monitor and control electronic equipment. **Blynk Cloud** is open-source

Examples of platform applications are Smart Home, environmental monitoring, industrial equipment remote control. With Blynk, you can create smartphone applications that allow you to easily interact with microcontrollers or even full computers such as the Raspberry Pi. The main focus of the Blynk platform is to make it super-easy to develop the mobile phone application. As you will see in this course, developing a mobile app that can talk to your Arduino is as easy as dragging a widget and configuring a pin. With Blynk, you can control an LED or a motor from your mobile phone with literally zero programming. This is actually the first experiment that I will demonstrate in this course.



But don’t let this simplicity make you think that Blynk is only useful for trivial applications. Blynk is a robust and scalable tool that is used by hobbyists and the industry alike.You can use it to monitor the soil humidity of your vegetable garden and turn on the water, or open up your garage door, with your phone. You can also use it to control smart furniture that can learn from your routines, or embed IoT and AI to traditional industrial products such as a boiler, or for improving the integrity and safety of oilfields.

Blynk is free to use for personal use and prototyping. Their business model generates profits by selling subscriptions to businesses that want to publish Blynk-powered apps for their hardware products or services.

About Arduino IDE

Arduino IDE(Integrated Development Environment) is the software for Arduino.

It is a text editor like a notepad with different features.

It is used for writing code, compiling the code to check if any errors are there and uploading the code to the Arduino.

It is a cross-platform software which is available for every Operating System like Windows, Linux, macOS.

It supports C/C++ language

It is open-source software, where the user can use the software as they want it to. They can also make their own modules/functions and add them to the software

It supports every available Arduino board including Arduino mega, Arduino Leonardo, Arduino Ethernet and more

Word file is called a Document similarly, Arduino file is called a **Sketch** where the user writes code.

The format of Arduino is saved as **.ino**

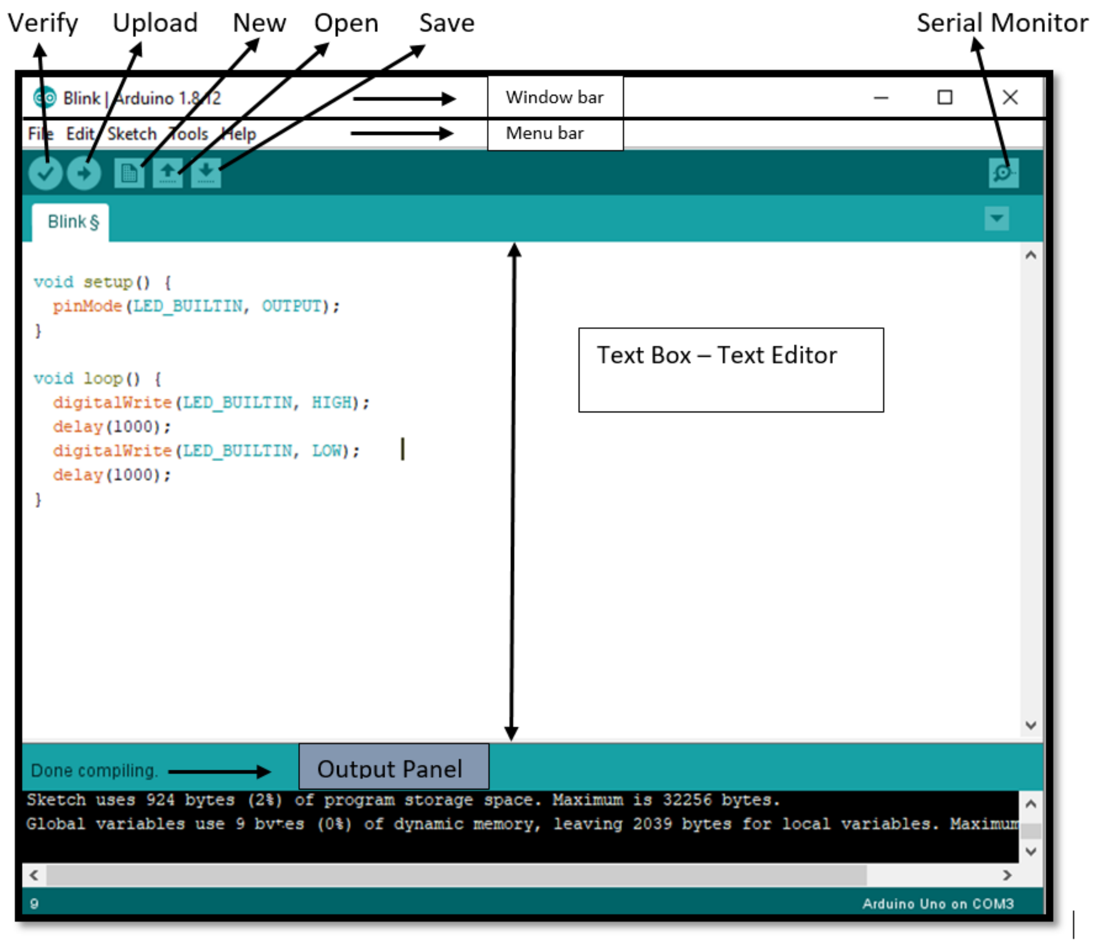
How Arduino IDE works?

