SMART GARDENING SYSTEM USING IOT

A MINI PROJECT REPORT

Submitted by

SUDHARSANAM.K	**-**	**1931048**
VIGNESH BALAJI.K	**-**	**1931053**
MADHAN.S	**-**	**2031L02**

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

GOVERNMENT COLLEGE OF ENGINEERING, SALEM -11

(An Autonomous Institution Affiliated to Anna University, Chennai)

JUNE 2022

BONAFIDE CERTIFICATE

Certified that this mini project report **"SMART GARDENING SYSTEM USING IOT"** is the mini project work of

SUDHARSANAM.K	**-**	**1931048**
VIGNESH BALAJI.K	**-**	**1931053**
MADHAN.S	**-**	**2031L02**

who carried out the mini project work under my supervision

SIGNATURE	**SIGNATURE**
SUPERVISOR:	**HEAD OF THE DEPARTMENT:**
M.Priadarsini	Dr.A.M.Kalpana
Assistant Professor	Head of the Department(IC)
Department of Electronics and	Department of Electronics and
Communication Engineering,	Communication Engineering,
Government College of	Government College of
Engineering, Salem-636011.	Engineering, Salem-636011.

Submitted for University Mini Project Viva-Voce held on

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

At this delightful moment of having successfully completed our mini-project, we convey our sincere thanks and gratitude to our beloved **Principal Dr.R.Malayalamurthi** for providing us the great opportunity with all facilities required for the successful completion of our mini-project work.

We wish to express our gratitude and profound thanks to **Dr.A.M.Kalpana Head of the Department**, Department of Electronics and Communication Engineering, Government College of Engineering, Salem, for forwarding us to do our mini project and offering adequate duration and constant encouragement in completing the mini project work.

We wish to record my deep sense of gratitude and profound thanks to our Project Supervisor **M.Priadarsini Assistant professor**, Department of Electronics and Communication Engineering, Government College of Engineering, Salem, who has induced us to innovate this mini-project with technological intents, inspiring guidance, keen interest and constant encouragement throughout project work, to bring this report into fruition.

We would like to thank the Mini Project Review Committee Members for their comments which help to improve the overall quality of the mini-project work. We also convey our sincere thanks to all the teaching faculty and non-teaching staff members of our Department, for their kind help and valuable support which facilitate us to complete the mini-project work successfully. Finally, we wish to express our heartfelt thanks to our beloved parents and all our friends for the kindly help rendered to them.

ABSTRACT

Watering various herbs,shrubs,crops and indoor plants at the right time with the right amount of time and the right amount of water is essential for the proper growth of the plant. Learning how to grow plants is complicated and costly. Plants are resilient,but just one innocent oversight can ruin your crop. Too much water can cause soil erosion,deplete nutrients such as nitrogen present in pores of soils and can cause plant root decay. At the same time,a lack of water can damage plant growth,resulting in their death. Therefore,an appropriate amount of water is needed for each plant. Also our busy schedule makes it difficult to properly monitor and water the plants. So,we created an iot based gardening system which can update useful information of the garden,such as temperature,humidity and the moisture content in the soil from various sensors into a cloud database using NodeMCU. Once the information is in the cloud,it can be accessed from anywhere using a smartphone app. This provides Global moitoring and operating capabilities. The app can also be used to relay commands to the system for timely watering.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

I want to start gardening, but I knew I wouldn’t keep up the regular schedule of watering the plants and making sure that they remain healthy. So, I recruited a micro-controller and suite of sensors to help with these tasks. Watering is the most important cultural practice and most labour intensive task in daily gardening operation. Watering systems ease the burden of getting water to plants when they need it. Knowing when and how much to water is two important aspects of watering process. To make the gardener works easy, the automatic plant watering system is created.The sensors detects the values and provide it to the nodemcu microcontroller. It converts these values into its appropriate form that is executing in it and gives the output in the form water flow according to the input values.

CHAPTER 2

LITERATURE SURVEY

https://www.ijitee.org/wp-content/uploads/papers/v8i5s/ES3457018319.pdf

In this system we use NodeMCU, soil moisture sensors and motor pump. The motor will get switched ON/OFF depending upon the soil moisture condition and the condition of motor can be displayed on OLED. The values of the various sensor are also relayed to a cloud which can be accessed from anywhere in the world.

CHAPTER 3

THE PROPOSED SYSTEM

3.1 Block diagram

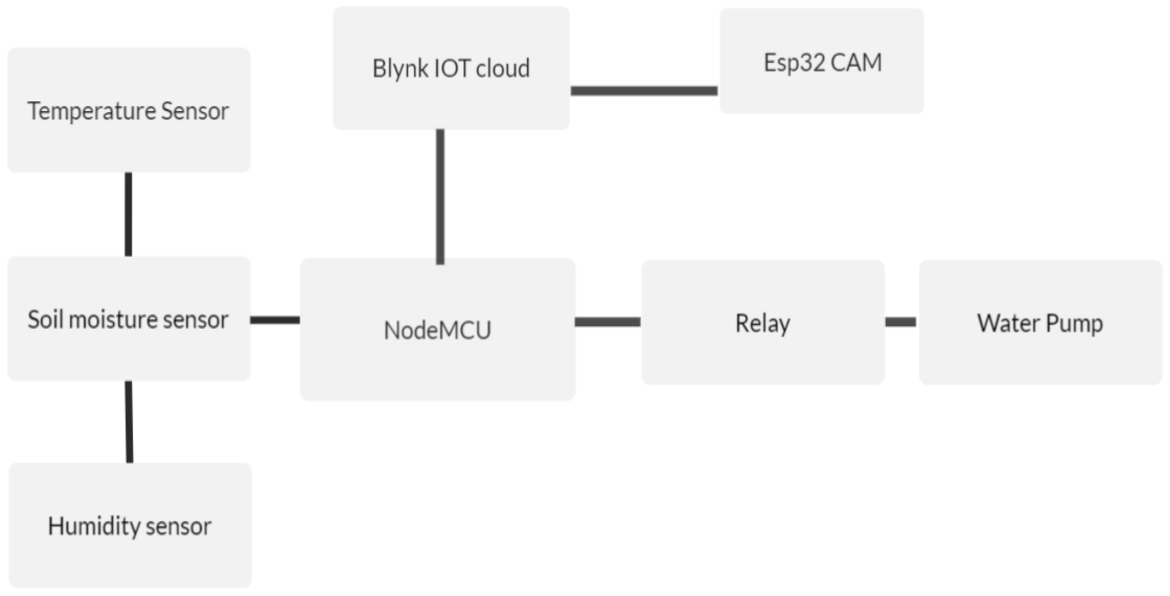


Figure 3.1-Block Diagram

NodeMCU is an open source platform based on ESP8266 which can connect objects and let data transfer using the Wi-Fi protocol. In addition, by providing some of the most important features of microcontrollers such as GPIO, PWM, ADC, and etc, it can solve many of the project's needs alone.

3.2 Circuit diagram

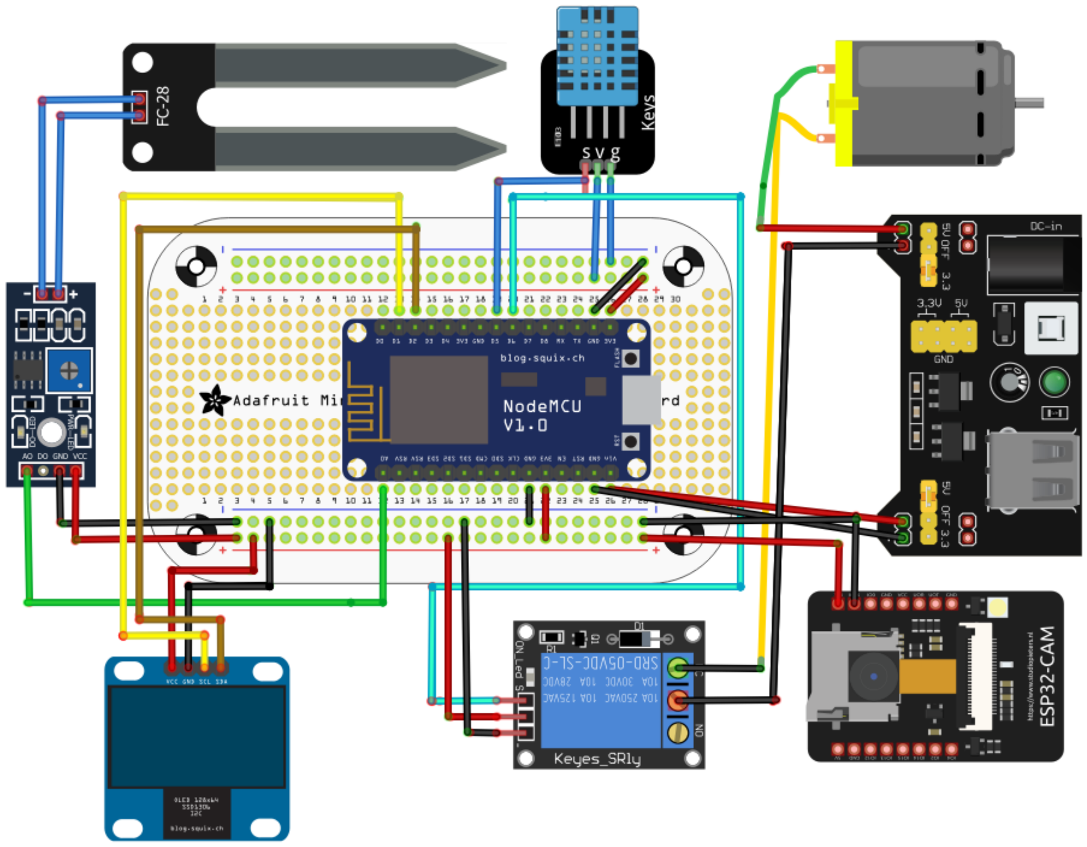


Figure 3.2-Circuit Diagram

Let us see the schematic of the IoT Smart Gardening & Automatic Watering System project. I use Fritzing to make an schematic for most of my projects. All you need is to place and connect a component that is super easy.

Connect the soil moisture sensor to A0 of Nodemcu and DHT11 to D4 Pin. The motor connects to Relay. To control the relay, we use the D5 Pin of NodeMCU. Connect the OLED display to the I2C pin of NodeMCU. You can power the Motor and Relay using the 5V pin of NodeMCU. The DHT11 Sensor, Capacitive Soil Moisture Sensor, and OLED Display require a 3.3V Supply only.

3.3 COMPONENTS USED

3.3.1 NodeMCU

ESP-12E Module

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

ESP-12E Chip

- Tensilica Xtensa® 32-bit LX106
- 80 to 160 MHz Clock Freq.
- 128kB internal RAM
- 4MB external flash
- 802.11b/g/n Wi-Fi transceiver

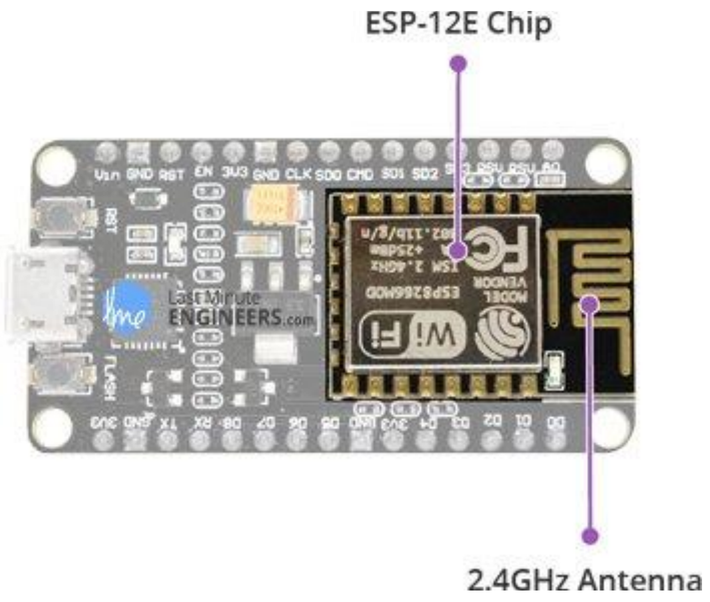


Figure 3.3.1(a)-NodeMCU

ESP8266 Development Platforms:

Platform IO :

According to the PlatformIO documentation "PlatformIO is a cross-platform, cross-architecture, multiple framework, professional tool for embedded systems engineers and for software developers who write applications for embedded products."

Put another way, this is a development tool that can run on most operating systems and under many different code editor packages. A few of the editors it runs under are:

- Visual Studio Code (VS Code)
- Atom
- Codeblocks
- Eclipse
- Netbeans
- Sublime Text

It can also be run on cloud-based packages like Codeanywhere and Eclipse Che.

We will be running it under Microsoft Visual Studio Code, a free development platform available for Linux, Windows and Mac OS X.

