SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be University)
FACULTY OF SCIENCE AND HUMANITIES

DEPARTMENT OF COMPUTER SCIENCE

PRACTICAL RECORD

NAME OF THE STUDENT	:
REGISTER NO.	: RA23322410500
PROGRAMME	: MCA
SEMESTER/YEAR	: III/ II
COURSE CODE	: PCA20D09J
COURSE NAME	: INTERNET OF THINGS

OCTOBER 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be University)
FACULTY OF SCIENCE AND HUMANITIES

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that the bonafide record work is done by________________________ Register No. : RA23322410500 for the course "INTERNET OF THINGS **(PCA20D09J)"** at SRM Institute of Science and Technology, Tiruchirappalli in October 2024.

STAFF IN-CHARGE **HEAD OF THE DEPARTMENT**

Submitted for the University Practical Examination held at SRM Institute of Science and Technology, Tiruchirappalli on__________________________.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

Ex. No.1: Define and Explain Eclipse IoT Project

Date: 02/07/24

- **Eclipse IoT** refers to a set of open-source projects under the Eclipse Foundation, focused on building an open Internet of Things (IoT) ecosystem. The goal is to provide a comprehensive platform and set of tools for developing, deploying, and managing IoT solutions, from devices to cloud-based applications.

  Key Components of Eclipse IoT:
- **Device Connectivity and Communication**:
  - **Eclipse Paho**: A project that provides client implementations of MQTT (Message Queuing Telemetry Transport), a lightweight messaging protocol for IoT.
  - **Eclipse Milo**: An open-source implementation of the OPC UA (Open Platform Communications Unified Architecture) standard, which is used for industrial automation and IoT.

  IoT Gateways:
- **Eclipse Kura**: A framework for building IoT gateways, offering features like device management, network configuration, and communication protocols.

  IoT Cloud Platforms:
- **Eclipse Hono**: Provides a uniform API for connecting large numbers of IoT devices to a cloud-based backend.

  Eclipse Kapua: An IoT cloud platform that offers device management, data management, and application development services.

  Development Tools and Frameworks:
- **Eclipse Vorto**: A tool to define IoT information models and generate code for different platforms.
- **Eclipse Ditto**: A framework for managing digital twins, which are digital representations of physical IoT devices.

  Security and Standards:
- **Eclipse Leshan**: A device management server and client based on the Lightweight M2M (LwM2M) protocol, which is designed to ensure secure communication in IoT environments.

Purpose and Importance:

- **Open Standards**: Eclipse IoT projects focus on open standards to avoid vendor lock-in and ensure interoperability across different devices and platforms.
- **Community-Driven**: The projects are community-driven, which allows for rapid innovation and adaptation to the evolving IoT landscape.
- **End-to-End Solutions**: From device connectivity to cloud integration and data management, Eclipse IoT provides an end-to-end ecosystem that developers can leverage to build robust IoT solutions.

Ex. No. : 2 List and summarize few Eclipse IoT Projects

Date: 10/07/24

- **Eclipse Mosquitto:** An open-source MQTT broker, which is a lightweight messaging protocol designed for small sensors and mobile devices optimized for high-latency or unreliable networks.
- **Eclipse Kura:** A framework for building IoT gateways. It provides a set of common services for Java developers, including device communication, data management, and remote management.
- **Eclipse Paho:** A set of MQTT client libraries for different programming languages like Java, C, Python, JavaScript, and more. It‟s designed for machine-to-machine (M2M) communication.
- **Eclipse Hono:** Provides uniform APIs for connecting IoT devices to a backend infrastructure, facilitating reliable data ingestion and command & control.

- **Eclipse Ditto:** A framework for managing digital twins, enabling the representation of physical devices in the digital world and facilitating communication between devices and applications
- **Eclipse Vorto:** A tool for defining IoT device information models, which helps in standardizing the way devices and their capabilities are described.
- **Eclipse IoT Packages:** A set of tools and libraries for IoT developers that include Eclipse Concierge, Eclipse Californium (for CoAP protocol), and more.
- **Eclipse Kapua:** A modular IoT cloud platform that manages and integrates devices on the edge. It provides a framework for managing IoT gateways and smart edge devices.

- **Eclipse Mita:** A programming language designed specifically for IoT, focusing on reducing complexity in developing IoT applications.
- **Eclipse Unide**: A project aimed at providing open and standardized data formats and services for Industry 4.0 applications, particularly in the area of production process data exchange.
- **Eclipse Whiskers**: A software platform for building edge computing systems, focusing on low-latency processing close to the IoT devices.

- **Eclipse Leshan**: A Java implementation of the Lightweight M2M (LwM2M) protocol, used for managing constrained devices.

- **Eclipse Agail**: A lightweight messaging protocol designed for IoT applications, similar to MQTT but optimized for different use cases or environments.
- **Eclipse Fog05**: A fog computing platform that enables the orchestration of compute, storage, and networking resources across cloud and edge environments. It‟s designed for IoT and distributed systems, allowing for dynamic and efficient resource management close to the data source.
- **Eclipse 4diac**: An open-source industrial automation framework that supports the IEC 61499 standard for distributed control systems. It‟s designed for developing industrial IoT (IIoT) applications, offering a flexible and scalable solution for industrial automation.
- **Eclipse Oniro**: This project focuses on developing an open-source operating system for IoT devices, aiming to create a secure, flexible, and interconnected ecosystem of devices.
- **Eclipse Cyclone DDS**: An implementation of the Data Distribution Service (DDS) protocol for real-time systems, particularly useful in environments where low latency and high reliability are crucial, such as autonomous vehicles and industrial automation.

Ex. No. :3 Arduino Installation and Blink LED using Node MCU Esp8266 Required Hardware

Date: 16/07/24

Requirements:

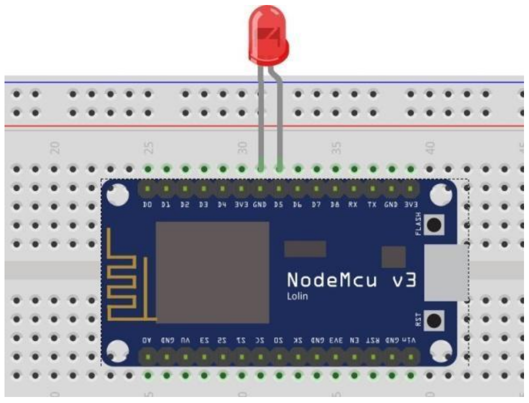
1. **Node MCU**
2. **LED**
3. **Bread Board**
4. **Micro**