When a user writes code and compiles, the IDE will generate a Hex file for the code**. (Hex file are Hexa Decimal files which are understood by Arduino)** and then sent to the board using a USB cable. Every Arduino board is integrated with a microcontroller, the microcontroller will receive the hex file and runs as per the code written.

Functions of Arduino IDE:

Arduino IDE consists of different sections

1. WindowBar
2. MenuBar
3. ShortcutButtons
4. Text Editor
5. Output Panel



```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

```
Sketch uses 924 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
```

Window Bar:

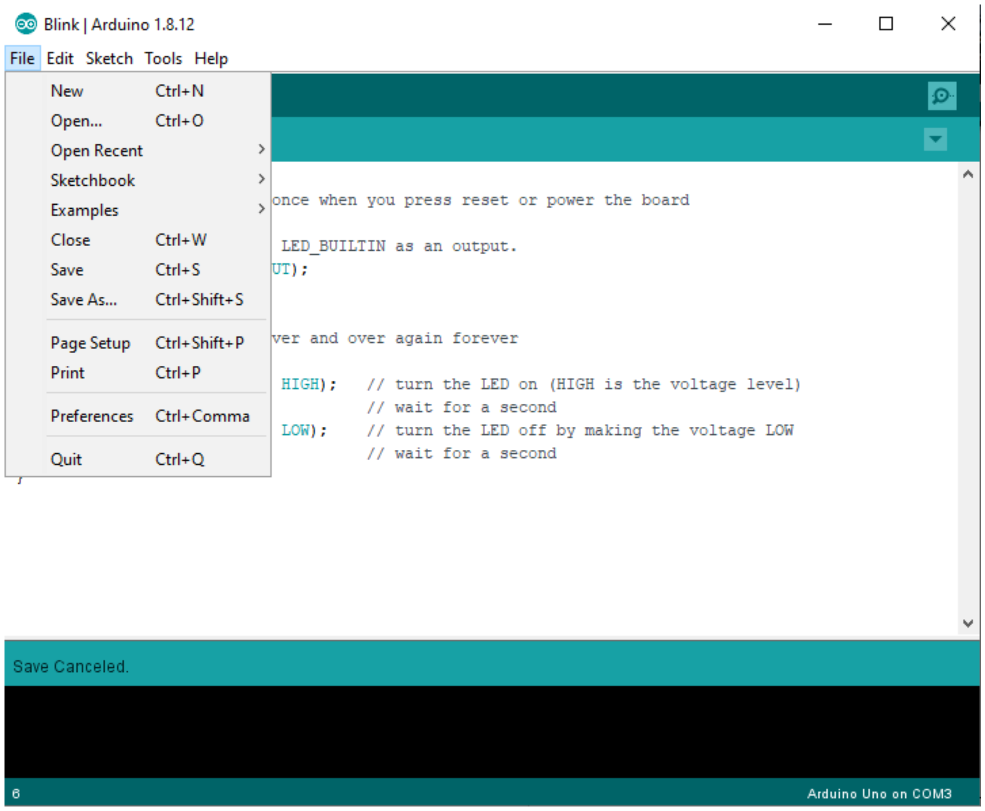
The window bar consists the name of File and the Arduino IDE software version

Menu Bar:

The menu bar consists of

- File
- Edit
- Sketch
- Tools
- Help

File:



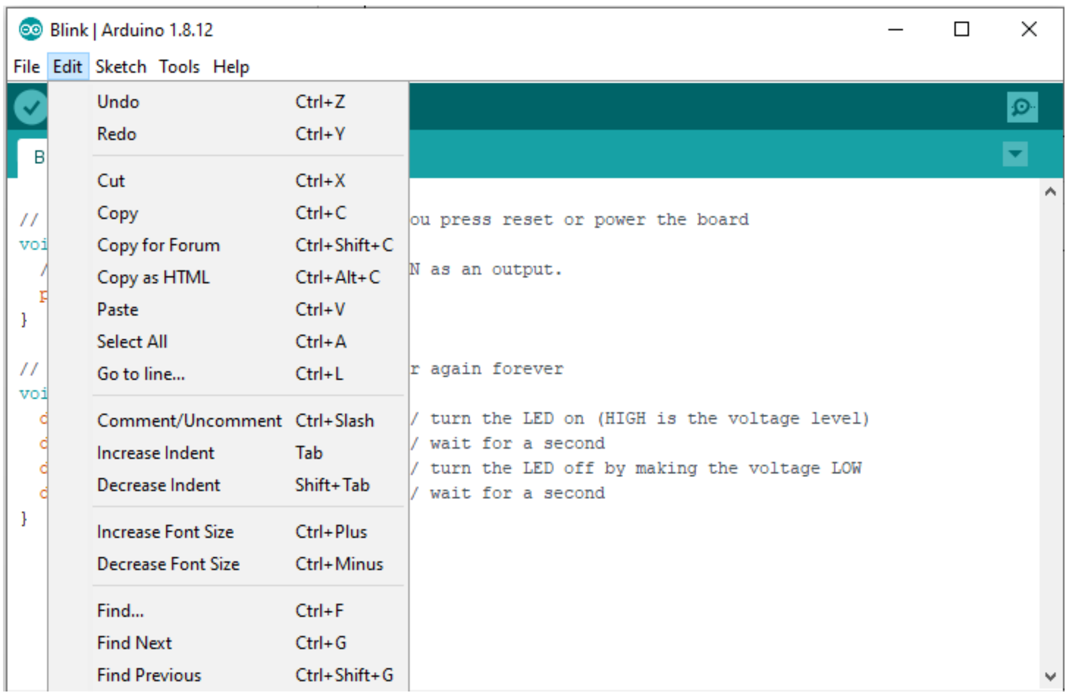
```
once when you press reset or power the board