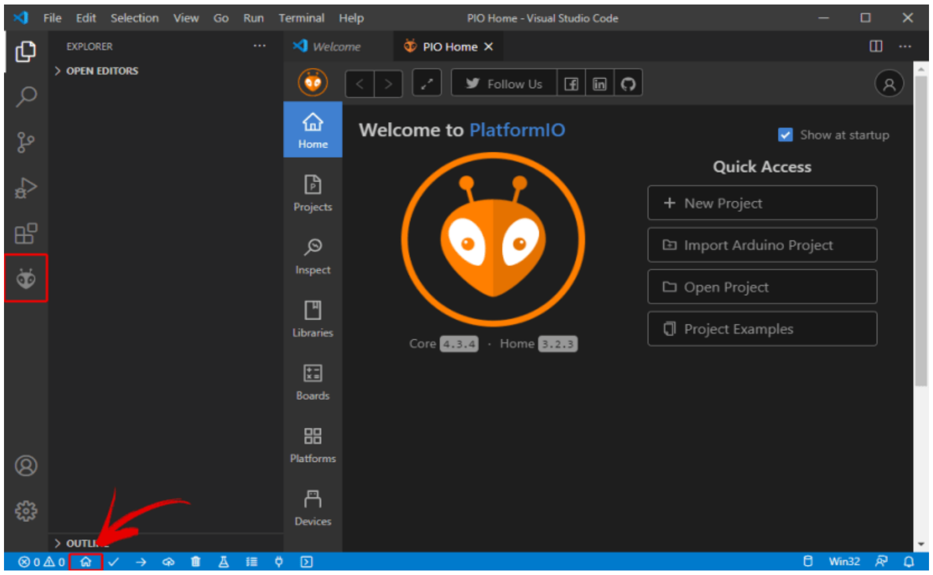


Figure 3.3.1(b) - Platformio Homepage

PlatformIO Basics :

When you first start Visual Studio Code with the PlatformIO extension you'll be greeted by the PlatformIO Home screen. If you don't see it look for a small icon on the bottom taskbar shaped like a "house" and click on it.

The Home screen displays the version of PlatformIO and also has a Quick Access section, which we will use to start our first project.

Before we get to that let's examine some of the other icons down the side of the PlatformIO screen.

- Home – You have already seen this, it has the current version and the Quick Access box that allows you to create new projects.
- Projects – A list of all the projects you have created. You can edit these to add descriptions.
- Inspect – This allows you to inspect a project for statistics like memory utilization.
- Libraries – This is the Library Manager, which we will describe in detail later on in this article.
- Boards – A list of the boards supported by PlatformIO. As of this writing, there are over 900.
- Platforms – Platforms like the ArduinoAVR, Espressif ESP32, and others are listed here. The list will grow as you build projects with new boards.

- Devices – A list of the boards that are currently attached to your computer. This is built up automatically so you don't need to select the port, unlike the Arduino IDE.

Create a New Project :

On VS Code, click on the PlartfomIO **Home** icon. Click on **+ New Project** to start a new project

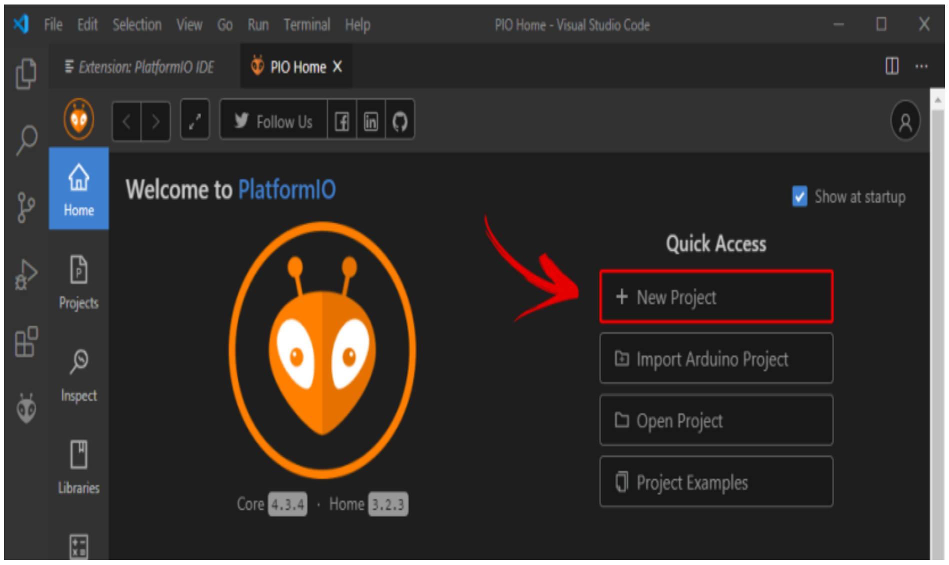
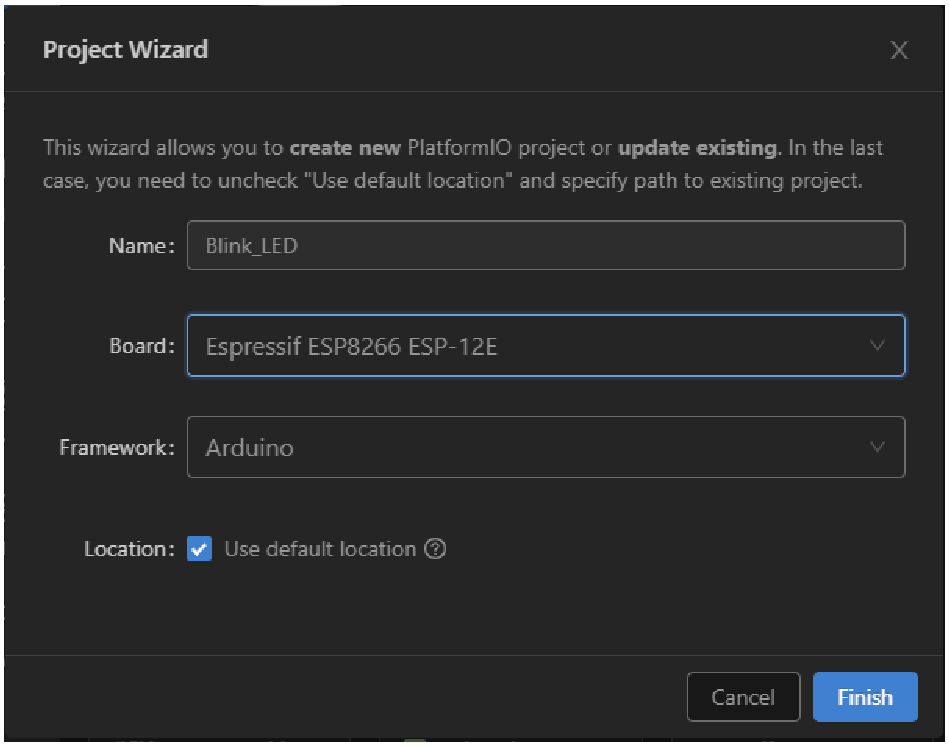
.

Figure 3.3.1(c) – Create new project

Give your project a name (for example Blink_LED) and select the board you're using. In our case, we're using the ESP8266 NodeMCU board . The Framework should be "Arduino" to use the Arduino core . You can choose the default location to save your project or a custom location.

The default location is in this path Documents >PlatformIO >Projects. For this test, you can use the default location. Finally, click "Finish".



Project Wizard

This wizard allows you to **create new** PlatformIO project or **update existing**. In the last case, you need to uncheck "Use default location" and specify path to existing project.

Name: Blink_LED

Board: Espressif ESP8266 ESP-12E

Framework: Arduino

Location: ✓ Use default location

Figure 3.3.1(d) – Selecting the board

The Blink_LED project should be accessible from the Explorer tab.

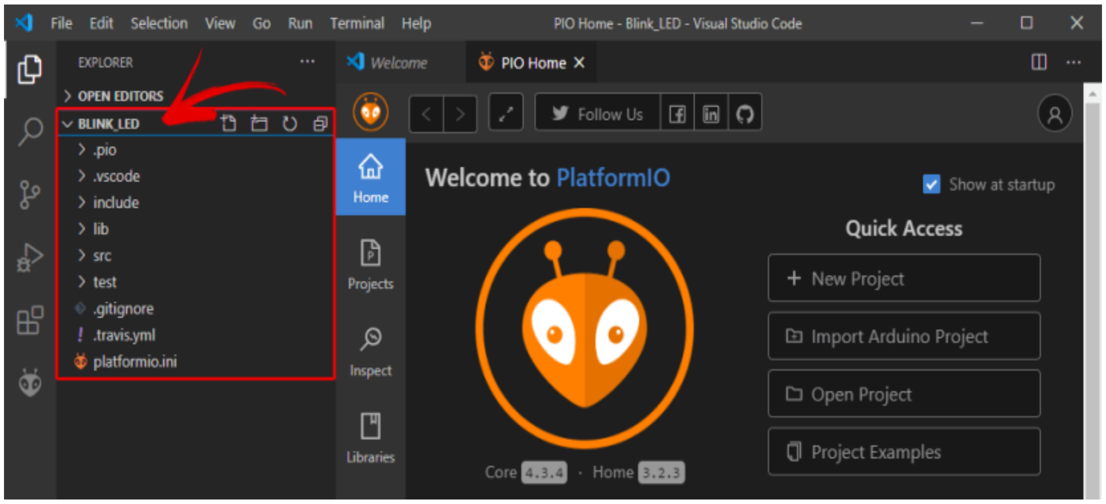


Figure 3.3.1(e) – Accessing the project

VS Code and PlatformIO have a folder structure that is different from the standard *.ino* project. If you click on the Explorer tab, you'll see all the files it

3.3.2 Esp32 CAM

Introduction :

The ESP32-CAM is a full-featured microcontroller that also has an integrated video camera and microSD card socket. It's inexpensive and easy to use, and is perfect for IoT devices requiring a camera with advanced functions like image tracking and recognition.

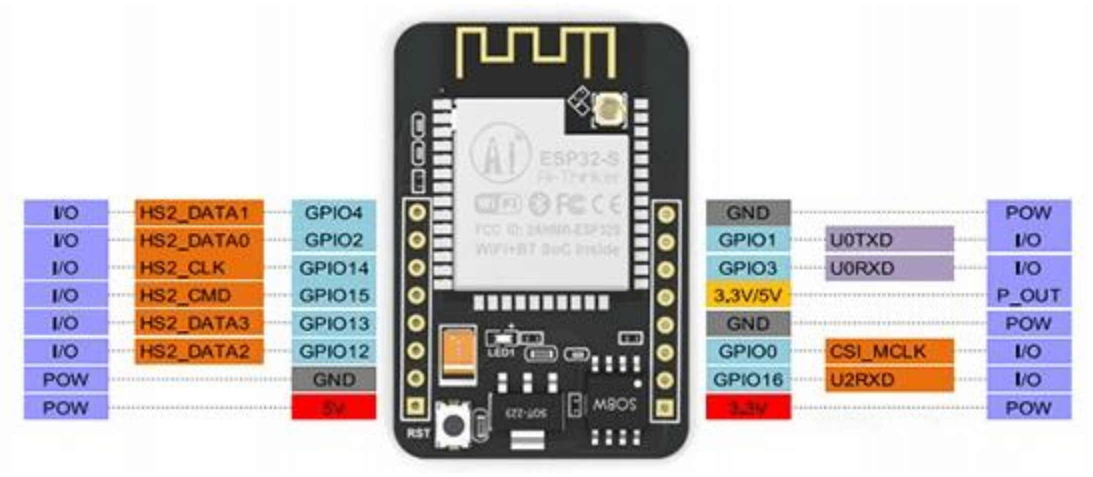


Figure 3.3.2(a) – Esp32 Cam Pinout

The sample software distributed by Espressif includes a sketch that allows you to build a web-based camera with a sophisticated control panel. After you get the hang of programming the device you'll find that it is very easy to use.

ESP32-CAM Specifications :

The ESP32-CAM is based upon the ESP32-S module, so it shares the same specifications. It has the following features:

- 802.11b/g/n Wi-Fi
- Bluetooth 4.2 with BLE
- UART, SPI, I2C and PWM interfaces

- Clock speed up to 160 MHz
- Computing power up to 600 DMIPS
- 520 KB SRAM plus 4 MB PSRAM
- Supports WiFi Image Upload
- Multiple Sleep modes
- Firmware Over the Air (FOTA) upgrades possible
- 9 GPIO ports
- Built-in Flash LED

Working with the ESP32-Cam :

Using the ESP32-CAM is similar to using the ESP32 modules we looked at previously, with one major difference. The ESP32-CAM board has no USB port, so you can't just connect it up to your computer and start loading programs.

Instead you will need to add an external FTDI adapter. This is the same adapter you would use programming an Arduino Pro Mini, so if you've worked with the Pro Mini then you probably already have one of these.