USB Cable
Software

1. Arduino
IDE Hardware
Connection

1. **Connect LED +ve pin with D5 pin**
2. **Connect LED -ve pin**

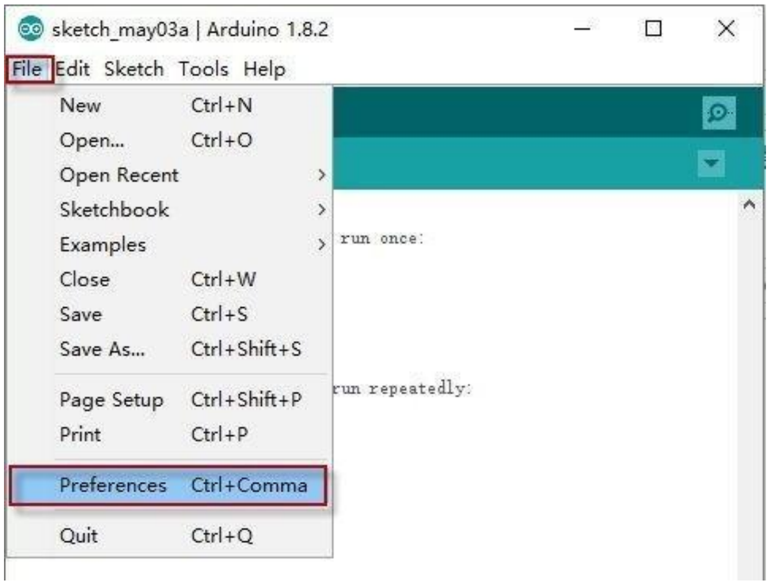
with GNDSchematic



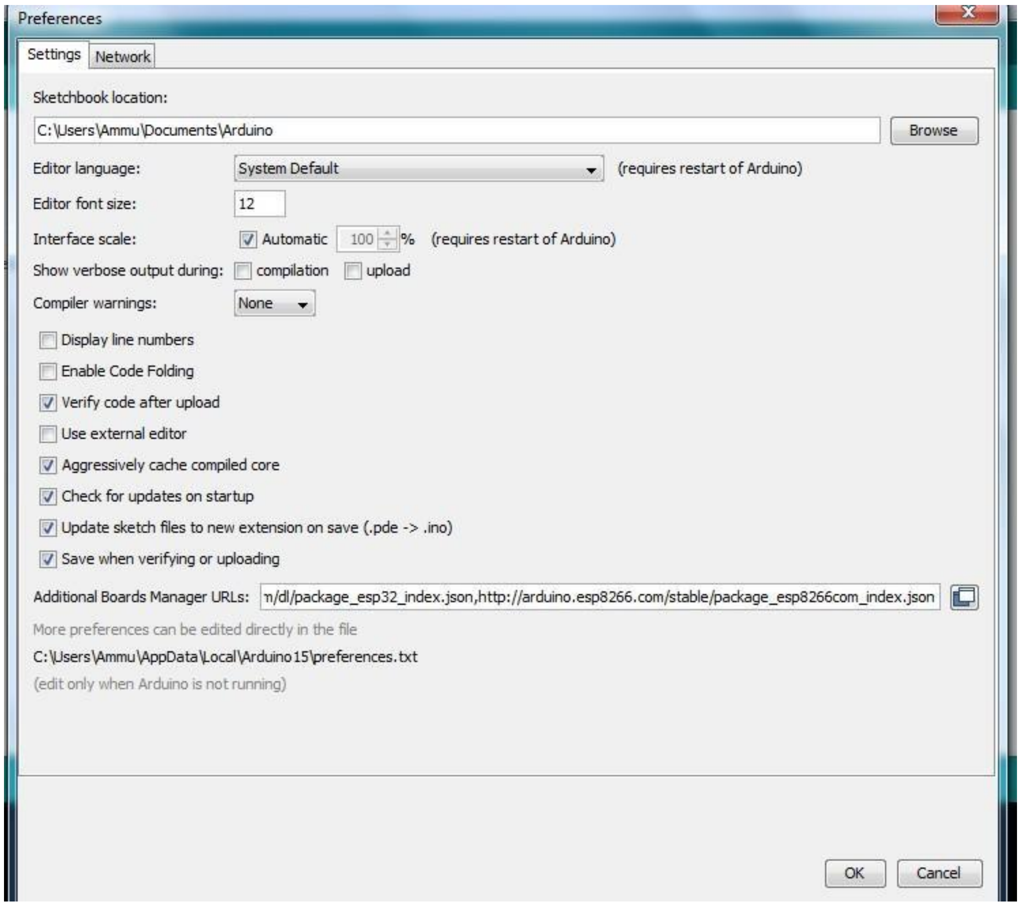
Install Arduino IDE

1. In order to upload code to the ESP8266 Node MCU and use the serial console, connect anydata-capable micro USB cable to ESP8266 IOT Board and the other side to your computer‟s USB port.
2. The new version NodeMCUv1.0 comes with the CP2102 serial chip,you can downloadand install the driver from following link
   Link: https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers
3. The NodeMCUv0.9 comes with the CH340 serial chip,you can download and install thedriver from following link
   Link: https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers
4. Download Arduino IDE from www.Arduino.cc
5. After Installing IDE then we have to Install the ESP8266 Board Package

6. Open Arduino IDE → File → Preferences → Settings



7. Paste the following link in Additional Board Manager URL :
http://arduino.esp8266.com/stable/package_esp8266com_index.json



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

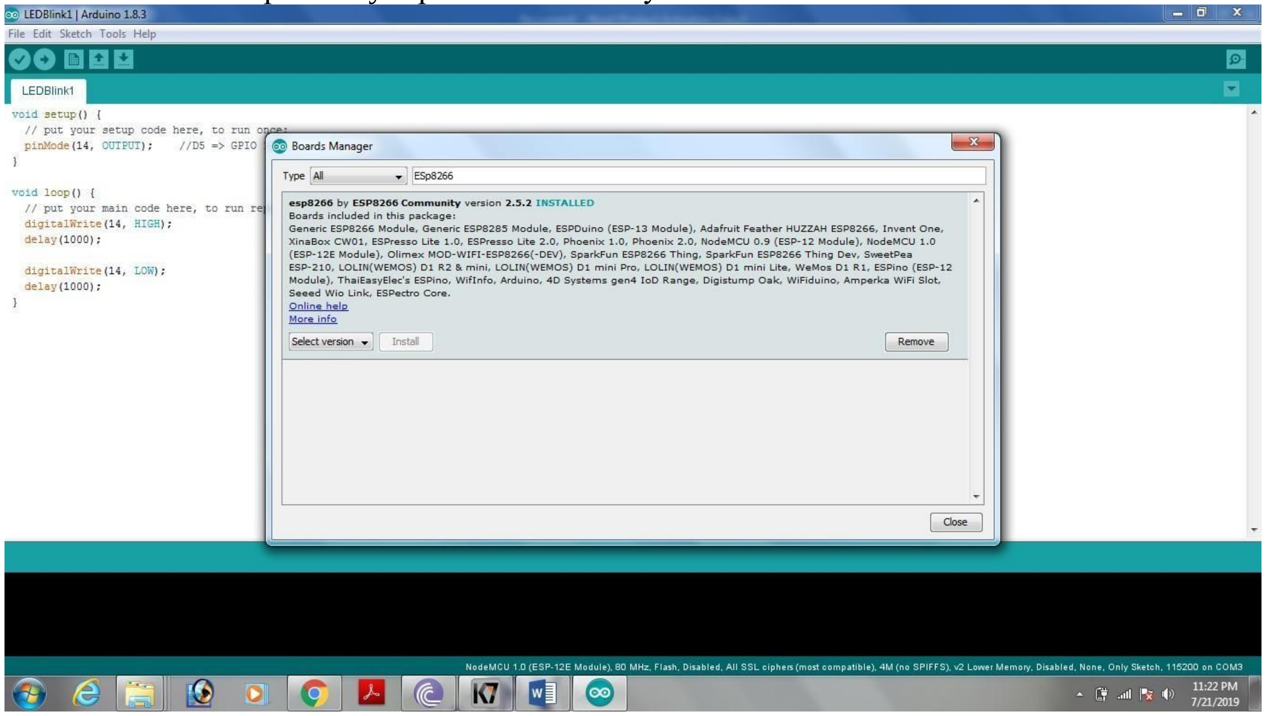
Editor language: System Default (requires restart of Arduino)
Editor font size: 12
Interface scale: [✓] Automatic 100 % (requires restart of Arduino)
Show verbose output during: [ ] compilation [ ] upload
Compiler warnings: None

- [ ] Display line numbers
- [ ] Enable Code Folding
- [✓] Verify code after upload
- [ ] Use external editor
- [✓] Aggressively cache compiled core
- [✓] Check for updates on startup
- [✓] Update sketch files to new extension on save (.pde -> .ino)
- [✓] Save when verifying or uploading

Additional Boards Manager URLs: n/dl/package_esp32_index.json,http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file
C:\Users\Ammu\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

8. Next go to Tools → Boards → Board Manager and Type Esp8266 in Search, you will see A module called “esp8266 by esp8266 Community” and Click install



```
void setup() {
  // put your setup code here, to run once:
  pinMode(14, OUTPUT);    //D5 => GPIO
}

void loop() {
  // put your main code here, to run re
  digitalWrite(14, HIGH);
  delay(1000);

  digitalWrite(14, LOW);
  delay(1000);
}
```