 LED_BUILTIN as an output.
UT);


ver and over again forever

 HIGH);   // turn the LED on (HIGH is the voltage level)
          // wait for a second
 LOW);    // turn the LED off by making the voltage LOW
          // wait for a second
```

```
Save Canceled.
6                                                    Arduino Uno on COM3
```

- **New**
  It creates a new File. **(Ctrl+N)**
- **Open**
  It is used to open the file which was saved before. **(Ctrl+O)**
- **Open Recent**
  It shows the shortlist of Recently opened programs.
- **Sketchbook**
  Shows the current sketches which you have used for your project
- **Examples**
  Examples of a few basic problems for reference.
- **Close**
  Closes the main screen window. **(Ctrl+W)**
- **Save**
  It is used to save the current sketch. **(Ctrl+S)**
- **Save as…**
  Allows saving the current sketch with a different name. **(Ctrl+Shift+S)**
- **Page Setup**
  Page settings for modifying the page(Text). **(Ctrl+Shift+P)**
- **Print**
  Used to print the current program. **(Ctrl+P)**
- **Preferences**
  Settings of the IDE software can be changed here. **(Ctrl+,)**

- **Quit**
  Closes all IDE windows. **(Ctrl+Q)**

Edit:

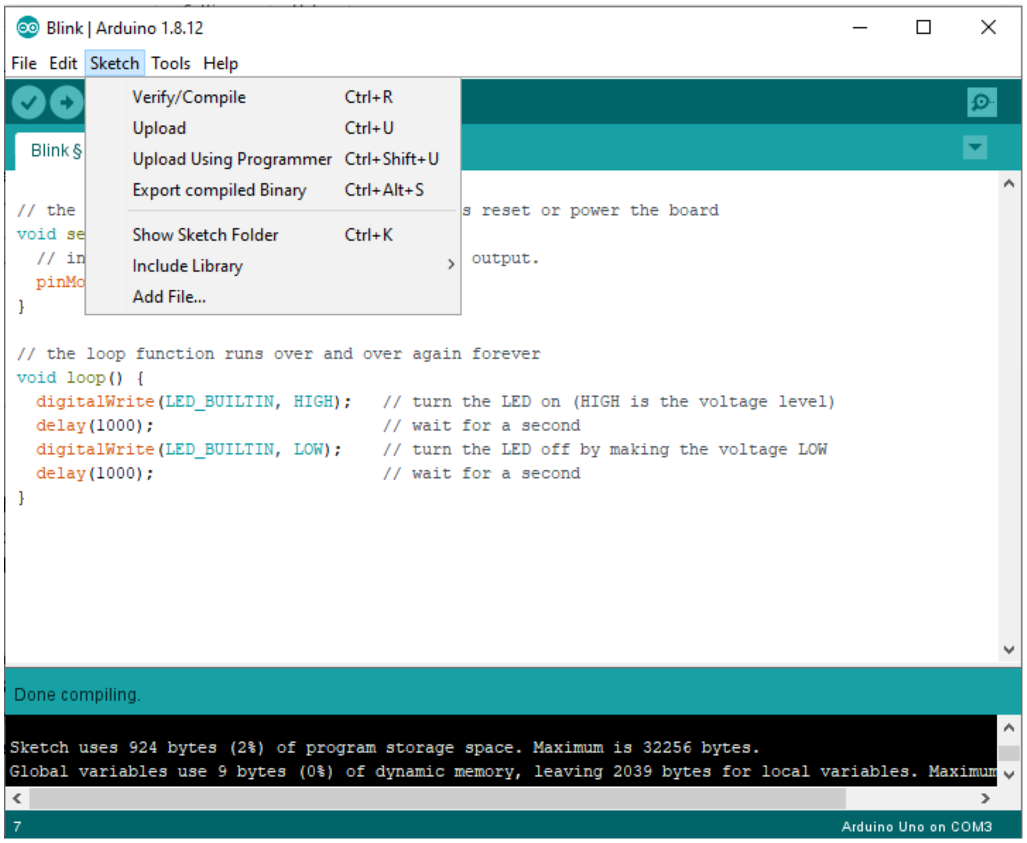

Blink | Arduino 1.8.12
File Edit Sketch Tools Help
Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Copy for Forum Ctrl+Shift+C
Copy as HTML Ctrl+Alt+C
Paste Ctrl+V
Select All Ctrl+A
Go to line... Ctrl+L
Comment/Uncomment Ctrl+Slash
Increase Indent Tab
Decrease Indent Shift+Tab
Increase Font Size Ctrl+Plus
Decrease Font Size Ctrl+Minus
Find... Ctrl+F
Find Next Ctrl+G
Find Previous Ctrl+Shift+G