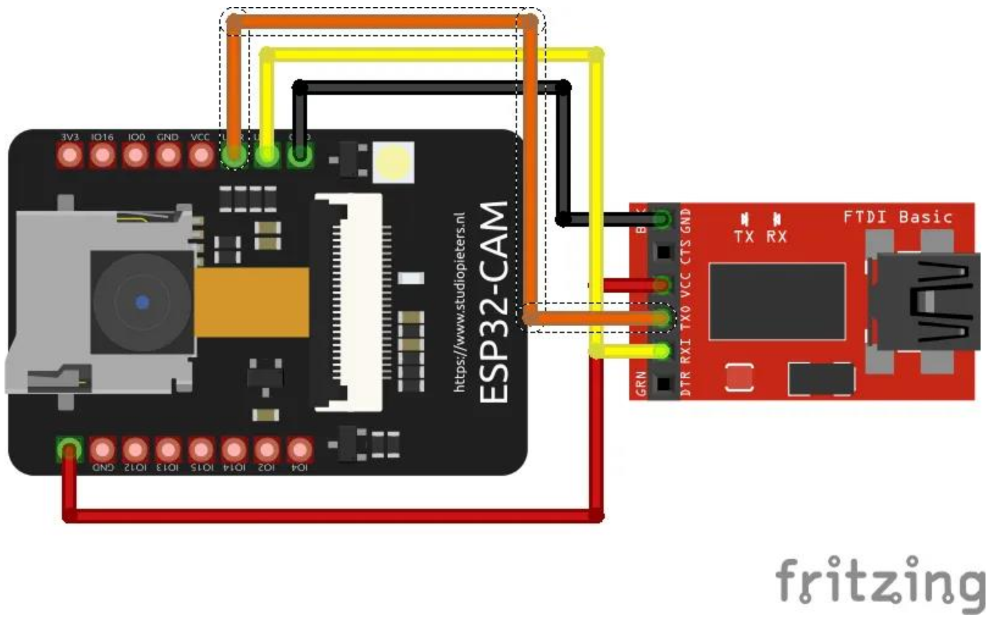


Figure 3.3.2(b) – Interfacing Esp32 Cam with FTDI adapter

3.3.3 Soil moisture sensor

Hardware Overview :

A typical soil moisture sensor has two components.

The Probe -

- The sensor contains a fork-shaped probe with two exposed conductors that goes into the soil or anywhere else where the water content is to be measured.
- Like said before, it acts as a variable resistor whose resistance varies according to the soil moisture.

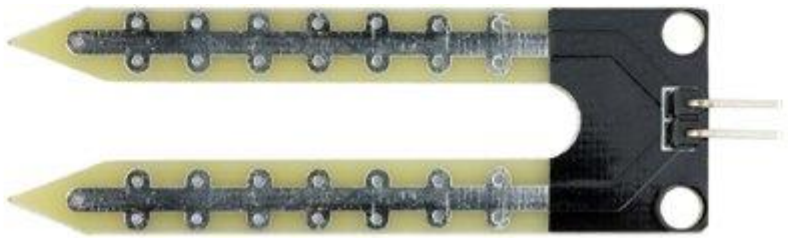


Figure 3.3.3(a) - Probe

The Module -

- The sensor also contains an electronic module that connects the probe to the Arduino.
- The module produces an output voltage according to the resistance of the probe and is made available at an Analog Output (AO) pin.
- The same signal is fed to a LM393 High Precision Comparator to digitize it and is made available at an Digital Output (DO) pin.

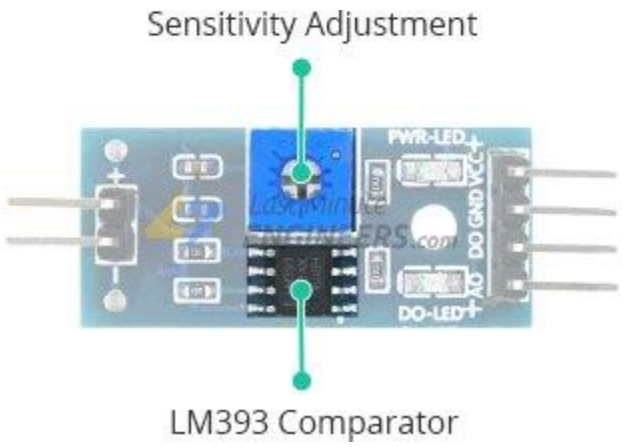


Figure 3.3.3(b) - Module

How Soil Moisture Sensor works?

The working of the soil moisture sensor is pretty straightforward. The fork-shaped probe with two exposed conductors, acts as a variable resistor (just like a potentiometer) whose resistance varies according to the water content in the soil.

This resistance is inversely proportional to the soil moisture.

- The more water in the soil means better conductivity and will result in a lower resistance.
- The less water in the soil means poor conductivity and will result in a higher resistance.

The sensor produces an output voltage according to the resistance, which by measuring we can determine the moisture level.

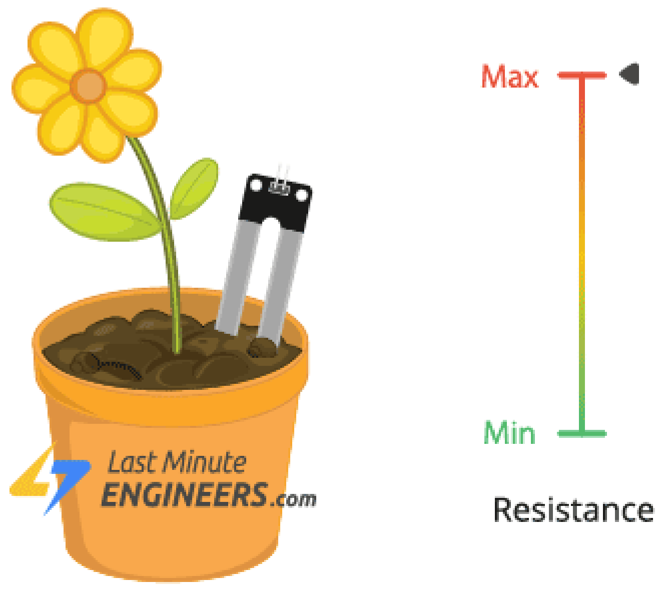


Figure 3.3.3(c) – Working of Soil mosture sensor

3.3.4 One channel relay module

We are going to use one channel relay module. However there are other modules with two, four and eight channels. You can choose the one that best suits your needs.

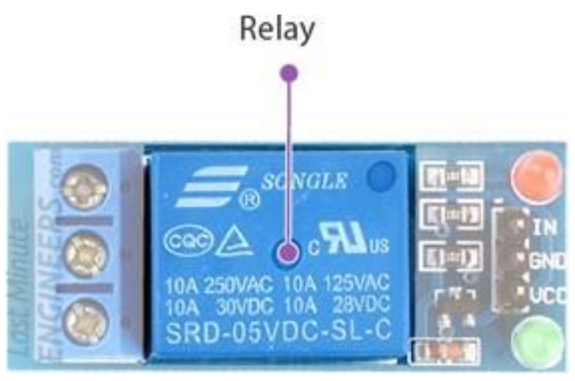


Figure 3.3.4(a) – Relay module

This module is designed for switching only a single high powered device from your Arduino. It has a relay rated up to 10A per channel at 250VAC or 30VDC.

One Channel Relay Module Pinout :

Let’s have a look at the pinout of one channel relay module.

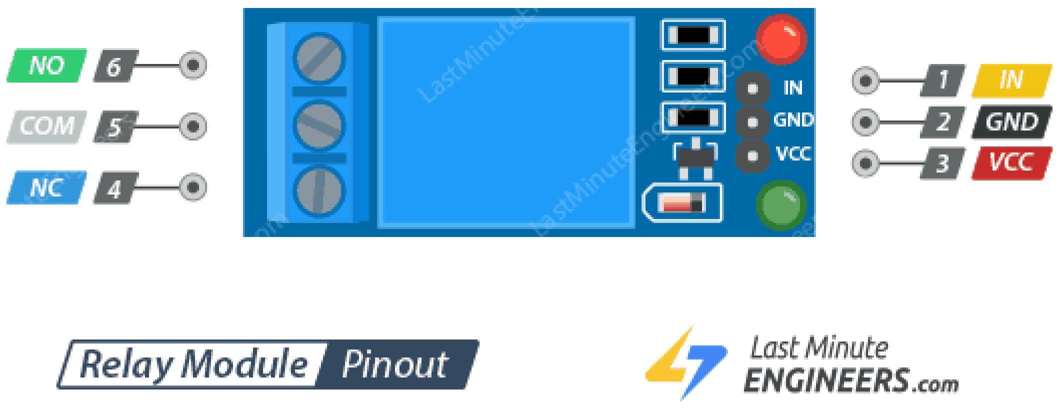


Figure 3.3.4(b) – Relay module pinout

3.3.5 DHT11 module

Give your next Arduino project the ability to sense the world around it with the inexpensive DHT11 digital temperature & humidity sensor module from AOSONG. This sensor is pre-calibrated and don’t require extra components so you can start measuring relative humidity and temperature right away. The DHT11 module is fairly easy to connect. It has only three pins:

- +VCC pin supplies power for the sensor. 5V supply is recommended, although the supply voltage ranges from 3.3V to 5.5V. In case of 5V power supply, you can keep the sensor as long as 20 meters. However, with 3.3V supply voltage, cable length shall not be greater than 1 meter. Otherwise, the line voltage drop will lead to errors in measurement.
- Out pin is used to communication between the sensor and the Arduino.

- -GND should be connected to the ground of Arduino.

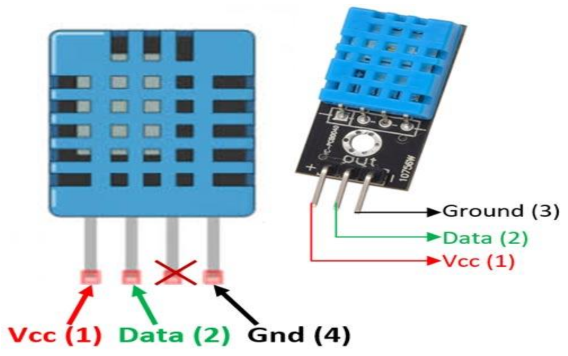


Figure 3.3.5 – DHT sensor module

3.3.6 OLED Display Module

The OLED display module breaks out a small monochrome OLED display. It's 128 pixels wide and 64 pixels tall, measuring 0.96" across. It's micro, but it still packs a punch – the OLED display is very readable due to the high contrast, and you can fit a deceivingly large amount of graphics on there.

As the display makes its own light, no backlight is required. This significantly reduces the power required to run the OLED and is why the display has such high contrast, extremely wide viewing angle and can display deep black levels.

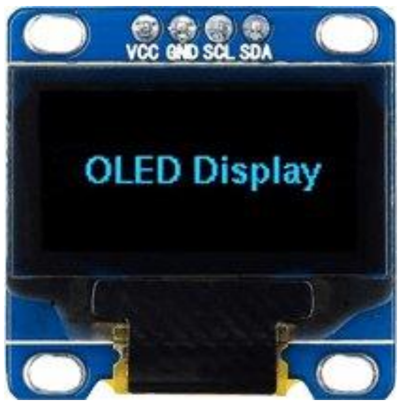


Figure 3.3.6(a) – OLED display module

At the heart of the module is a powerful single-chip CMOS OLED driver controller – SSD1306, which handles all the RAM buffering, so that very little work needs to be done by your ESP32. Also the operating voltage of the SSD1306 controller is from 1.65V to 3.3V – Perfect for interfacing with 3.3V microcontrollers like ESP32.

OLED Memory Map :

To have absolute control over your OLED display module, it's important to know about its memory map.

Regardless of the size of the OLED module, the SSD1306 driver has a built-in 1KB Graphic Display Data RAM (GDDRAM) for the screen which holds the bit pattern to be displayed. This 1K memory area is organized in 8 pages (from 0 to 7). Each page contains 128 columns/segments (block 0 to 127). And each column can store 8 bits of data (from 0 to 7). That surely tells us we have The whole 1K memory with pages, segments and data is highlighted below.

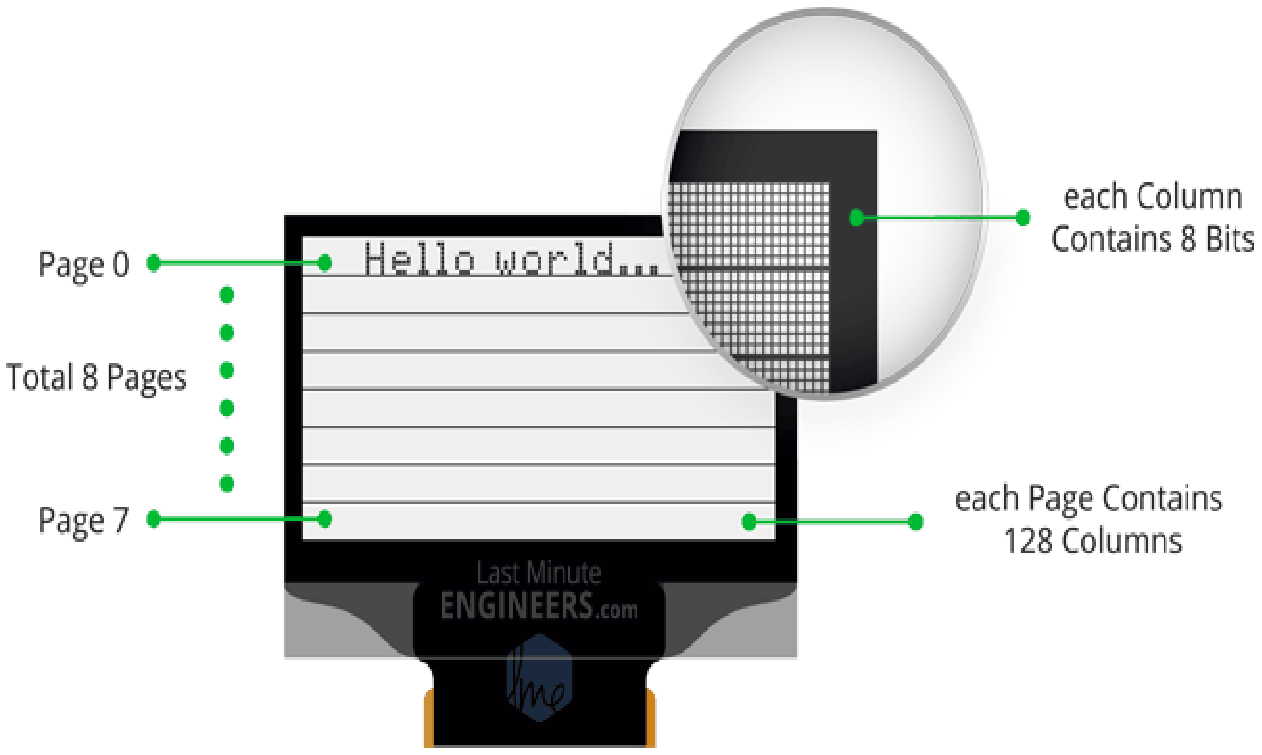

Page 0
Total 8 Pages
Page 7
Hello world...
each Column Contains 8 Bits
each Page Contains 128 Columns
Last Minute ENGINEERS.com

Figure 3.3.6(b) : Memory map

Each bit represents particular OLED pixel on the screen which can be turned ON or OFF programmatically.