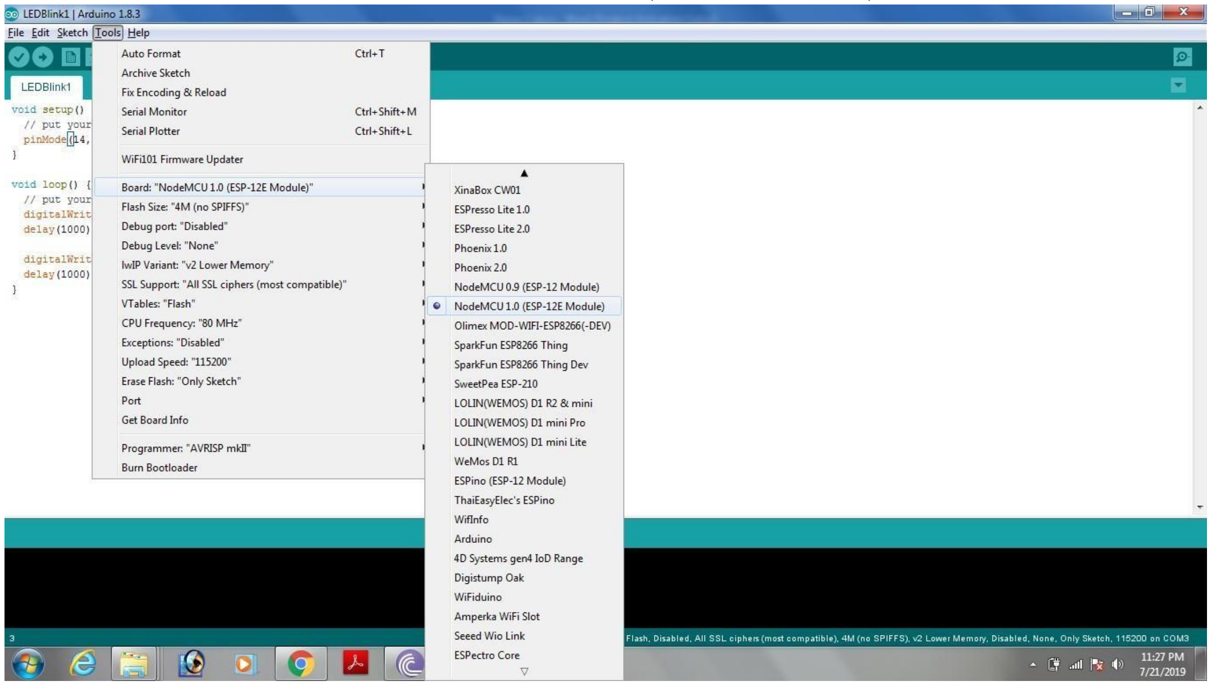
Boards Manager

Type: All — ESp8266

esp8266 by **ESP8266 Community** version **2.5.2** INSTALLED
Boards included in this package:
Generic ESP8266 Module, Generic ESP8285 Module, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, WeMos D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core.

NodeMCU 1.0 (ESP-12E Module), 80 MHz, Flash, Disabled, All SSL ciphers (most compatible), 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3

9. Go to Tools → Boards → Select Node Mcu V 1.0(ESP12E Module) and Port → COM3

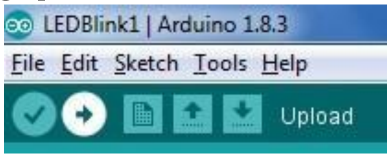


Board: "NodeMCU 1.0 (ESP-12E Module)"
Flash Size: "4M (no SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
SSL Support: "All SSL ciphers (most compatible)"
VTables: "Flash"
CPU Frequency: "80 MHz"
Exceptions: "Disabled"
Upload Speed: "115200"
Erase Flash: "Only Sketch"
Programmer: "AVRISP mkII"

Flash, Disabled, All SSL ciphers (most compatible), 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3

Arduino Script

Now write following code and upload it into the Board using upload button



```
void setup() {
  // put your setup code here, to run once:
  pinMode(14, OUTPUT);    //D5 => GPIO 14
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(14, HIGH);
  delay(1000);

  digitalWrite(14, LOW);
  delay(1000);
}
```

Ex. No.: 4 **Sketch the architecture of IoT**
Date: 23/07/24

Description:
The architecture of IOT includes 4 Layers as follows:

1. Application Layer
2. Transport Layer
3. Network Layer
4. Link Layer

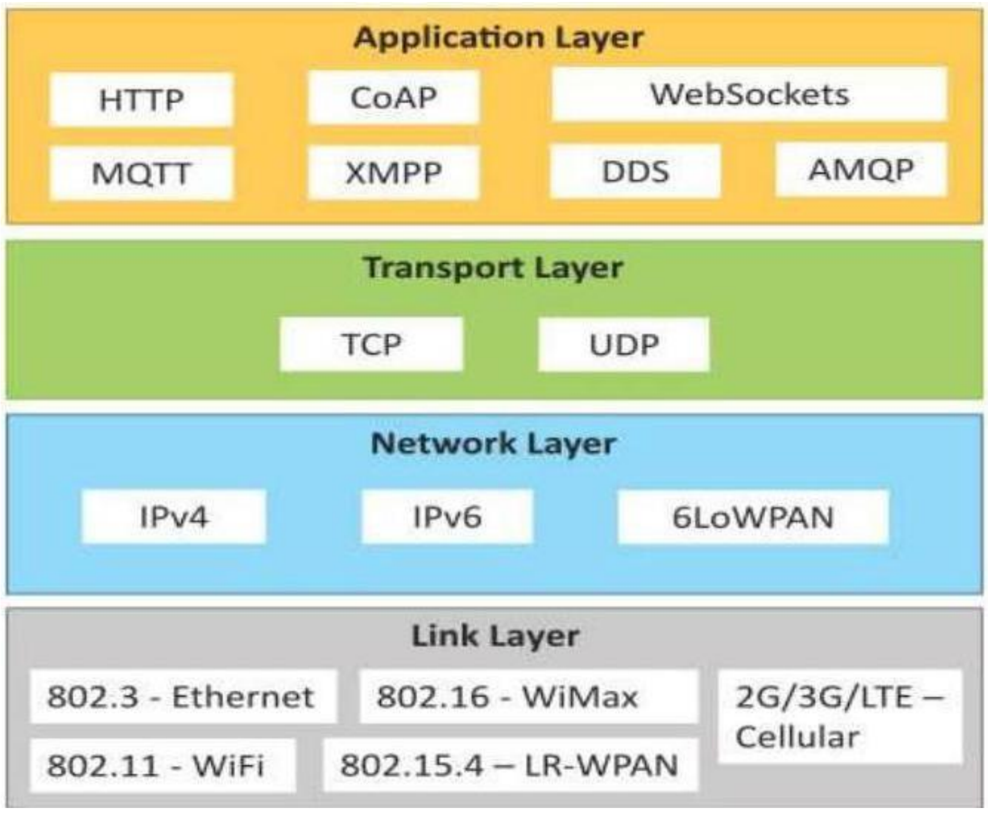


Fig. 1: Architecture of IoT

Ex. No.: 5 Interfacing Light Sensor with Node MCU Esp8266Required Hardware

Date: 13/08/24

Requirements:

1. **Node MCU ESP 8266**
2. **LDR resistor**
3. **Bread Board**
4. **Micro USB Cable**

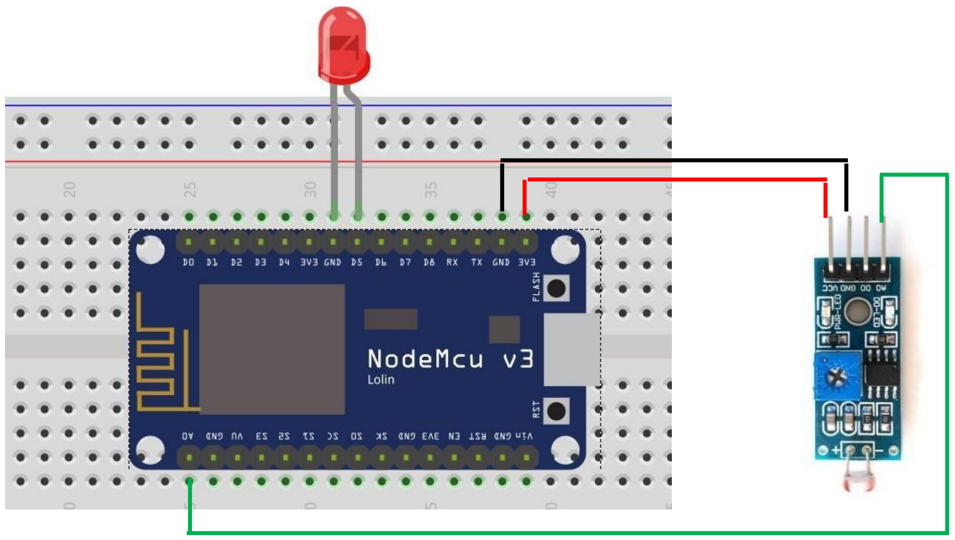
Software

1. Arduino IDE

Hardware Connection

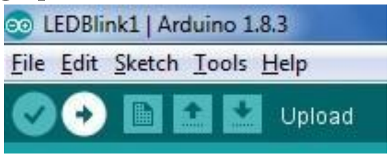
1. **Connect LDR VCC pin with 3.3v pin in Node MCU**
2. **Connect LDR GND pin with GND in Node MCU**
3. **Connect LDR A0 pin with A0 in Node MCU**
4. **Connect LED +ve pin with D5**
5. **Connect LED -ve pin with GND**