```
ou press reset or power the board

N as an output.



r again forever

/ turn the LED on (HIGH is the voltage level)
/ wait for a second
/ turn the LED off by making the voltage LOW
/ wait for a second
```

- **Undo/Redo**
  Goes back of one or more steps you did while editing.
- **Cut**
  Cuts the selected text from the editor.
- **Copy**
  Copies the selected text from the editor
- **Copy for Forum**
  It copies and changes the style of code suitable for the forum.
- **Copy as HTML**
  It copies and changes the style of code suitable for the Html.
- **Paste**
- It pastes the text from the copied text.
- **Select All**
  Select's all the content from the editor.
- **Comment/Uncomment**
  It is used to comment and uncomment selected lines of code.
- **Increase/Decrease Indent**
  Adds or removes a space at the beginning of each selected line
- **Find**
  Finds the typed text in the editor
- **Find Next**
  Finds the next position of the searching word.

- **Find Previous**
  Finds the previous position of the searching word.

Sketch:

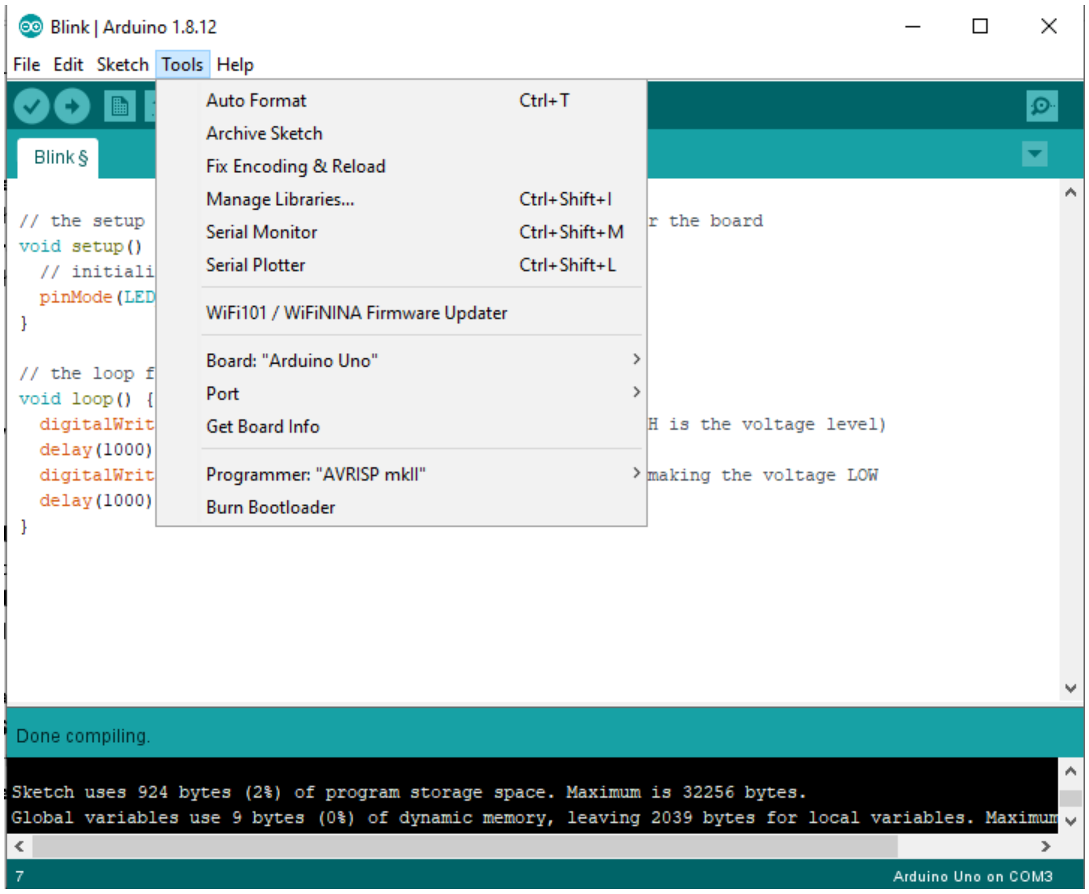


```
// the                                     s reset or power the board
void se
  // in                                      output.
  pinMo
}