3.3.7 Adapter 5v

This Orange 1A 9V Adapter Power supply Charger with a 5.5mm DC Plug is of high quality and reliable.

Figure 3.3.7 – DC adapter

SPECIFICATION:

- AC Input Voltage: 100 ~ 240 V.
- Output Voltage(V): 9.
- Output Current (Amp): 1.
- Rated Frequency (Hz): 50 – 60.
- Cable Length: Approx 1.1m

3.3.8 XD-42 5V/3.3V Dual Channel Solderless Breadboard Power Supply Module

This is a power supply module for the standard breadboard. Its concave design will not waste your proto zone. This breadboard power shield supports Mini USB power supply and power jack 7-12V supply. There are two independent power optional: 5V/3.3V. With this board, you can send power to both power lane of the breadboard, and each side has an on/off switch.

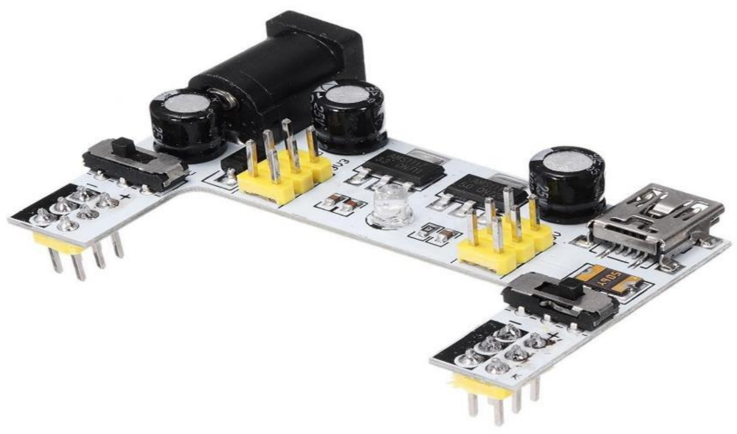


Figure 3.3.8 – Breadboard Supply module

Specifications and Features:

- Locking On/Off Switch
- Green LED Power Indicator
- Voltage Swapping Switch on both rail Channel
- Its concave design will not waste your proto zone.
- Easy to install on a breadboard with Compact portable size

3.3.9 5V Mini Submersible Pump

This DC 5V Mini Submersible Noiseless Water Pump is a low-cost, small size Submersible Pump. It can take up to 150 liters per hour with a very low current consumption from 300mA to 1A max. Just connect the tube pipe to the motor outlet, submerge it in water, and power it.

Figure 3.3.9 – DC water pump

Features :

- Simple Plug and Play operation.
- It Can operate on any 5V DC power supply like a Power bank
- 1 Meter cable with DC jack provides the convenience to connect and disconnect projects easily
- Can pump 2.5 litres per minute.

3.3.10 Breadboard

What does a bread board do?

A breadboard allows for easy and quick creation of temporary electronic circuits or to carry out experiments with circuit design Breadboards enable developers to easily connect components or wires thanks to the rows and columns of internally connected spring clips underneath the perforated plastic enclosure.

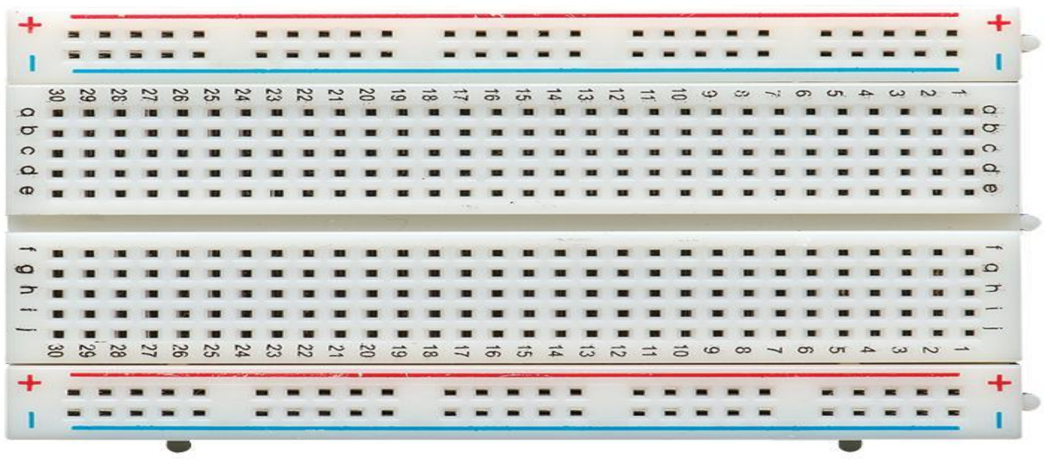


Figure 3.3.10 – Breadboard

3.4 Blynk IOT setup

Here we will setup blynk iot platform to view the sensor readings from anywhere in the world.

What are the new features of the Blynk IoT platform?

- Blynk.360: a web portal to monitor the connected devices, do over-the-air updates.
- A new approach to device model (template) creation.
- Blynk.inject: a WI-FI manager built into the new Blynk IoT app.
- All kinds of Automations (Eventor on steroids).
- Voice assistants with Amazon Alexa and Google Home (will be available soon).

Create an account in Blynk IoT Cloud Platform :

- First, you have to create an account in Blynk cloud platform. To sign up click on the following link
- Enter your email ID and click on Sign Up.
- After that, you will receive a verification email from Blynk.
- In that email click on Create Password to set the password for Blynk cloud account.

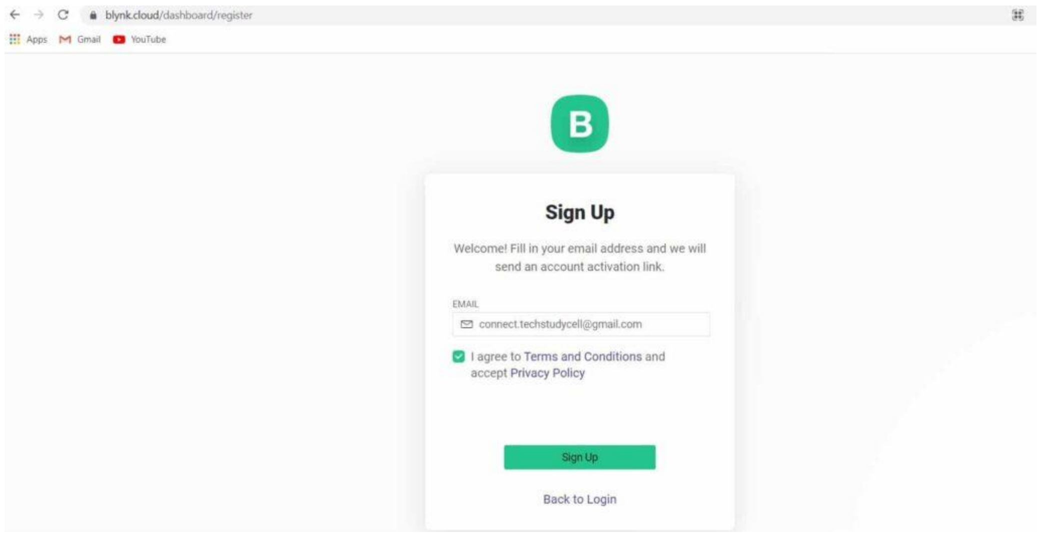

blynk.cloud/dashboard/register
Apps Gmail YouTube
Sign Up
Welcome! Fill in your email address and we will send an account activation link.
EMAIL
connect.techstudycell@gmail.com
I agree to Terms and Conditions and accept Privacy Policy
Sign Up
Back to Login

Figure 3.4(a) – Sign Up page

Create Template in Blynk IoT Cloud :

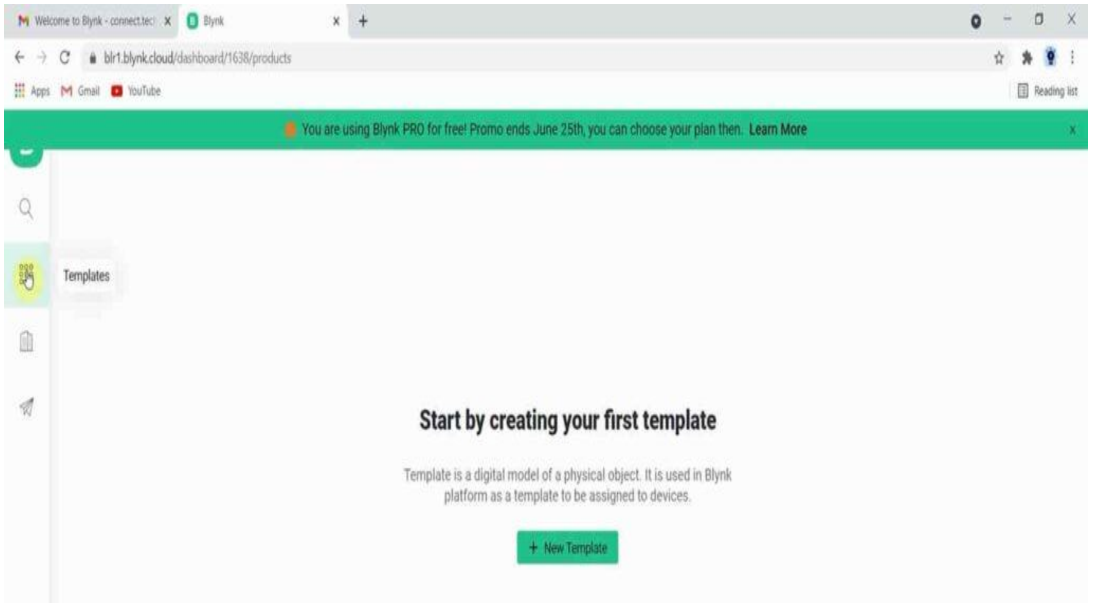
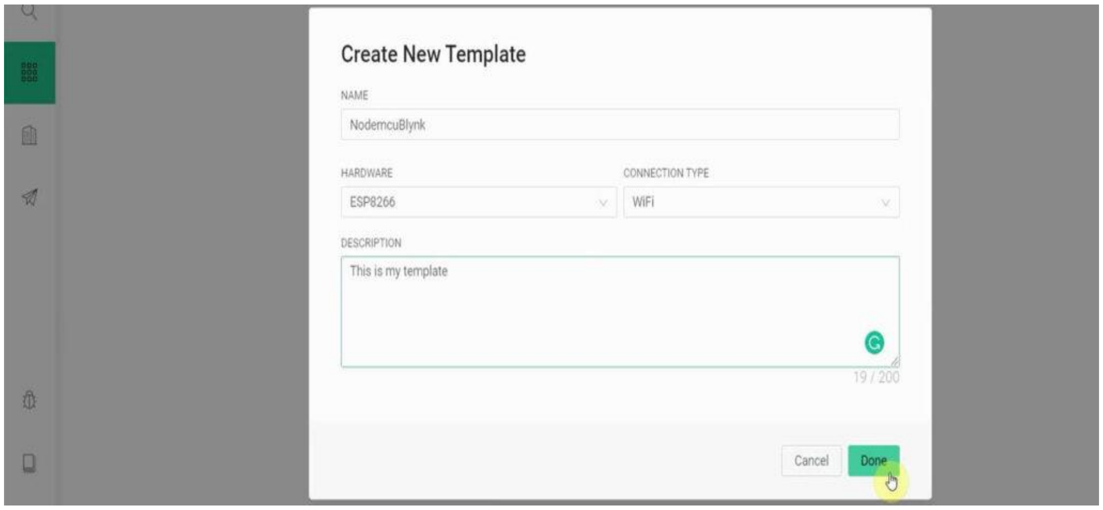

blr1.blynk.cloud/dashboard/1638/products
You are using Blynk PRO for free! Promo ends June 25th, you can choose your plan then. Learn More
Templates
Start by creating your first template
Template is a digital model of a physical object. It is used in Blynk platform as a template to be assigned to devices.
+ New Template

Figure 3.4(b) – Blynk home page

- After login to Blynk IoT account, first you have create a template. In the dashboard click on the Templates from the left side menu.