Schematic


D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
FLASH
NodeMcu v3
Lolin
RST
A0 GND VU S3 S2 S1 SC S0 SK GND 3V3 EN RST GND Vin

Arduino Script

Now write following code and upload it into the Board using upload button



```
void setup() {
// initialize serial communication at 9600 bits per second:
Serial.begin(9600);
pinMode(D5, OUTPUT);
}

void loop() {
// read the input on analog pin 0: int sensorValue = analogRead(A0);

// print out the value you read:
Serial.println(sensorValue);
If(sensorValue < 200)
{
        digitalWrite(D5, HIGH);
 }
else
{
digitalWrite(D5, LOW);
}
}
```

Ex . No.:6 Interfacing Soil Moisture Sensor with Node MCU Esp8266

Date: 21/08/24

Required Hardware

1. **Node MCU ESP 8266**
2. **Soil Moisture Sensor**
3. **Bread Board**
4. **Micro USB Cable**
5. **LED**

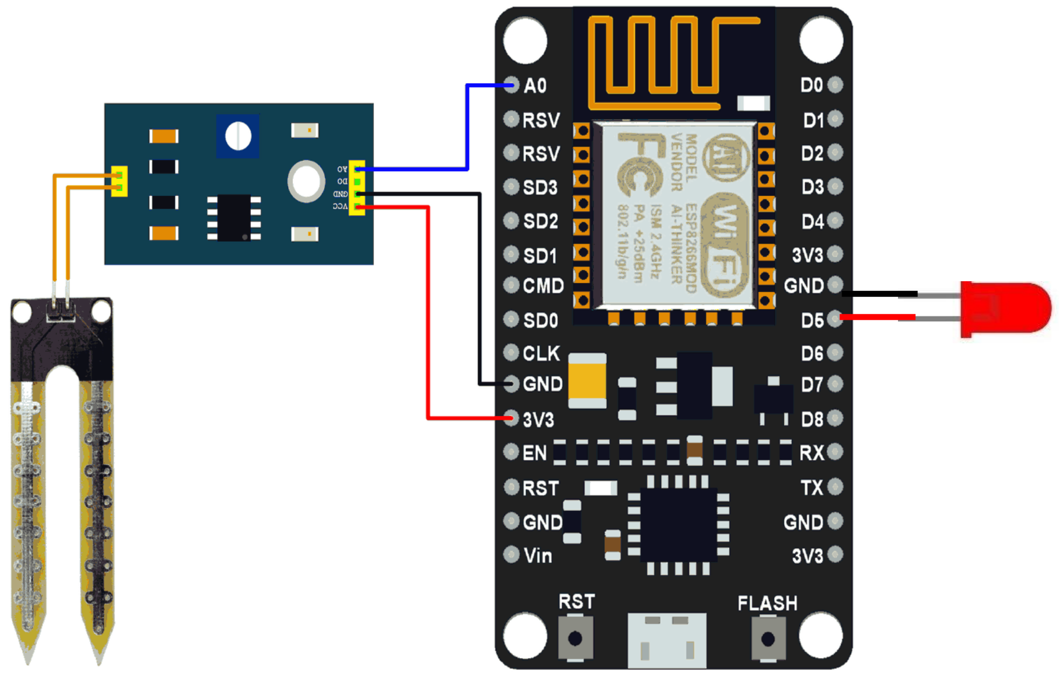
Software

1. **Arduino IDE**

Hardware Connection

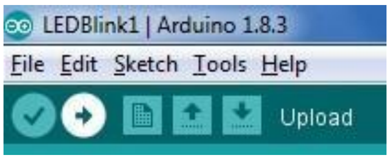
1. **Connect Soil Moisture Sensor VCC pin with 3.3v pin in Node MCU**
2. **Connect Soil Moisture Sensor GND pin with GND in Node MCU**
3. **Connect Soil Moisture Sensor A0 pin with A0 in Node MCU**
4. **Connect LED +ve pin with D5**
5. **Con nect LED -ve pin with GND**

Schematic



Arduino Script

Now write following code and upload it into the Board using upload button



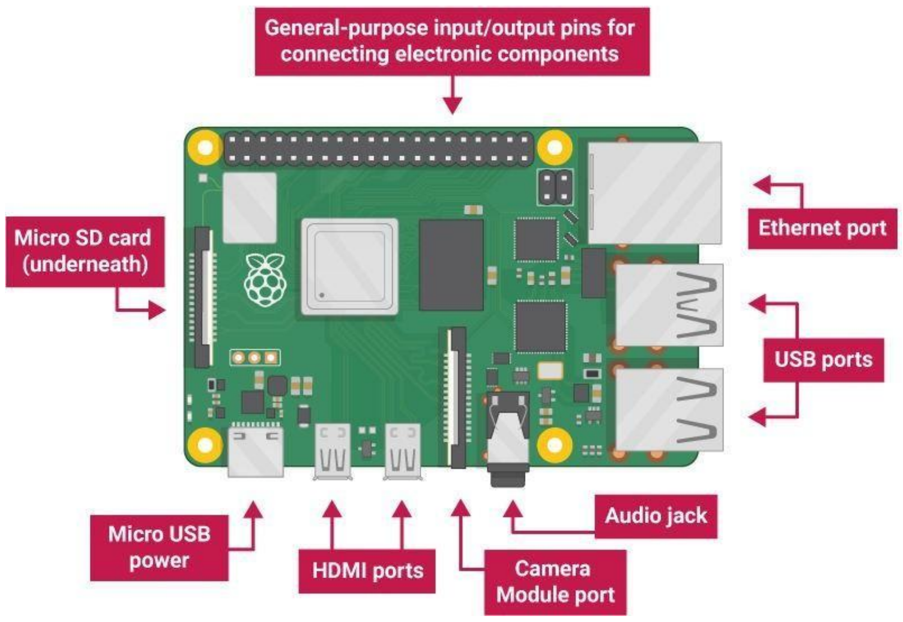
```
#include <ESP8266WiFi.h>

const byte relayOnState = LOW;
const byte relayOffState = HIGH;

int SoilMoist = A0; // connect ir sensor to arduino pin 9
int LED = D5; // conect Led to arduino pin 6

void setup()
{

  pinMode (SoilMoist, INPUT); // sensor pin INPUT
  pinMode (LED, OUTPUT); // Led pin OUTPUT
  Serial.begin(9600);
}


void loop()
{

   float moisture_percentage;
  moisture_percentage = ( 100.00 - ( (analogRead(SoilMoist)/1023.00) * 100.00 ) );
   Serial.println(moisture_percentage);
  if(moisture_percentage <= 10)
  {
    digitalWrite(LED, relayOnState); // LED LOW

  }
  else
  {
    digitalWrite(LED, relayOffState); // LED High
  }

}

}
```

Ex. No. : 7 **Explain working of Raspberry Pi**
Date: 27/08/24

RaspberryPi

You are going to take a first look at RaspberryPi! You should have a RaspberryPi computer in front of you for this. The computer shouldn‟t be connected to anything yet.

- LookatyourRaspberryPi.Can youfindallthethingslabelledonthediagram?


General-purpose input/output pins for connecting electronic components
Micro SD card (underneath)
Ethernet port
USB ports
Audio jack
Micro USB power
HDMI ports
Camera Module port

- USB ports— these are used to connect a mouse and keyboard. You can also connect other components, such as a USB drive.
- SD card slot —you canslot the SD card in here. This is where theoperating system software and your files are stored.
- Ethernetport—thisisusedtoconnectRaspberryPitoanetworkwithacable. Raspberry Pi can also connect to a network via wireless LAN.
- Audiojack—youcan connectheadphonesorspeakers here.