// 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
}
```

```
Sketch uses 924 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
```

- **Verify/Compile**
  Checks or verifies your program if any error is there, and displays in the output panel.
- **Upload**
  It compiles and also uploads the code to the Arduino board.
- **Upload Using Programmer**
  Uploads code using Programmer which is available in Tools Tab.
- **Export Compiled Binary**
  Saves a .hex file in the System
- **Show Sketch Folder**
  Opens the current sketch folder.
- **Include Library**
  Adds a library to your sketch by inserting #include statements at the start of your code
- **Add File…**
  Adds a file to the sketch and the new file appears in a new tab in the window.

Tools:

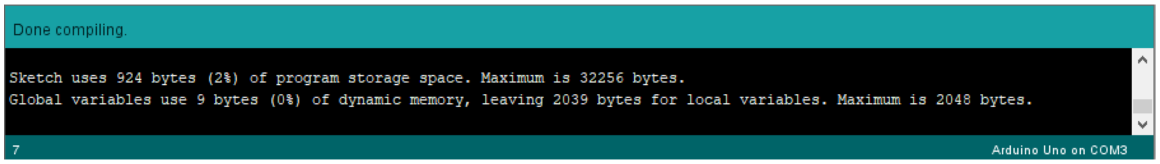

Blink | Arduino 1.8.12
File Edit Sketch Tools Help
Blink §
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Done compiling.
Arduino Uno on COM3

```
// the setup                                          r the board
void setup()
  // initiali
  pinMode(LED
}

// the loop f
void loop() {
  digitalWrit                                          H is the voltage level)
  delay(1000)
  digitalWrit                                          making the voltage LOW
  delay(1000)
}
```

```
Sketch uses 924 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
```

- **Auto Format**
  This option formats your code to a nice format so everyone can understand.
- **Archive Sketch**
  Copies the code into winrar format(.zip)
- **Fix Encoding & Reload**
  Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.
- **Serial Monitor**
  Serial monitor shows the visual communication by sending and receiving data
- **Board**
  To select the type of Arduino Board
- **Port**
  To select the port where you have connected the Arduino
- **Programmer**
  For selecting a hardware programmer when programming a board or chip and not using the USB type of communication.
- **Burn Bootloader**
- It is used to burn bootloader to the Arduino board

Output Panel:



```
Sketch uses 924 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.
```

This output panel is used to give comments about the code

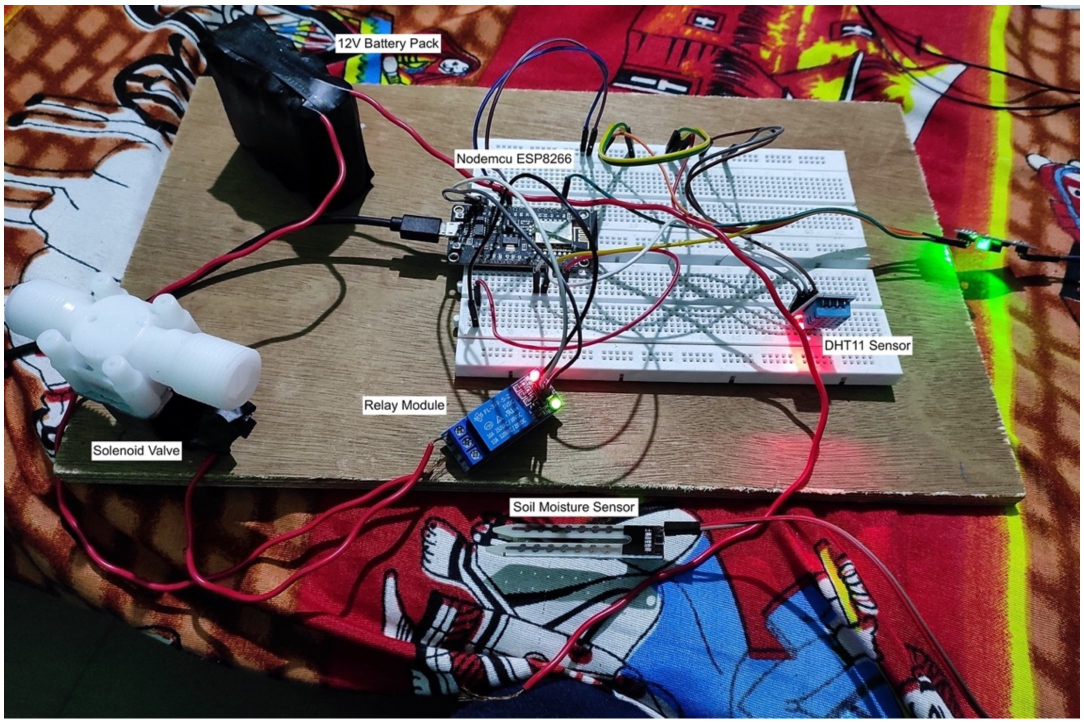
- if the code is successfully compiled or any error occurs.
- If the code has been successfully uploaded to the board.

Working

The automatic watering system is shown in picture down below. The automatic watering system consists of three main functions, namely the watering system, monitoring system, and notification system.

The watering system performs the watering function as shown in picture and explained as follows:

1. The soil moisture sensor is connected to the NodeMCU ESP 8266 microcontroller.
2. Soil moisture is implanted into the soil to detect water content in the soil.
3. Soil moisture sensor detect the water content from the soil and get analogue input signal to be processed in the microcontroller.
4. The NodeMCU microcontroller send the output signal to the relay.
5. The relay received output signal from the microcontroller and act as switch to the open or close the solenoid valve according to the input given to it.
6. The solenoid valve opened the valve when the relay is on and closed the valve if the relay is off. Water flows through the watering pipe when the solenoid valve opened.
7. Watering pipe is constructed on top of the plant so that the water splash to the plant and to the soil.



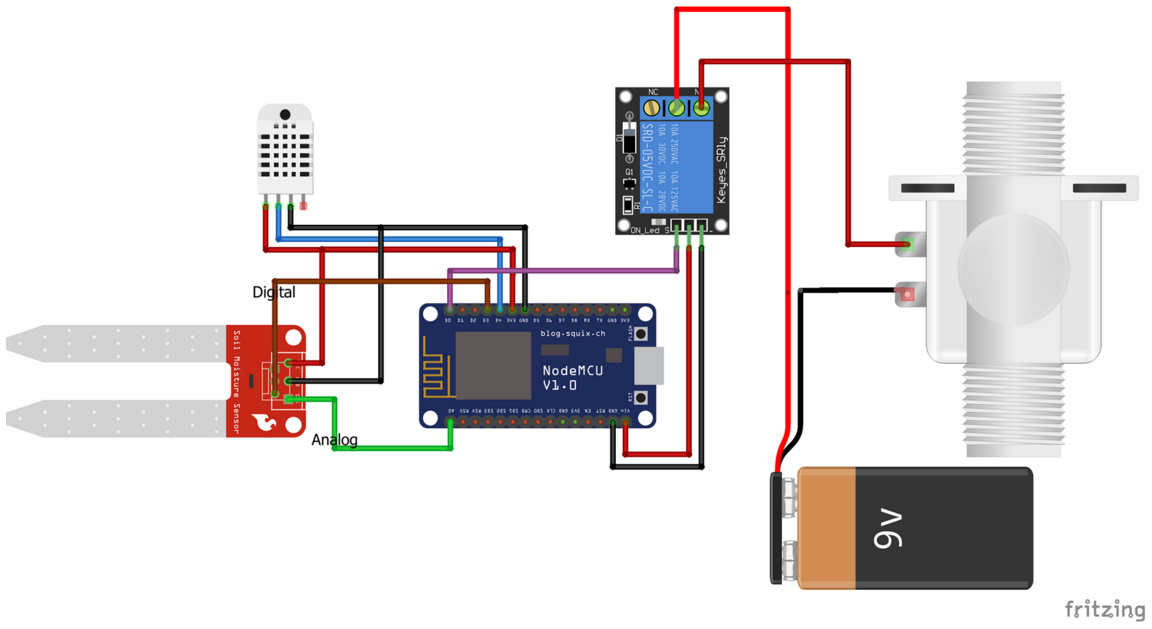
The second function is the monitoring system, which is monitoring the soil moisture using Blynk and explained as follows.

1. The signal for detecting the soil moisture was successfully received by the NodeMCU microcontroller from soil moisture sensor. NodeMCU microcontroller that has been equipped by wi-fi module and connected to the wi-fi access point, send cloud data to Blynk application.
2. Users can use the Blynk app from a smartphone to access the data.
3. After login, user can access data on the Blynk to view and monitor the soil moisture through a graphical form.

The third system is the notification system which is run when the watering device has started or finished. System send a notification to the user's smartphone. When the watering device is activated, user gets a notification on the smartphone. When the watering device has deactivated, the user gets a notification that the device is disabled. The processed is describe as follows:

1. Watering system that has been active or deactivated will then send data to Blynk App.
2. Blynk app get data from the NodeMCU microcontroller, processes the data, and send the notification to the user's smartphone.
3. Users can check the notification on Blynk App in the smartphone about the watering device whether it is activated or deactivated.

The schematic watering device is shown in figure down below. The prototype consists of a soil moisture sensor, relay module, jumper cables, solenoid valve, power cable and microcontroller NodeMCU which is connected to Blynk IoT. The figure shows an implementation of an automatic watering device. The watering pipe could be adjust according to the area of watering for plants. Figure made using Fritzing software.

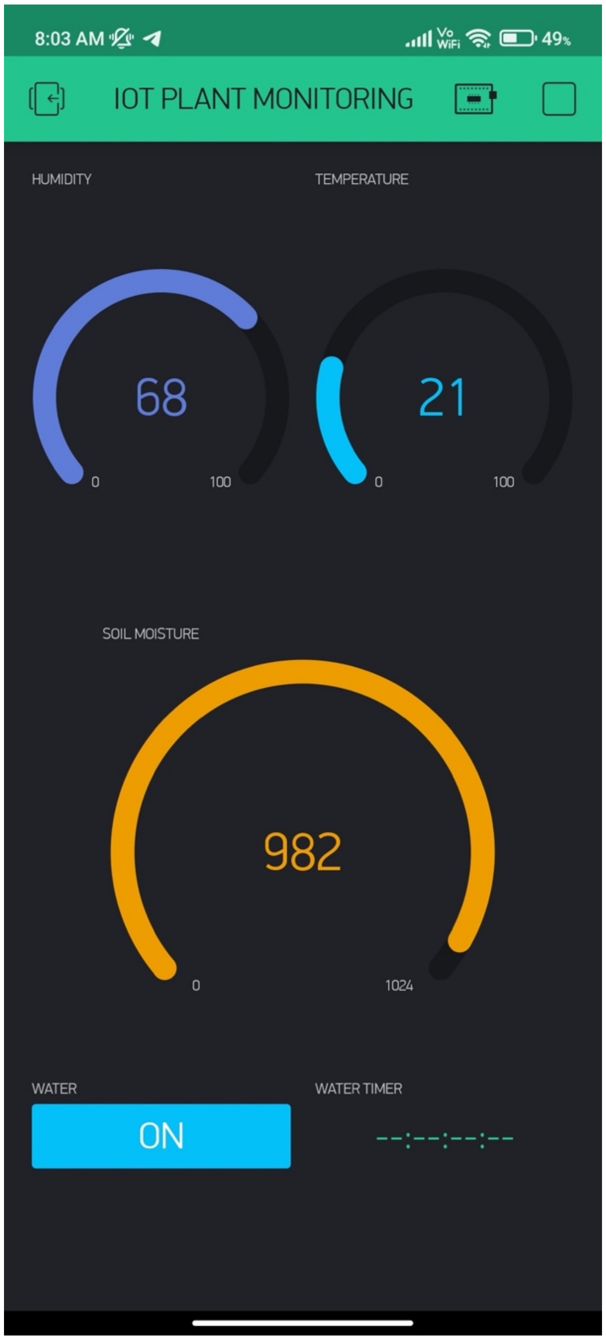


Code