- Then click on New Template.


Create New Template

NAME: NodemcuBlynk

HARDWARE: ESP8266

CONNECTION TYPE: WiFi

DESCRIPTION: This is my template

19 / 200

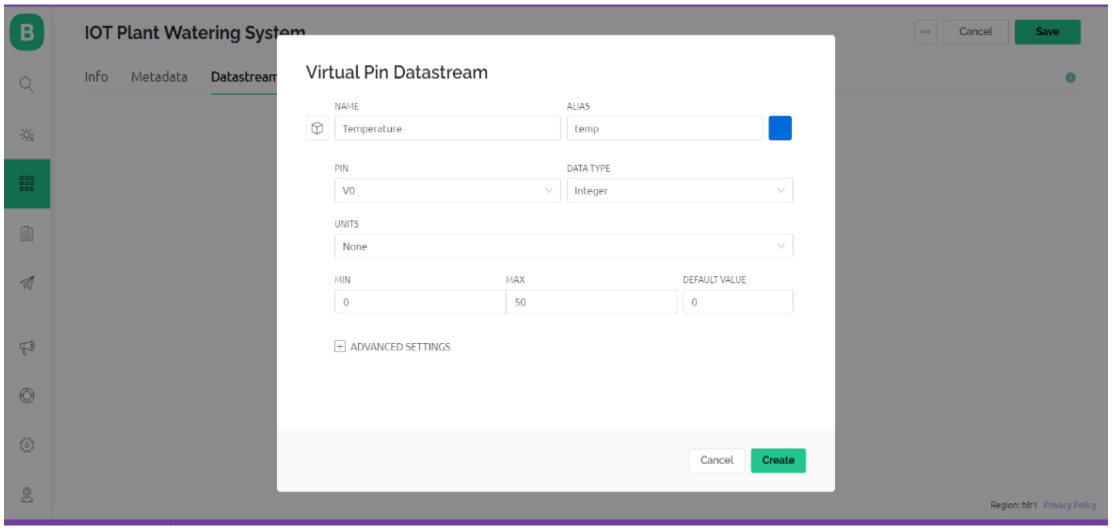
Figure 3.4(c) – Create template

- Then, you have to enter a name for that template.
- Select the microcontroller or board you will use in the project from the Hardware dropdown menu.
- The connection type should be WiFi.
- You can enter small description for the template. This field is optional.
- Now click on Done.

Create Datastreams in the Blynk IoT template :

- After creating the template, go to Datastreams tab.
- Then click on “New Datastream” and select Virtual Pin.
- After creating the template, go to Datastreams tab.
- Then click on “**New Datastream**” and select **Virtual Pin**.
- Now, you have to enter a **name** for this Datastream.
- Select the **Virtual Pin** and **Datatype** from the dropdown menu as per the requirement.
- Here I will control a relay, so I have selected Integer, For the Temparetute you have to select Double datatype and related UNIT.

- You can also set the **MAX, MIN,** and **default value** for the datastream.
- After that click on **Create**.



Virtual Pin Datastream

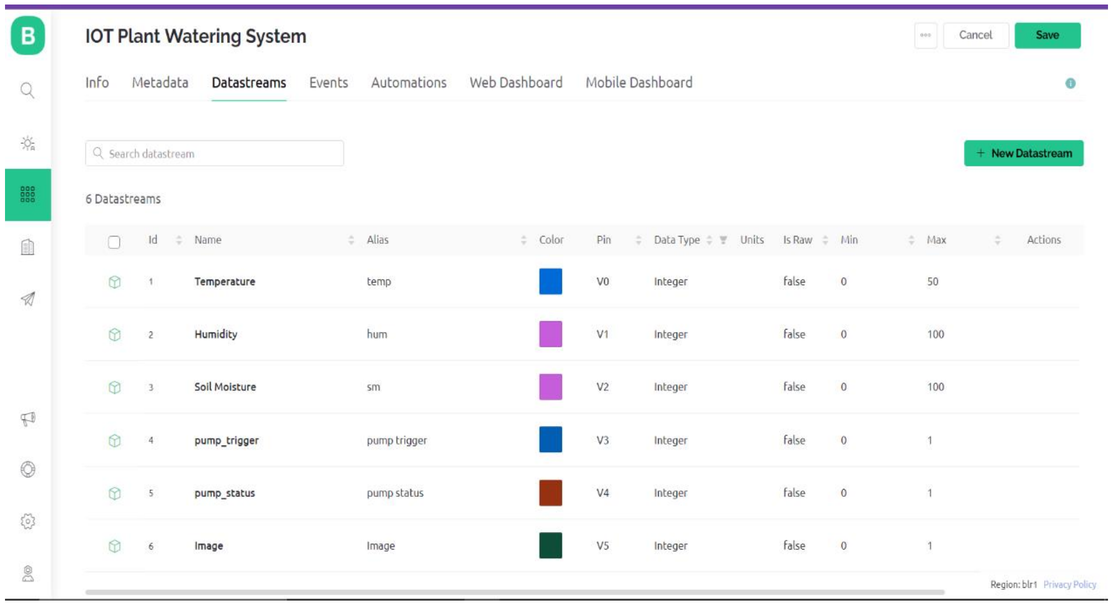
NAME	ALIAS
Temperature	temp

PIN	DATA TYPE
V0	Integer

UNITS: None

MIN	MAX	DEFAULT VALUE
0	50	0

Figure 3.4(d) – Create Datastream



6 Datastreams

Id	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Min	Max	Actions
1	Temperature	temp	🟦	V0	Integer		false	0	50	
2	Humidity	hum	🟪	V1	Integer		false	0	100	
3	Soil Moisture	sm	🟪	V2	Integer		false	0	100	
4	pump_trigger	pump trigger	🟦	V3	Integer		false	0	1	
5	pump_status	pump status	🟥	V4	Integer		false	0	1	
6	Image	Image	🟩	V5	Integer		false	0	1	

Figure 3.4(e) – Datastream

Install Blynk IoT app to set up Mobile Dashboard :

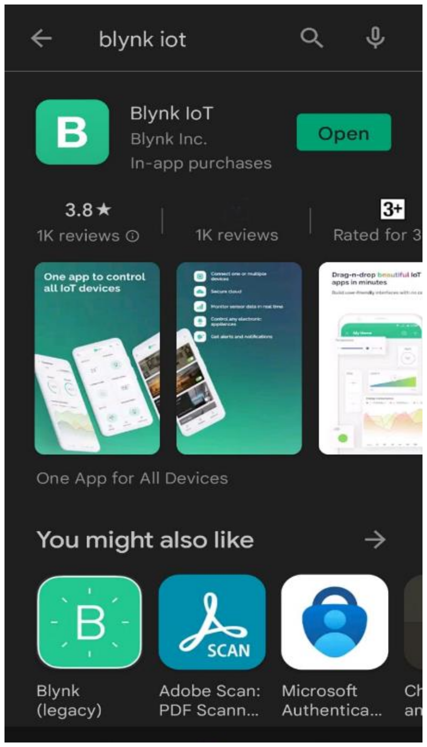


Figure 3.4(g) – Blynk IOT app

Steps to install the Blynk IoT App :

- Install the new **Blynk IoT** app from Google Play Store or App Store. Then **log in** to the Blynk IoT Platform.
- Tap on the **Developer Mode**.
- Tap on the template which you have already made in Blynk cloud. Now tap on the **3-dash icon** (on the right) to add widgets.

Steps to add widgets in Blynk IoT Mobile Dashboard:

- Tap on the **3-dash icon** to open the **Widget Box** and select the widget as per requirement. Here I have selected **Button**.
- Then tap on the widget to go to the **settings**.
- In the settings, enter the **name**, select the **Datastream**, Mode will be **Switch**. Then exit from the widget.
- After setting all the widgets tap on the **exit** button on the top.

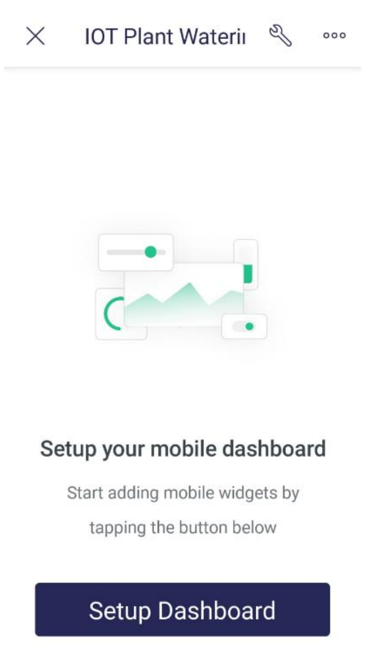
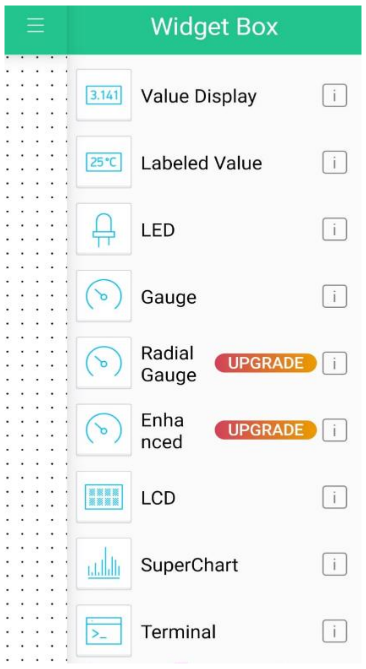

IOT Plant Waterii
Setup your mobile dashboard
Start adding mobile widgets by tapping the button below
Setup Dashboard


Figure 3.4(h) – Setting up the dashboard and widgets

Now both the **Web dashboard** and **Mobile Dashboard** of the Blynk IoT platform is ready. So we can program the microcontroller with Arduino IDE.

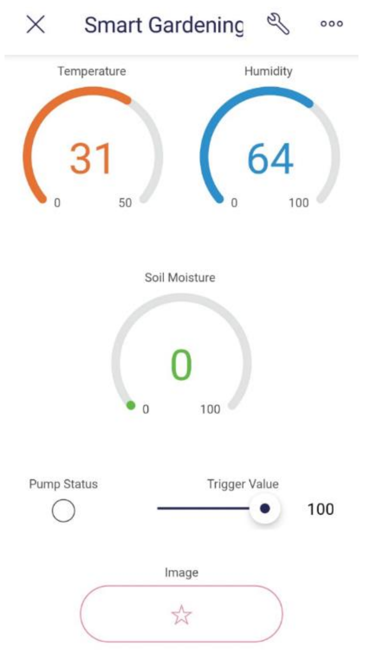


Figure 3.4(i) – Mobile app setup

3.5 Working principle

The smart garden monitors the temperature, humidity, light levels and soil moisture of the plant. It has an automated system that waters the plant when the soil is too dry . This maintains an ideal and consistent soil condition for the plant, and makes it convenient for those who tend to forget to water their plants regularly.

We will be using an NodeMCU to receive data from the sensors and control the different actuators. The surrounding temperature and air humidity will be recorded, as well as the soil moisture levels. These values will then be displayed on the OLED screen, which allow users to know the environmental conditions of the plants when they check on them.

When the soil moisture level goes below a threshold the water pump will start to run and pump water into the soil automatically. This is very convenient for users as they do not need to water their plants every time but instead let the system water their plants automatically based on the moisture level of the soil.

The temperature, humidity and soil moisture values will also be published to Blynk IOT. Through a mobile app, the data will be displayed using widgets where it shows real-time data coming from the sensors. This will allow users to view the real-time environmental conditions of the plants on the go .

The web page will also allow users to control the water pump by setting the desired trigger level.