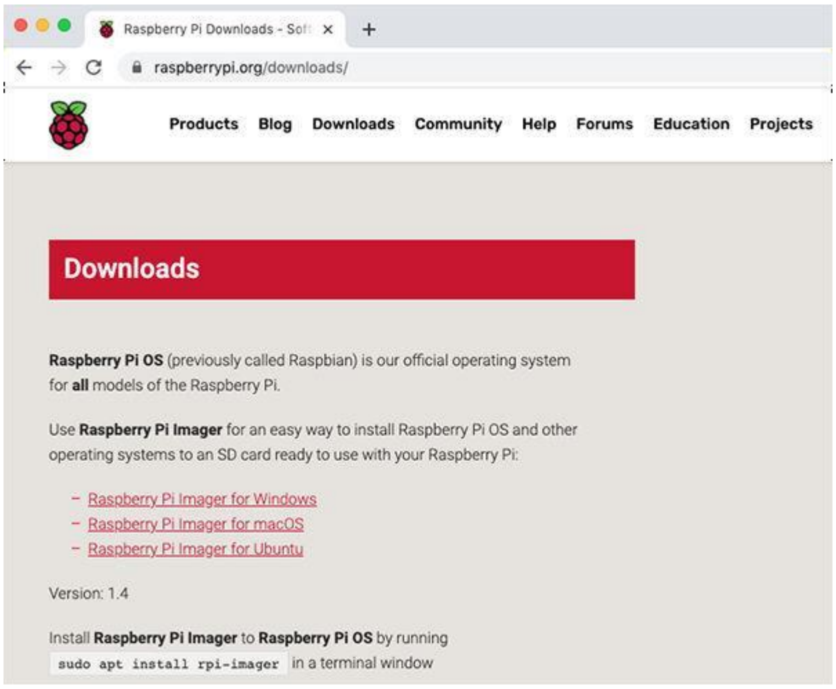
- HDMI port— this is where you connect the monitor (or projector) that you are using to displaytheoutput from theRaspberryPi. If yourmonitorhas speakers, you can also use them to hear sound.
- MicroUSBpowerconnector—thisiswhere youconnectapower supply. Youshould always do this last, after you have connected all your other components.
- GPIO ports— these allow you to connect electronic components such as LEDs and buttons to Raspberry Pi.

SetupyourSD card

If youhaveanSDcardthatdoesn‟thavetheRaspberryPiOSoperatingsystemon it yet,orif you want to reset your Raspberry Pi, you can easily install Raspberry Pi OS yourself. To do so, youneeda computer thathasanSDcardport —most laptopanddesktopcomputershave one. TheRaspberryPiOS operatingsystem viathe RaspberryPiImager

Usingthe RaspberryPi Imager is the easiest wayto install RaspberryPi OS on your SD card. Note:More advanced users looking to install a particular operating system should use this guide to installing operating system images.
DownloadandlaunchtheRaspberryPi Imager

- VisittheRaspberryPidownloads page



Downloads

Raspberry Pi OS (previously called Raspbian) is our official operating system for **all** models of the Raspberry Pi.

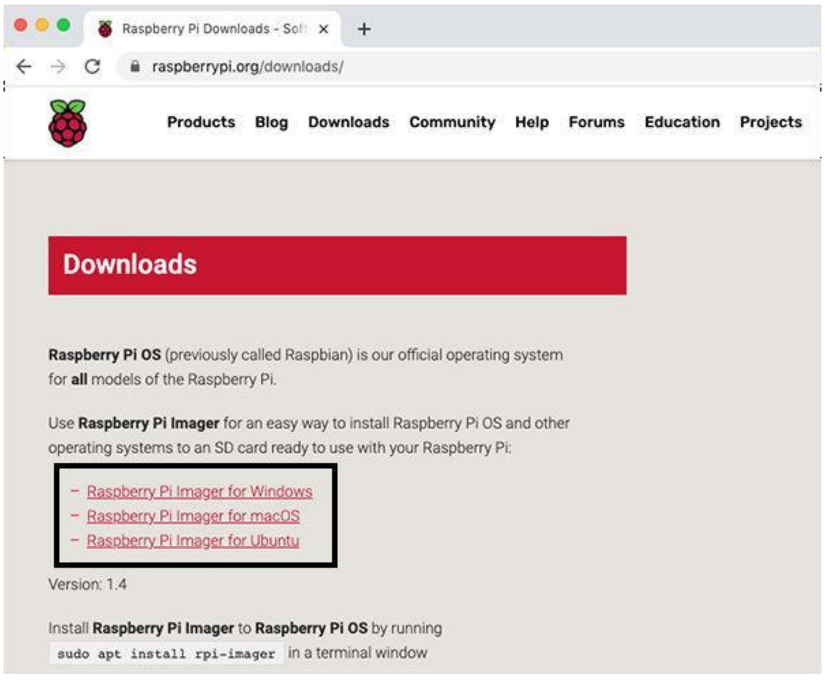
Use **Raspberry Pi Imager** for an easy way to install Raspberry Pi OS and other operating systems to an SD card ready to use with your Raspberry Pi:

- Raspberry Pi Imager for Windows
- Raspberry Pi Imager for macOS
- Raspberry Pi Imager for Ubuntu

Version: 1.4

Install **Raspberry Pi Imager** to **Raspberry Pi OS** by running `sudo apt install rpi-imager` in a terminal window

- ClickonthelinkfortheRaspberryPiImagerthatmatchesyouroperating system


Raspberry Pi Downloads - Soft

raspberrypi.org/downloads/

Downloads

Raspberry Pi OS (previously called Raspbian) is our official operating system for **all** models of the Raspberry Pi.

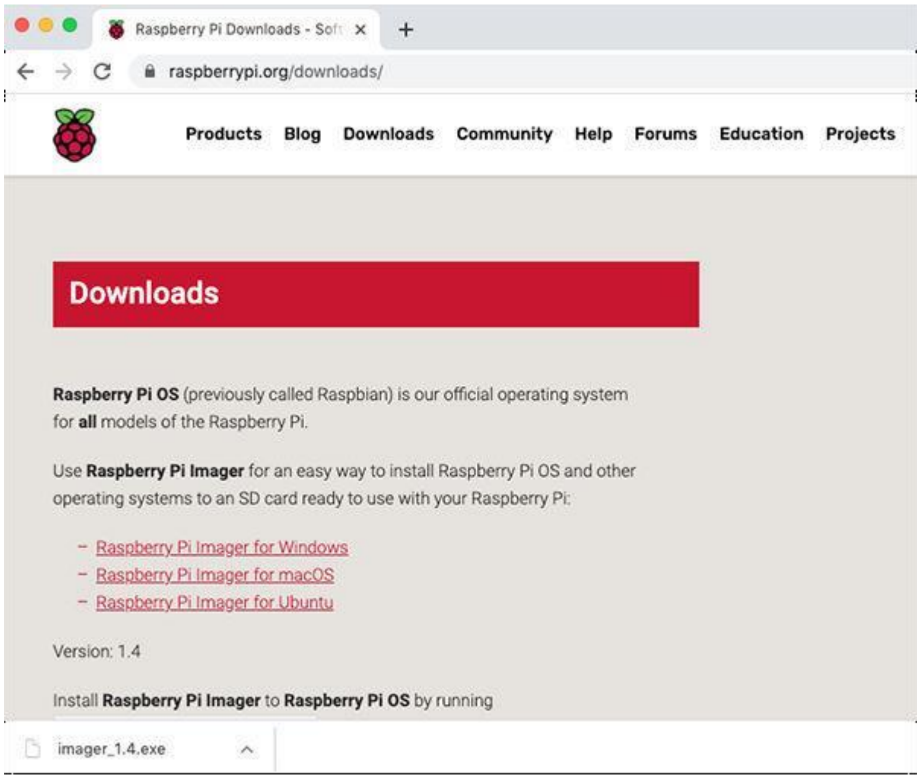
Use **Raspberry Pi Imager** for an easy way to install Raspberry Pi OS and other operating systems to an SD card ready to use with your Raspberry Pi:

- Raspberry Pi Imager for Windows
- Raspberry Pi Imager for macOS
- Raspberry Pi Imager for Ubuntu

Version: 1.4

Install **Raspberry Pi Imager** to **Raspberry Pi OS** by running `sudo apt install rpi-imager` in a terminal window

- Whenthedownloadfinishes,clickittolaunchthe installer


Raspberry Pi Downloads - Soft

raspberrypi.org/downloads/

Downloads

Raspberry Pi OS (previously called Raspbian) is our official operating system for **all** models of the Raspberry Pi.