```
#define BLYNK_PRINT Serial
#include <SPI.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <SimpleTimer.h>
#include <DHT.h>
#define BLYNK_PRINT Serial
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS D2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

char auth[] ="jwJkUprL5cdLeP5BWxVVhJTdn90PdJuF";              //Authentication code
sent by Blynk
char ssid[] = "panda";                                    //WiFi SSID
char pass[] = "jai117799";                                  //WiFi Password

#define sensorPin D3
int sensorState = 0;
int lastState = 0;
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
SimpleTimer timer;

void sendSensor()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Blynk.virtualWrite(V5, h);  //V5 is for Humidity
  Blynk.virtualWrite(V6, t);  //V6 is for Temperature
}

void setup()
{
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
  pinMode(sensorPin, INPUT);
  dht.begin();

  timer.setInterval(1000L, sendSensor);
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);
  sensors.begin();
}

int sensor=0;
void sendTemps()
{
  sensor=analogRead(A0);
  sensors.requestTemperatures();
  float temp = sensors.getTempCByIndex(0);
  Serial.println(temp);
  Serial.println(sensor);
  Blynk.virtualWrite(V1, temp);
  Blynk.virtualWrite(V2,sensor);
  delay(1000);
}

void loop()
{
  Blynk.run();
  timer.run();
  sendTemps();
  sensorState = digitalRead(sensorPin);
  Serial.println(sensorState);