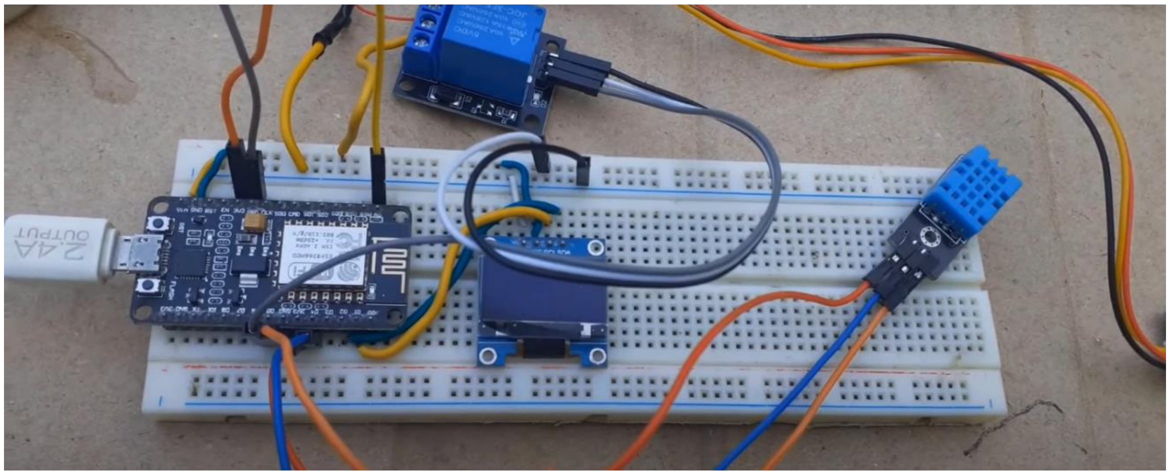
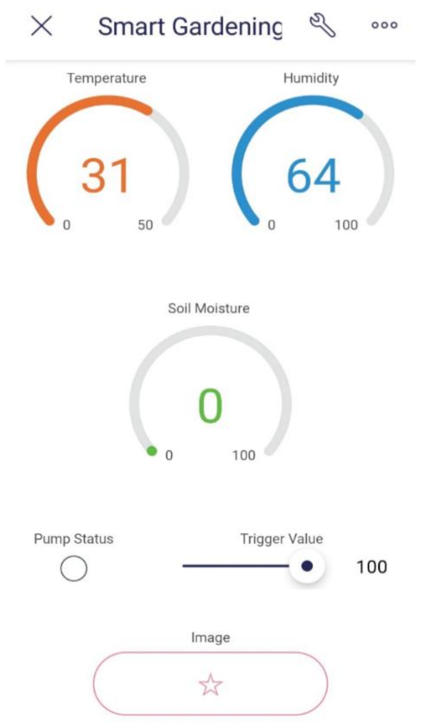


Figure 3.5(a) – System setup



Smart Gardening

Temperature: 31 (0–50)

Humidity: 64 (0–100)

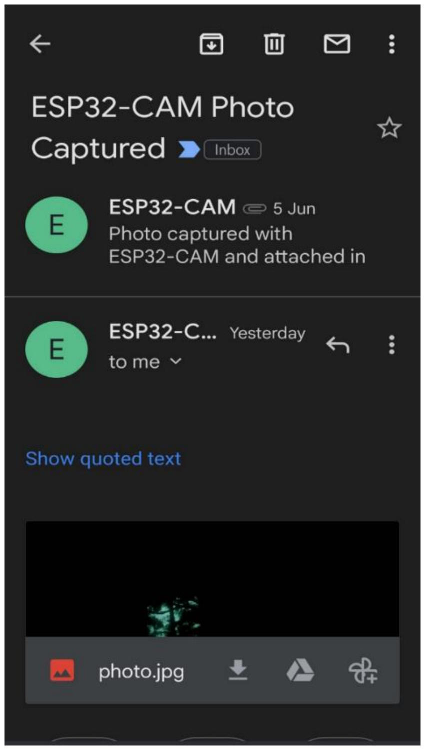
Soil Moisture: 0 (0–100)

Pump Status

Trigger Value: 100

Image

Figure 3.5(b) – Real time data monitoring


ESP32-CAM Photo Captured

Inbox

ESP32-CAM 5 Jun

Photo captured with ESP32-CAM and attached in

ESP32-C... Yesterday

to me

Show quoted text

photo.jpg

Figure 3.5(c) – Photo captured and sent through email

CHAPTER 4

4.1 SOFTWARE DEVOLOPMENT

- On VS Code, click on the PlartfomIO **Home** icon. Click on **+ New Project** to start a new project.

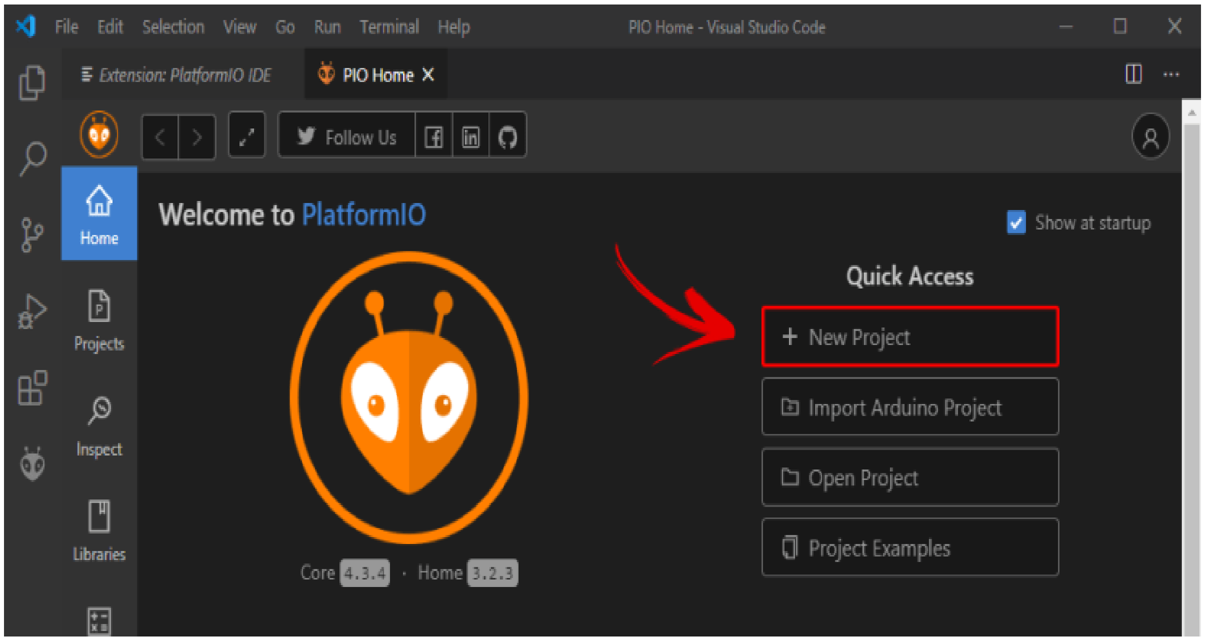
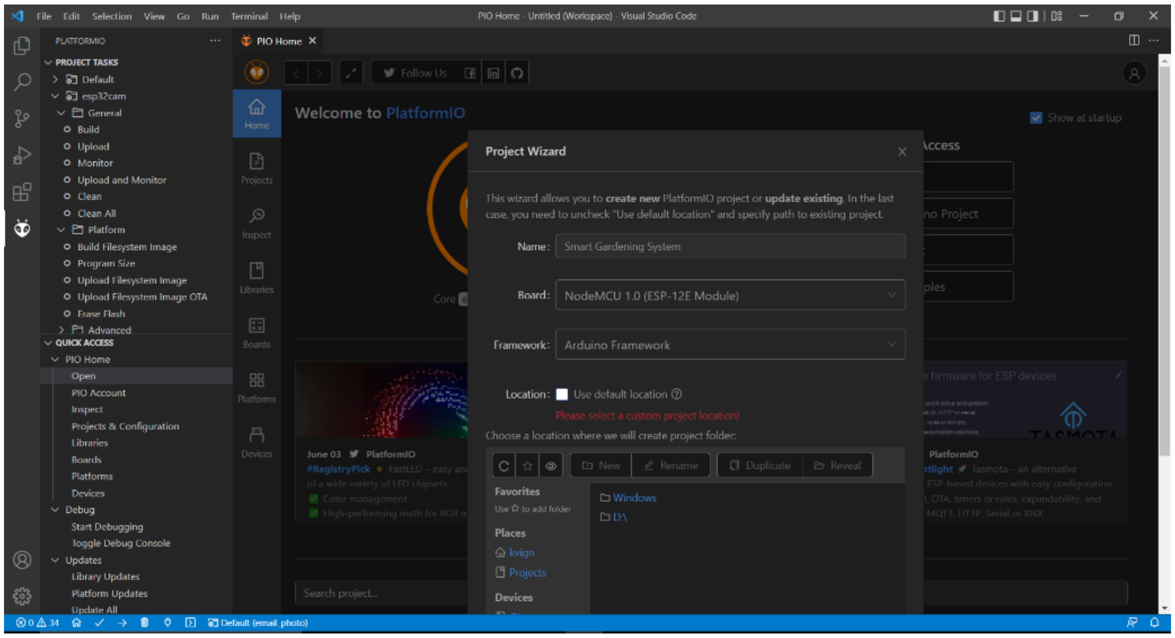


Figure 4.1(a) – Create project

- Give your project a name and select the board you're using. In our case, we're using the NodeMCU 1.0 (ESP-12E Module).
- The Framework should be "**Arduino**" to use the Arduino core.
- You can choose the default location to save your project or a custom location.



Project Wizard

This wizard allows you to **create new** PlatformIO project or **update existing**. In the last case, you need to uncheck "Use default location" and specify path to existing project.

Name: Smart Gardening System

Board: NodeMCU 1.0 (ESP-12E Module)

Framework: Arduino Framework

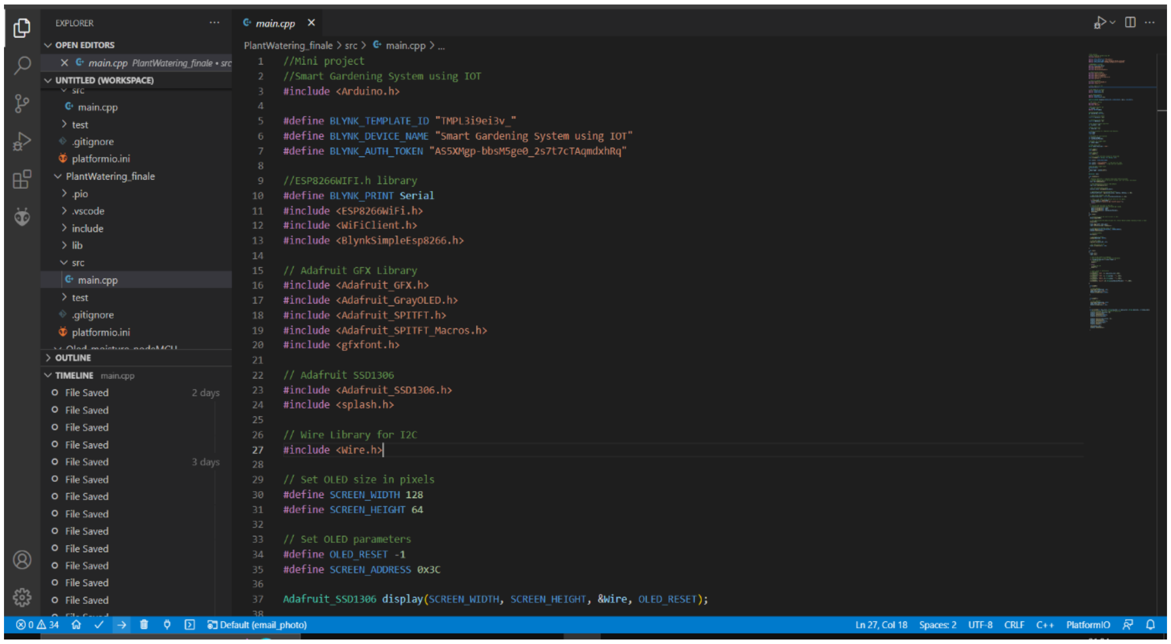
Location: ☐ Use default location

Please select a custom project location!

Choose a location where we will create project folder:

Figure 4.1(b) – Save the project

- After that, press **Ctrl+S** or go to **File** > **Save** to save the file.
- Now, you can click on the Upload icon to compile and upload the code. Alternatively, you can go to the PIO Project Tasks menu and select **Upload**.



PlantWatering_finale > src > main.cpp > ...

```cpp
//Mini project
//Smart Gardening System using IOT
#include <Arduino.h>

#define BLYNK_TEMPLATE_ID "TMPL3i9ei3v_"
#define BLYNK_DEVICE_NAME "Smart Gardening System using IOT"
#define BLYNK_AUTH_TOKEN "AS5XMgp-bbsM5ge0_2s7t7cTAqmdxhRq"

//ESP8266WIFI.h library
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp8266.h>

// Adafruit GFX Library
#include <Adafruit_GFX.h>
#include <Adafruit_GrayOLED.h>
#include <Adafruit_SPITFT.h>
#include <Adafruit_SPITFT_Macros.h>
#include <gfxfont.h>

// Adafruit SSD1306
#include <Adafruit_SSD1306.h>
#include <splash.h>

// Wire Library for I2C
#include <Wire.h>

// Set OLED size in pixels
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

// Set OLED parameters
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
```