Use **Raspberry Pi Imager** for an easy way to install Raspberry Pi OS and other operating systems to an SD card ready to use with your Raspberry Pi:

- Raspberry Pi Imager for Windows
- Raspberry Pi Imager for macOS
- Raspberry Pi Imager for Ubuntu

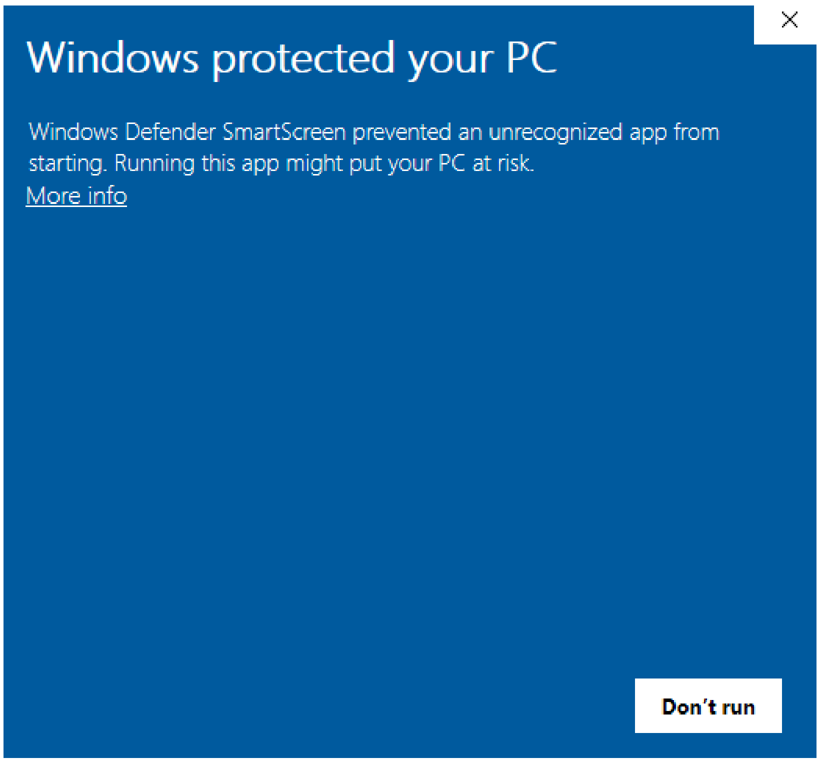
Version: 1.4

Install **Raspberry Pi Imager** to **Raspberry Pi OS** by running

imager_1.4.exe

UsingtheRaspberryPiImager

Anything that‟s stored on the SD card will be overwritten during formatting. If your SD card currently has any files on it, e.g. from an older version of Raspberry Pi OS, you may wish to back up these files first to prevent you from permanently losing them.
When you launch the installer, your operating system may try to block you from running it. For example, on Windows I receive the following message:

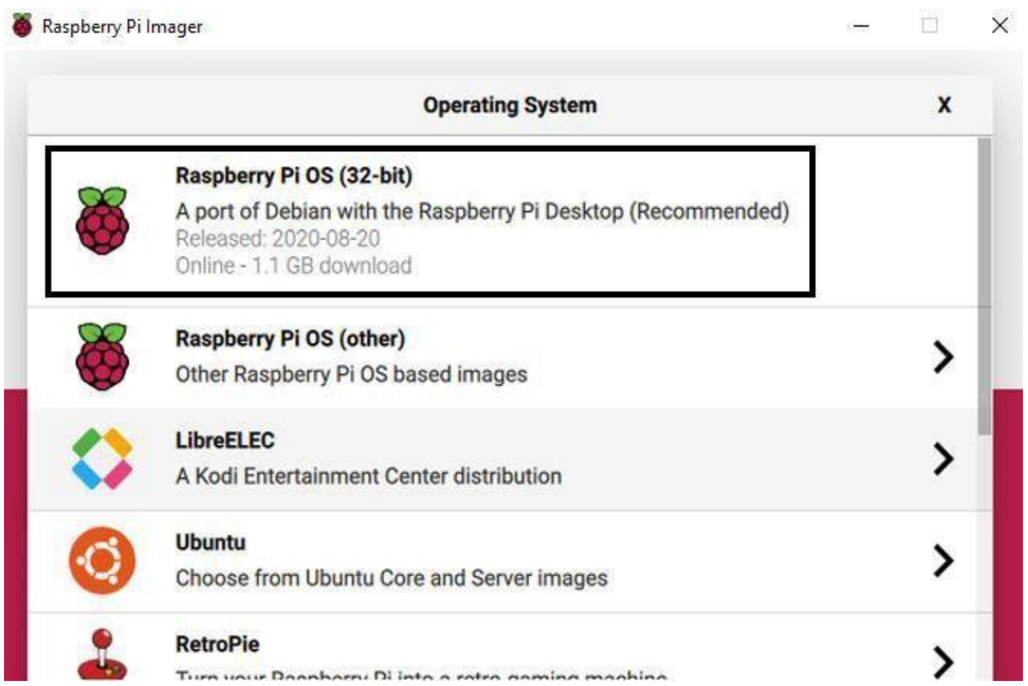


Windows protected your PC

Windows Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.
More info

- If thispops up,click onMoreinfoand thenRunanyway
- FollowtheinstructionstoinstallandruntheRaspberryPiImager
- InsertyourSDcardintothecomputerorlaptopSDcardslot
- In the Raspberry Pi Imager, select the OS that you want to install and the SD card you would like to install it on

Note:You will need to be connected to the internet the first time for the the Raspberry Pi Imager to download the OS that you choose. That OS will then be stored for future offline use. Being online for later uses means that the Raspberry Pi imager will always give you the latest version.



Raspberry Pi Imager

Operating System

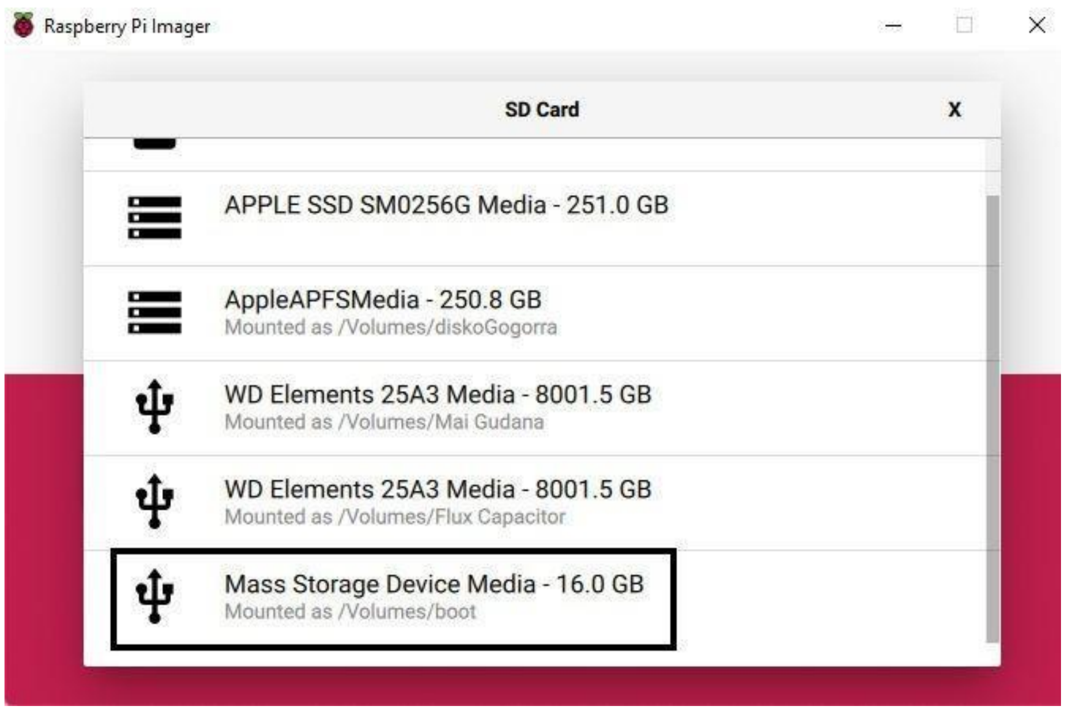
Raspberry Pi OS (32-bit)
A port of Debian with the Raspberry Pi Desktop (Recommended)
Released: 2020-08-20
Online - 1.1 GB download