  if (sensorState == 1 && lastState == 0) {
    Serial.println("needs water, send notification");
    Blynk.notify("Water your plants");
    lastState = 1;
    delay(1000);
    //send notification
  }
  else if (sensorState == 1 && lastState == 1) {
    //do nothing, has not been watered yet
  Serial.println("has not been watered yet");
  delay(1000);
  }
  else {
    //st
    Serial.println("does not need water");
    lastState = 0;
    delay(1000);
  }
  delay(100);
}
```

App Demo

Picture down below display the Blynk application for monitoring and notification. The Blynk application display two types of notifications sent to smartphone users. The first notification is the message when the device is turn ON and perform the watering function. When the tool has finished performing the watering function, the device is turn OFF and system send the second notification to user smartphone through Blynk apps.



The yellow gauge displays the soil moisture, dark blue humidity and light blue temperature. The solenoid valve can be switched on whenever the a person feels like using a button provided in the app. User can also set a timer for valve to get automatically switch on and switch off.

Conclusion

Testing was performed to find out whether the system working properly or not. Testing was conducted using the black box testing method. The test is carried out in the three parts. First part was device testing to test the functionalities of each components. Second part testing was related with the monitoring of soil moisture through the Thingspeak. Third part testing was about the notification through Blynk apps regarding the information about the device when it's activated or deactivated.

The testing results are shown on the Table. 1, Table. 2, Table. 3 and Table. 4.

1. Testing the functionalities of the watering device.
Table 1 show the testing results of the watering device. The watering device consist of the soil moisture sensor, Wemos D1 microcontroller, relay, solenoid valve and watering pipe. The parameter for this testing is based on the detection results from the soil moisture sensor. In this research, the reference of the soil moisture is preferred and set in the value between 30% - 35 %. When the sensor detected the soil moisture level is between 30% to 35% then the soil is in a dry level. When the soil is dry, it means the plants need water, so the relay state ON and triggered the solenoid valve to open the watering pipe to drain the water to the plants.

Table 1. Functionalities testing of the watering device.

No.	**Soil moisture detection value**	**Wemos D1**	**Relay**	**Solenoid Valve**	**Watering Pipe**	**Test Results**
1.	30 % - 35 %	ON	ON	Open	Enable drain water	Succeeded
2.	More than 35%	ON	OFF	Closed	Disable drain water	Succeeded

2. Testing the soil moisture monitoring through Blynk
The scenario of the testing was conducted to get the results value of the soil moisture detected by the sensor. In these experiments we use five different 250 ml cup. The soil was put in each cup and then 150 ml water poured out to the soil. The experiments were carried out in five experiments. The special treatment was applied to test the state of the soil moisture.

The first special treatment is related with the initial value of soil moisture which set to the value of 0%. The results of soil moisture testing is shown in the Table 2.

Table 2. Testing the soil moisture with initial soil moisture 0%

Experiments	*Initial value of soil moisture (%)*	*Final value of soil moisture (%)*	*Increment moisture value (%)*
1	0 %	50%	50%
2	0 %	66%	66%
3	0 %	63%	63%
4	0 %	65%	65%
5	0 %	58 %	58 %
Average final value of soil moisture		***60,4%***	**-**
Average increment value of soil moisture		**-**	***60,4%***

From Table. 2, it shown that the final soil moisture are varies between 50% to 66 %. The average value

of the soil moisture can be calculated with a simple formula as shown below.

$$S = \frac{F}{T} x\ 100\% \text{ ....} (1)$$

Average value of soil moisture (*S*) is equal then the total sum of final value of soil moisture *(F)* divide by number of testing (T) then multiplied by 100%. The Formula (1), could be applied to calculate the average final value of soil moisture or to calculate the average of the increment value of soil moisture.

The average final value is 60,4% which is the same with the average increment value of the soil moisture.

The second special treatment is concerned with the range of soil moisture reference value. For each experiment the initial value of the soil moisture was varies between 30% - 35% as shown in the Table. 3. From five experiments, the final moisture value ranges from 65% - 72%. The average final moisture value had been calculated with formula (1), and the results shown 68,2%. From each experiment, the increment moisture value could be calculated simply by subtraction of the final value with the initial value of soil moisture. The average increment value of soil moisture is 35,8%.

Table 3. Testing the soil moisture with reference soil moisture 30% – 35%

Experiments	*Soil moisture reference value (%)*	*Initial value of soil moisture (%)*	*Final value of soil moisture (%)*	*Increment moisture value (%)*
1	30% - 35%	32%	68%	36%
2	30% - 35%	30%	65%	35%
3	30% - 35%	35%	72%	37%
4	30% - 35%	33%	69%	36%
5	30% - 35%	32%	67 %	35%
Average final value of soil moisture			***68,2%***	-
Average increment value of soil moisture			-	***35,8%***

3. Testing the notification through Blynk apps

Table 2. Blynk apps functionalities

Soil moisture detection value	***Watering device***	***Notification status in Blynk apps***
30 % - 35 %	Activated	Succeeded send the notification to user smartphone
More than 35%	Deactivated	Succeeded send the notification to user smartphone

The automatic watering system integrated with IoT platforms Blynk could perform the functions of watering the plant according to the purpose of the research. Soil moisture sensor detected the water moisture in the soil and send signal to NodeMCU microcontroller. The reading results from sensor, processed by the microcontroller to generate the watering function automatically. The system sends notification to Blynk apps, when the device activated or deactivated the watering function. The system has the monitoring feature to record the soil moisture value through sensors which display the data through graph. The system had to connected to the internet to performs the real- time monitoring and notification. When the system disconnected from the internet, monitoring and notification functions could not be proceed, but the watering device will still perform watering function.

The initial value of the soil moisture could be adjust according to the moisture needed by the plants. This can be done by reprogramming the microcontroller. For this research, we set the initial value of the soil moisture in range of 30% - 35%. The testing results of the Blynk soil moisture monitoring based on the experimental scenario, calculated the average final value of soil moisture is 68,2%. This means that, in one cycle of watering, the device could perform the process of moisturizing the soil with average increment calculated 35,8%.

For further research development, the system could be added with more soil moisture sensors or another sensor such as temperature and humidity sensor and then conduct different experimental scenarios to gain the comparison. The watering pipe could be customized according to the area of the plants on the ground. This automatic watering system has the potential to used simply for gardening or implemented the field of agriculture. Moreover we can use an android app to add more features which are currently reserved for Blynk premium users.

References

https://www.viralsciencecreativity.com/post/iot-smart-plant-monitoring-system

https://blynk.io/

https://www.nodemcu.com/index_en.html

https://github.com/abhiraj121/IOT-Plant-Watering-System