Figure 4.1(c) – Upload the code

4.2 Arduino code of NodeMCU

```
//Mini project
//Smart Gardening System using IOT
#include <Arduino.h>

#define BLYNK_TEMPLATE_ID "TMPL3i9ei3v_"
#define BLYNK_DEVICE_NAME "Smart Gardening System using IOT"
#define BLYNK_AUTH_TOKEN "AS5XMgp-bbsM5ge0_2s7t7cTAqmdxhRq"

//ESP8266WIFI.h library
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp8266.h>

// Adafruit GFX Library
#include <Adafruit_GFX.h>
#include <Adafruit_GrayOLED.h>
#include <Adafruit_SPITFT.h>
#include <Adafruit_SPITFT_Macros.h>
#include <gfxfont.h>

// Adafruit SSD1306
#include <Adafruit_SSD1306.h>
```

```
#include <splash.h>

// Wire Library for I2C
#include <Wire.h>

// Set OLED size in pixels
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

// Set OLED parameters
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

// DHT sensor library
#include<DHT.h>
#include <DHT_U.h>

//DHT sensor digital pin
int DHTPIN = 14;
#define DHTTYPE DHT11

//Initialize DHT sensor
```

```
DHT dht(DHTPIN, DHTTYPE);

// Constant for dry sensor
const int DryValue = 1023;

// Constant for wet sensor
const int WetValue = 317;

//Soil moisture sensor input Port
int sm_sensor = A0;

//Relay control pin
int RELAY_OUT = 12;

// Define variables for Temperature and Humidity
float temp;
float hum;

// Variables for soil moisture
int moisture_value;
int soilMoisturePercent;

//Variable for trigger level
int pump_trigger;
```

```
// Pump Status Text
String pump_status_text = "OFF";

//turn pump on
void pumpOn();

//turn pump off
void pumpOff();

// Prints to OLED and holds display for delay_time
void printOLED(int, String, int, String, int);

char auth[] = BLYNK_AUTH_TOKEN;

char ssid[] = "AndroidAP5EAD";  // type your wifi name
char pass[] = "indo556kvb";  // type your wifi password

//Get the pump trigger value
BLYNK_WRITE(V4)
{
 pump_trigger = param.asInt();
}
```

```
BlynkTimer timer;

void sendSensor()
{
  //Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  hum = dht.readHumidity();

  // Read temperature as Celsius (the default)
  temp = dht.readTemperature();

  //Get the analog sensor value
  moisture_value= analogRead(sm_sensor);

  // Determine soil moisture percentage value
  soilMoisturePercent = map(moisture_value, DryValue, WetValue, 0, 100);

  // Keep values between 0 and 100
  soilMoisturePercent = constrain(soilMoisturePercent, 0, 100);

  // Check if any reads failed and exit early (to try again).
   if (isnan(hum) || isnan(temp)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
```

```
 }
 // You can send any value at any time.
 // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V0, temp);
  Blynk.virtualWrite(V1, hum);
  Blynk.virtualWrite(V2, soilMoisturePercent);
  delay(500);
}
void setup()
{
 // Initialize serial and wait for port to open:
 Serial.begin(9600);

 // This delay gives the chance to wait for a Serial Monitor without blocking if none
is found
 delay(1500);

 Blynk.begin(auth, ssid, pass);
 timer.setInterval(100L, sendSensor);

 // Initialize I2C display using 3.3 volts
 display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);
 display.clearDisplay();

 // Initialize DHT11
```

```
  dht.begin();

  // Set Relay as Output
  pinMode(RELAY_OUT, OUTPUT);

  // Turn off relay
  digitalWrite(RELAY_OUT, LOW);

  //Set pump status as off
  Blynk.virtualWrite(V3, LOW);

}

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

  // See if pump should be triggered
  // See if moisture is below or equal to threshold
  if (soilMoisturePercent <= pump_trigger) {
    // Turn pump on
    pumpOn();
  } else {
    // Turn pump off
    pumpOff();
  }

  // Cycle values on OLED Display
  // Pump Status
  printOLED(35, "PUMP", 40, pump_status_text, 2000);
  // Temperature
  printOLED(35, "TEMP", 10, String(temp) + "C", 2000);
  // Humidity
  printOLED(30, "HUMID", 10, String(hum) + "%", 2000);
  // Moisture
  printOLED(35, "Soil.M", 30, String(soilMoisturePercent)  + "%", 2000);

}

 void pumpOn()
 {
   // Turn pump on
   digitalWrite(RELAY_OUT, LOW);
   pump_status_text = "ON";
   Blynk.virtualWrite(V3, HIGH);
```

```
}



void pumpOff()

{

  // Turn pump off

  digitalWrite(RELAY_OUT, HIGH);

  pump_status_text = "OFF";

  Blynk.virtualWrite(V3, LOW);


}


void printOLED(int top_cursor, String top_text, int main_cursor, String main_text,
int delay_time){

  // Prints to OLED and holds display for delay_time

  display.setCursor(top_cursor, 0);

  display.setTextSize(2);

  display.setTextColor(WHITE);

  display.println(top_text);


  display.setCursor(main_cursor, 40);

  display.setTextSize(3);

  display.setTextColor(WHITE);

  display.print(main_text);

  display.display();
  delay(delay_time);
  display.clearDisplay();

}
```

4.3 Arduino code Esp32 CAM

```
#include "esp_camera.h"
#include "SPI.h"
#include "driver/rtc_io.h"
#include "ESP32_MailClient.h"
#include <FS.h>
#include <SPIFFS.h>
#include <WiFi.h>

// REPLACE WITH YOUR NETWORK CREDENTIALS
const char* ssid = "AndroidAP5EAD";
const char* password = "indo556kvb";

// To send Emails using Gmail on port 465 (SSL), you need to create an app password: https://support.google.com/accounts/answer/185833
#define emailSenderAccount    "smartiotgardening@gmail.com"
```

```
#define emailSenderPassword    "czpydvayzzlpagir"
#define smtpServer           "smtp.gmail.com"
#define smtpServerPort       465
#define emailSubject         "ESP32-CAM Photo Captured"
#define emailRecipient       "kvb307@gmail.com"

#define CAMERA_MODEL_AI_THINKER

#if defined(CAMERA_MODEL_AI_THINKER)
 #define PWDN_GPIO_NUM     32
 #define RESET_GPIO_NUM    -1
 #define XCLK_GPIO_NUM      0
 #define SIOD_GPIO_NUM     26
 #define SIOC_GPIO_NUM     27

 #define Y9_GPIO_NUM       35
 #define Y8_GPIO_NUM       34
 #define Y7_GPIO_NUM       39
 #define Y6_GPIO_NUM       36
 #define Y5_GPIO_NUM       21
 #define Y4_GPIO_NUM       19
 #define Y3_GPIO_NUM       18
 #define Y2_GPIO_NUM        5
 #define VSYNC_GPIO_NUM    25
```

```
  #define HREF_GPIO_NUM     23
  #define PCLK_GPIO_NUM     22
#else
  #error "Camera model not selected"
#endif



// The Email Sending data object contains config and data to send
SMTPData smtpData;

// Photo File Name to save in SPIFFS
#define FILE_PHOTO "/photo.jpg"
void capturePhotoSaveSpiffs( void );
void sendPhoto( void );
// Callback function to get the Email sending status
void sendCallback(SendStatus msg) {
  //Print the current status
  Serial.println(msg.info());
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector

  Serial.begin(115200);
  Serial.println();

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi...");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  if (!SPIFFS.begin(true)) {
    Serial.println("An Error has occurred while mounting SPIFFS");
    ESP.restart();
  }
  else {
    delay(500);
    Serial.println("SPIFFS mounted successfully");
  }

  // Print ESP32 Local IP Address
  Serial.print("IP Address: http://");
  Serial.println(WiFi.localIP());
```

```
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;

if(psramFound()){
  config.frame_size = FRAMESIZE_UXGA;
```

```
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

  // Initialize camera
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  pinMode(4,OUTPUT);
  digitalWrite(4,HIGH);

  capturePhotoSaveSpiffs();
  sendPhoto();
}

void loop() {
}

// Check if photo capture was successful
bool checkPhoto( fs::FS &fs ) {
  File f_pic = fs.open( FILE_PHOTO );
  unsigned int pic_sz = f_pic.size();
  return ( pic_sz > 100 );
}

// Capture Photo and Save it to SPIFFS
void capturePhotoSaveSpiffs( void ) {
  camera_fb_t * fb = NULL; // pointer
  bool ok = 0; // Boolean indicating if the picture has been taken correctly

  do {
    // Take a photo with the camera
    Serial.println("Taking a photo...");

    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      return;
    }

    // Photo file name
    Serial.printf("Picture file name: %s\n", FILE_PHOTO);
    File file = SPIFFS.open(FILE_PHOTO, FILE_WRITE);

    // Insert the data in the photo file
    if (!file) {
      Serial.println("Failed to open file in writing mode");
    }
    else {
      file.write(fb->buf, fb->len); // payload (image), payload length
      Serial.print("The picture has been saved in ");
      Serial.print(FILE_PHOTO);
      Serial.print(" - Size: ");
      Serial.print(file.size());
      Serial.println(" bytes");
    }
    // Close the file
    file.close();
    esp_camera_fb_return(fb);

    // check if file has been correctly saved in SPIFFS
    ok = checkPhoto(SPIFFS);
```

```
  } while ( !ok );
}

void sendPhoto( void ) {
  // Preparing email
  Serial.println("Sending email...");
  // Set the SMTP Server Email host, port, account and password
  smtpData.setLogin(smtpServer, smtpServerPort, emailSenderAccount,
emailSenderPassword);

  // Set the sender name and Email
  smtpData.setSender("ESP32-CAM", emailSenderAccount);

  // Set Email priority or importance High, Normal, Low or 1 to 5 (1 is highest)
  smtpData.setPriority("High");

  // Set the subject
  smtpData.setSubject(emailSubject);

  // Set the email message in HTML format
  smtpData.setMessage("<h2>Photo captured with ESP32-CAM and attached in this
email.</h2>", true);
  // Set the email message in text format
  //smtpData.setMessage("Photo captured with ESP32-CAM and attached in this
email.", false);
```

```
  // Add recipients, can add more than one recipient
  smtpData.addRecipient(emailRecipient);
  //smtpData.addRecipient(emailRecipient2);

  // Add attach files from SPIFFS
  smtpData.addAttachFile(FILE_PHOTO, "image/jpg");
  // Set the storage type to attach files in your email (SPIFFS)
  smtpData.setFileStorageType(MailClientStorageType::SPIFFS);

  smtpData.setSendCallback(sendCallback);

  // Start sending Email, can be set callback function to track the status
  if (!MailClient.sendMail(smtpData))
    Serial.println("Error sending Email, " + MailClient.smtpErrorReason());

  // Clear all data from Email object to free memory
  smtpData.empty();
}
```

CHAPTER 5

RESULTS AND DISCUSSION

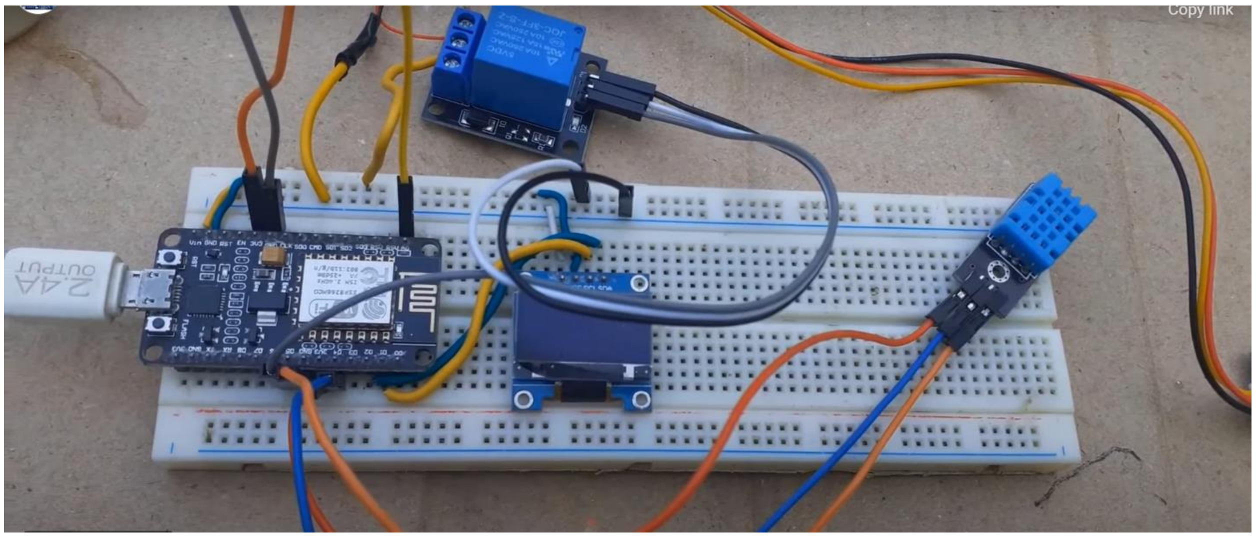


Fig 5(a) – Real time implementation

Realtime implementation :

- The values of temperature,humidity and soil moisture can be monitored through oled screen and through the blynk iot app.
- The values displayed on the blynk iot app can be monitored from anywhere anywhere in the world.
- The value are also shown as a historical graph in tab 2.
- When the values of temperature,humidity and soil moisture varies in the surrounding,they are immediately updated on the blynk app.

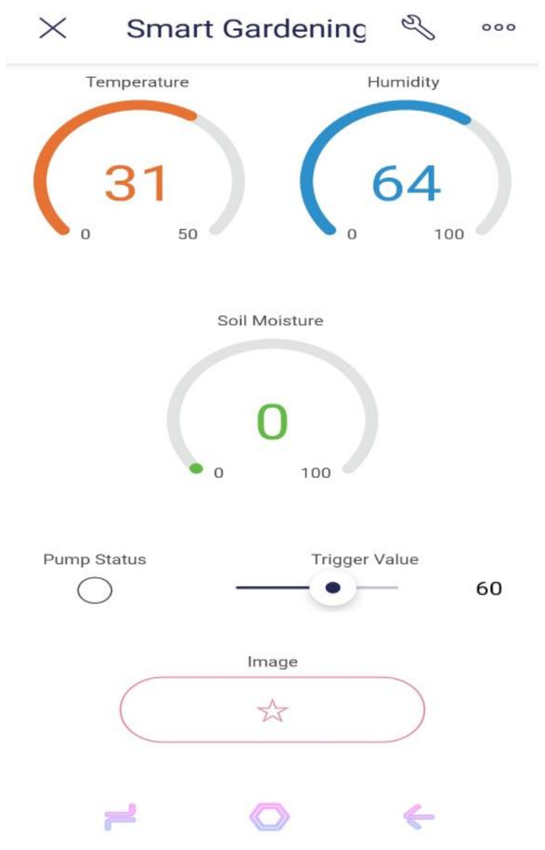


Fig 5(b) – Real time monitoring through blynk app

- The soil moisture sensor measures the soil moisture content in the soil and relays it back to the NodeMCU.
- When the soil moisture content in the soil drops below a certain threshold,NodeMCU turns the relay ON.
- We have set the pump Trigger value as the threshold for turning on the pump.
- This will trigger the water pump and the plant is watered.