Raspberry Pi OS (other)
Other Raspberry Pi OS based images

LibreELEC
A Kodi Entertainment Center distribution

Ubuntu
Choose from Ubuntu Core and Server images

RetroPie



Raspberry Pi Imager

SD Card

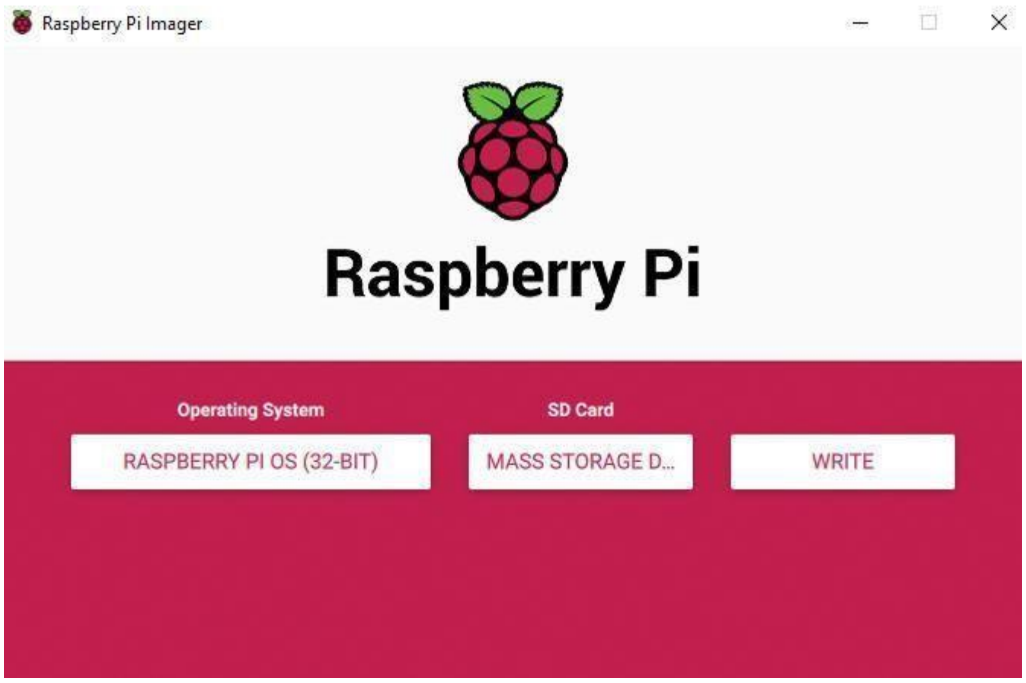
APPLE SSD SM0256G Media - 251.0 GB

AppleAPFSMedia - 250.8 GB
Mounted as /Volumes/diskoGogorra

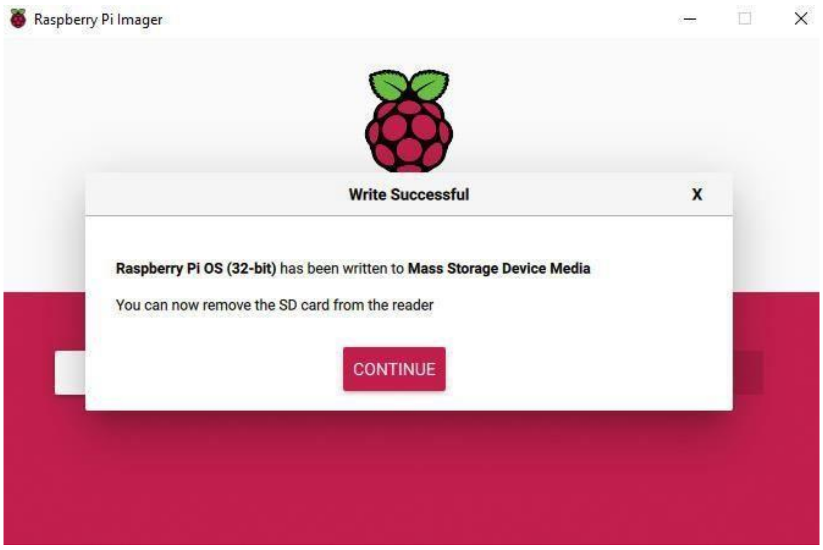
WD Elements 25A3 Media - 8001.5 GB
Mounted as /Volumes/Mai Gudana

WD Elements 25A3 Media - 8001.5 GB
Mounted as /Volumes/Flux Capacitor

Mass Storage Device Media - 16.0 GB
Mounted as /Volumes/boot



- Thensimplyclickthe WRITEbutton
- Waitforthe RaspberryPiImagertofinish writing
- Once you getthefollowingmessage,youcaneject yourSDcard



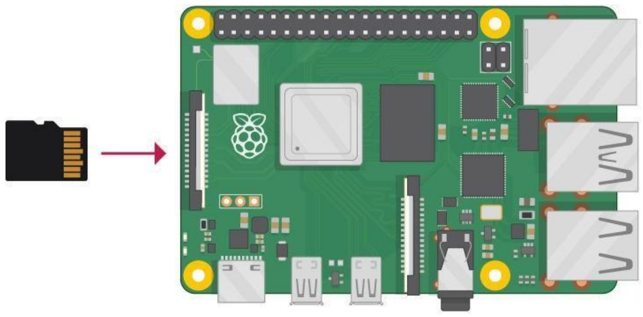
Raspberry Pi OS (32-bit) has been written to Mass Storage Device Media

You can now remove the SD card from the reader

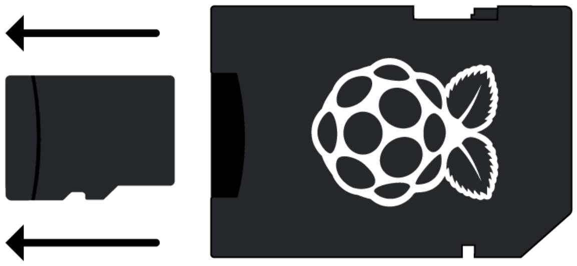
ConnectyourRaspberryPi

Let‟sconnectupyourRaspberryPiand getitrunning.

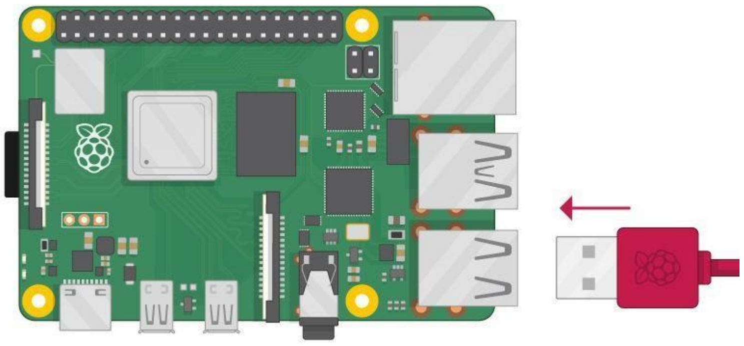
- Check the slot on the underside of your Raspberry Pi to see whether an SD card is inside. If no SD card is there, then insert an SD card with Raspbian installed (via NOOBS).



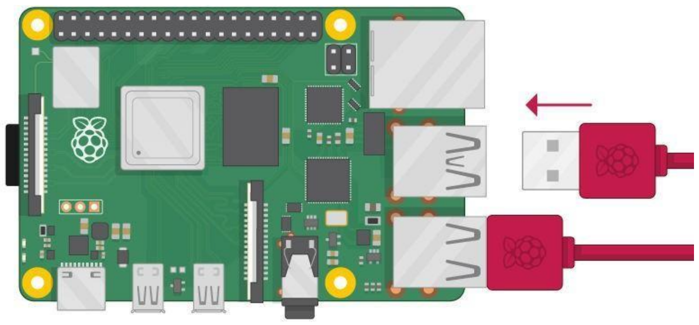
Note:ManymicroSDcards comeinsidealargeradapter — you can slidethesmallercardout using the lip at the bottom.



- Find the USB connector end of your mouse‟s cable, and connect the mouse to a USB port on your Raspberry Pi (it doesn‟t matter which port you use).



Connectthekeyboardinthesameway.