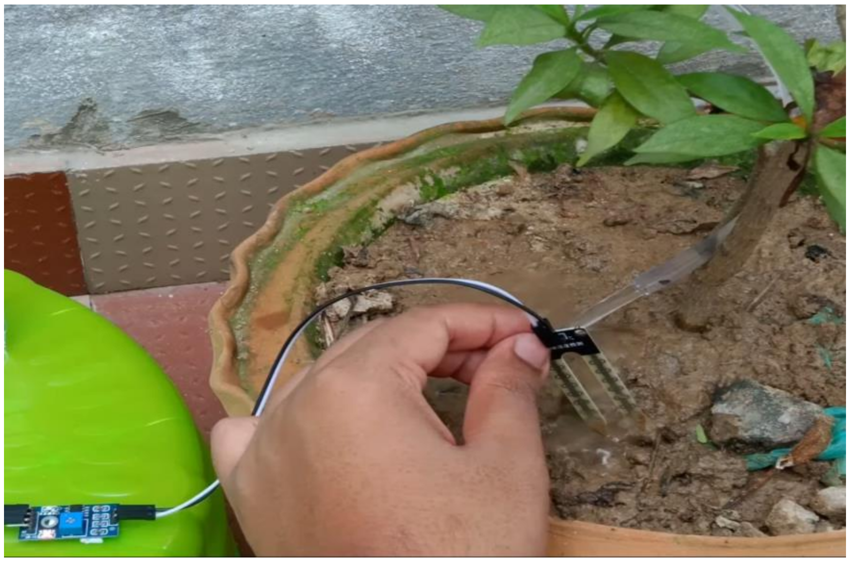


Fig 5(c) – Pump ON mode

- When the soil moisture level increases above the threshold , NodeMCU turns the relay OFF.
- This will stop the water pump.

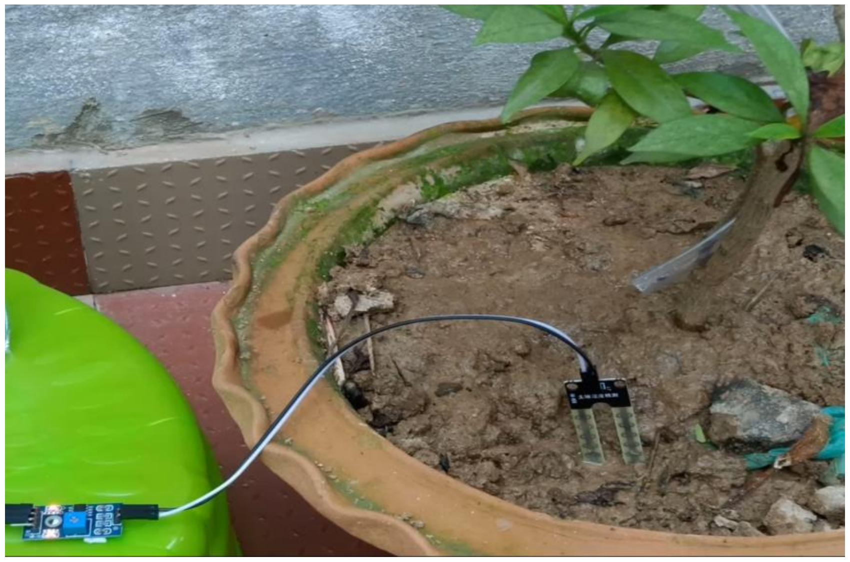


Fig 5(d) – Pump OFF mode

- We can manually turn ON the water pump by setting the trigger value.

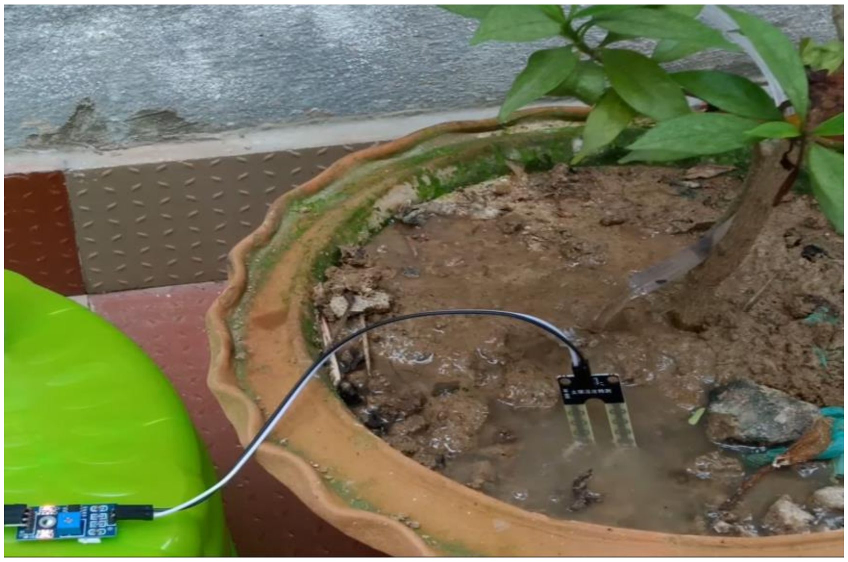


Fig 5(e) – Setting Trigger value

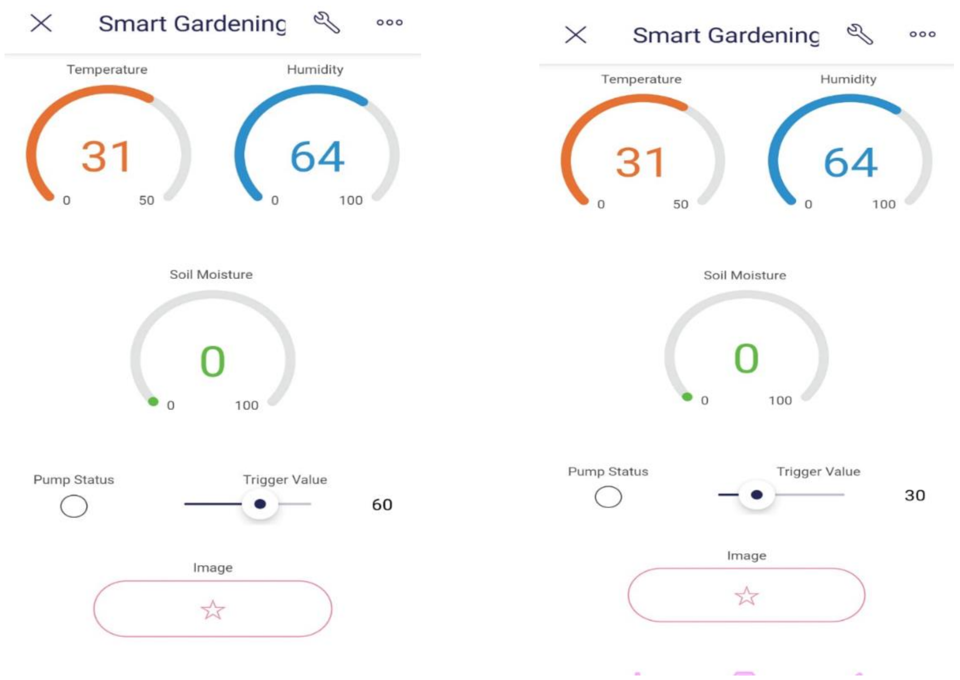
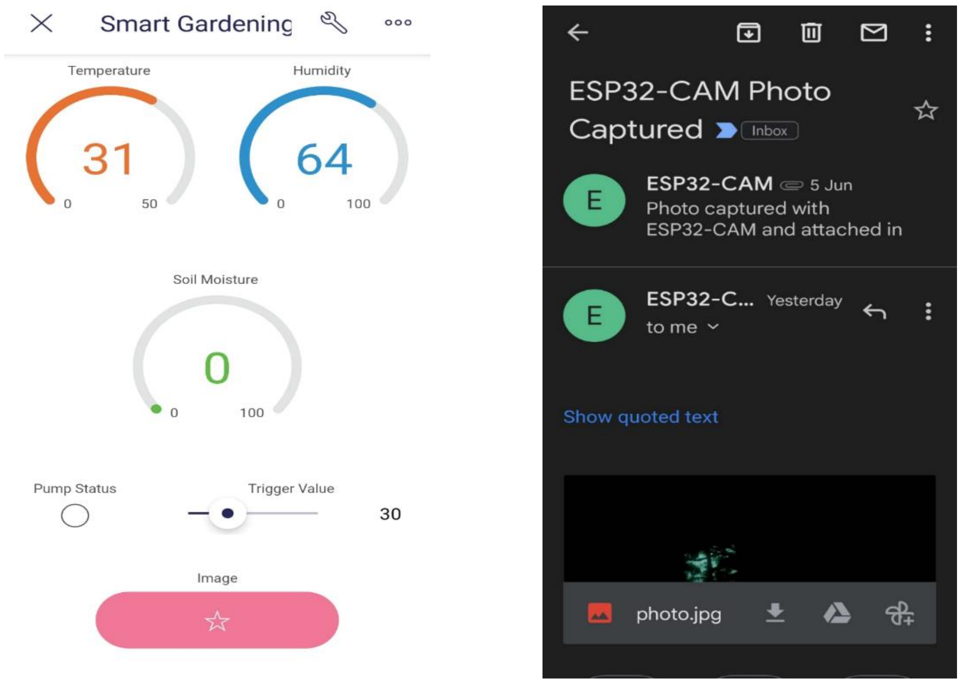


Fig 5(f) – Setting trigger level through blynk app

- We can request an image to monitor the real time condition of plant by clicking on the image button in the app.



ESP32-CAM Photo Captured

ESP32-CAM 5 Jun
Photo captured with ESP32-CAM and attached in

ESP32-C... Yesterday
to me

Fig 5(g) – Requesting image through blynk app

CHAPTER 6

CONCLUSION

This Smart gardening system can automatically water the plants by analyzing the soil parameters, and it will optimize the usage of water by reducing wastage. By providing the blynk app the user can monitor and control the water requirement in the garden. The system will minimize human intervention and provide a time and cost effective system to water the plants and monitor them from anywhere in the world. This system also has a trigger value to manually set the trigger level for the pump to water the plant at certain value. This system is needed since different plants require different soil moisture levels for optimal growth. We are all able to receive the image of the plant for relatime monitoring from anywhere in the world.

FUTURE SCOPE

This automated Smart Gardening System using IoT is found to be cost-effective for enhancing the techniques to preserve water resources. This system helps to grow plants by working automatically and smartly. With placing multiple sensors in the soil, water can be only provided to the required piece of land. This system requires less maintenance so it is easily affordable by all. This system helps to reduce water consumption. With using this system the plant growth increases to a great extent. As per future perspective, this system can be the more intelligent system which predicts user actions, nutrient level of the plant etc. With using Machine Learning algorithms more advancements can be done in the future which will help plant lover a lot and water consumption can also be reduced in gardening.

CHAPTER 7

REFERENCES

[1]http://countrystudies.us/laos/45.htm

[2]http://www.nationsencyclopedia.com/economies/Asia-and-the-Pacific/Laos-AGRICULTURE.html

[3]Suprabha, J. and Shailesh, H. (2014) Android based Automated Irrigation System using Raspberry Pi. International Journal of Science and Research, 5, Issue 6.

[4]Bhagyashree, K., Chate and Rana, J.G. (2016) Smart Irrigation System Using Raspberry Pi. International Research Journal of Engineering and Technology, 3, Issue 5

[5]Mahesh, A. Reddy, Raghava Rao, K. (2016) An Android based Automatic Irrigation System Using a WSN and GPRS Module. Indian Journal of Science and Technology, 9, Issue 30.https://doi.org/10.17485/ijst/2016/v9i30/98719

[6]Gavali, M.S., Dhus, B.J. and Vitekar, A.B. (2016) A Smart Irrigation System for Agriculture Base on Wireless Sensors. International Journal of Innovative Research in Science, Engineering and Technology, 5, Issue 5.

[7]Harishankar, S., Kumar, R.S., Sudharsan, K.P., Vignesh, U. and Viveknath, T. (2014) Solar Powered Smart Irrigation System. Advance in Electronic and Electric Engineering, 4, 341-346.

[8]Jadhav, S. and Hambarde, H. (2016) Android Based Automated Irrigation System using Rasspberry Pi. International Journal of Science and Research, 5, Issue 6.

[9]Ata, S.R. (2016) Web Based Automatic Irrigation System Using Wireless Sensor Network and Embedded Linux Board. International Journal of Advancement in Engineering Technology Management & Applied Science, 3, Issue 2.