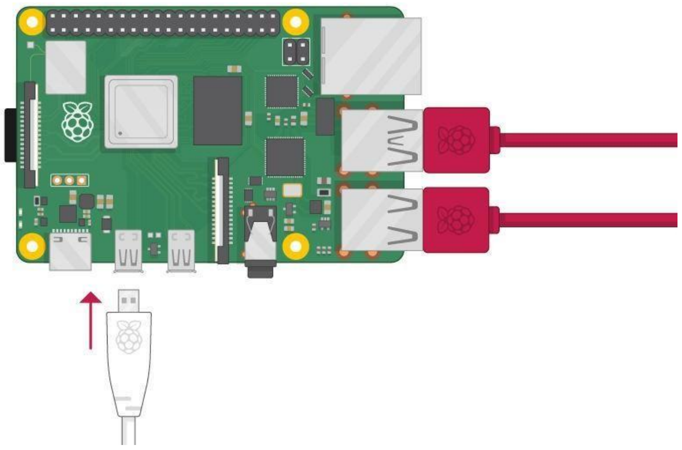


- Makesureyourscreenis pluggedintoawallsocketandswitchedon.
- Lookatthe HDMIport(s)onyour RaspberryPi— noticethattheyhaveaflatsideon top.

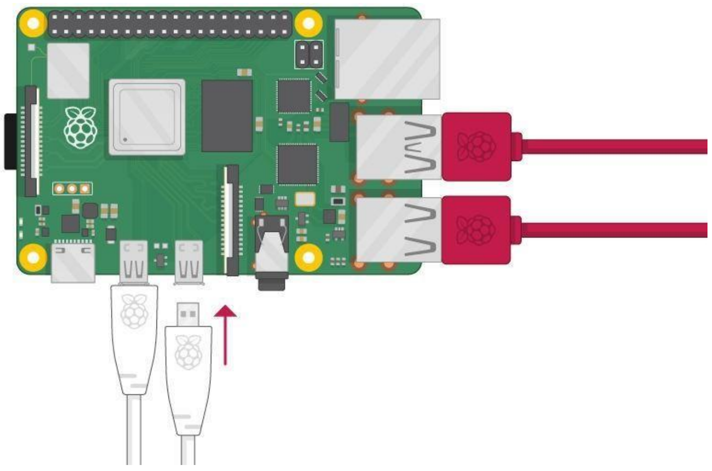
- UseacabletoconnectthescreentotheRaspberry Pi‟sHDMI port—useanadapterif necessary.

RaspberryPi 4

Connect yourscreentothefirstofRaspberryPi4‟sHDMIports,labelledHDMI0.

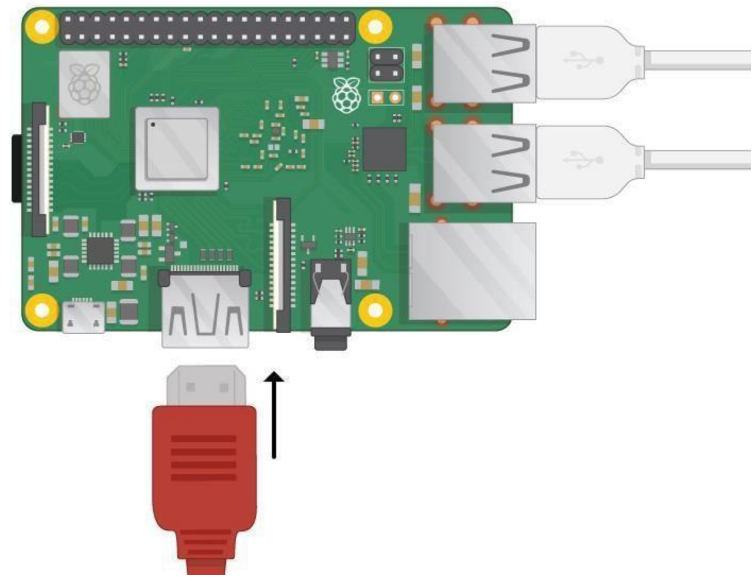


Youcouldconnect anoptionalsecondscreeninthesameway.



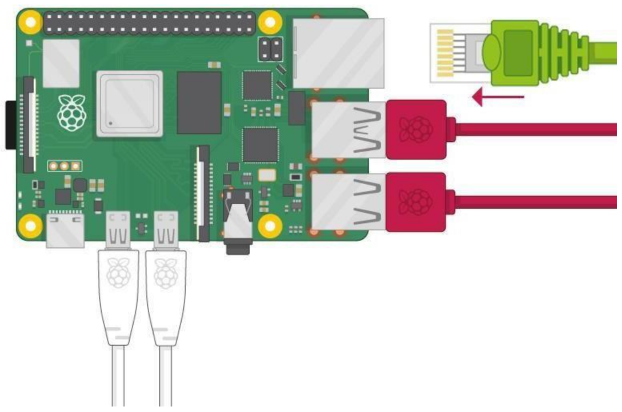
RaspberryPi1,2,3

Connect yourscreentothesingleHDMIport.

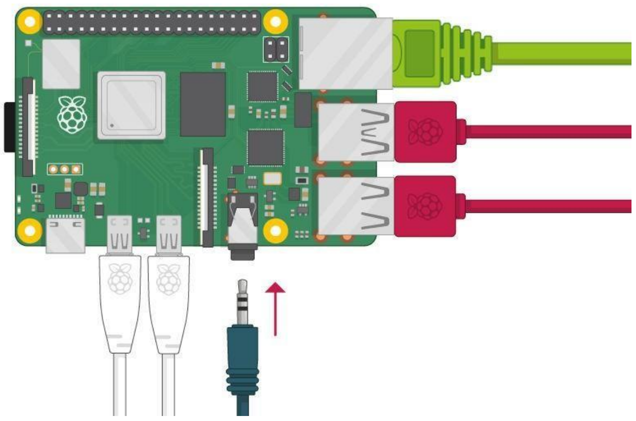


Note:nothingwilldisplayonthe screen,becausetheRaspberryPi isnot runningyet.

- If you want to connect the Pi to the internet via Ethernet, use an Ethernet cable to connect the Ethernet port on the RaspberryPi to an Ethernet socket on the wall or on your internet router. You don‟t need to do this if you want to use wireless connectivity, or if you don‟t want to connect to the internet.



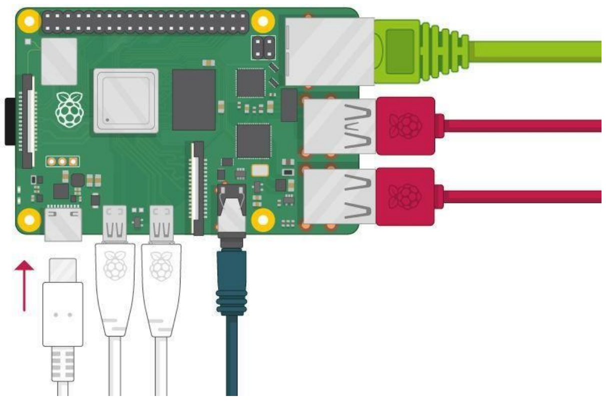
- If yourscreenhasspeakers, yourRaspberryPicanplaysoundthroughthese.Or youcould connect headphones or speakers to the audio port.



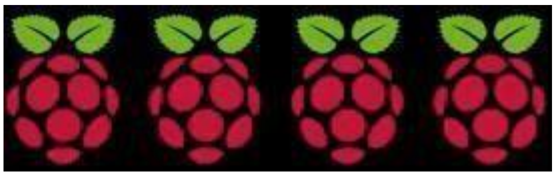
StartupyourRaspberryPi

Your Raspberry Pi doesn‟t have a power switch. As soon as you connect it to a power outlet, it will turn on.

- PlugthepowersupplyintoasocketandconnectittoyourRaspberryPi‟spowerport.



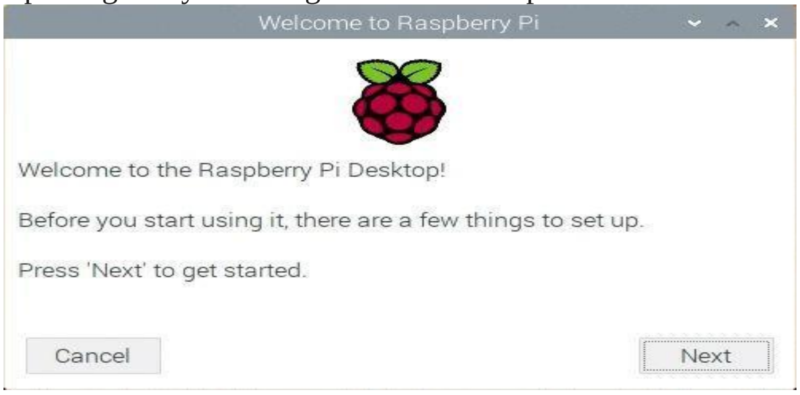
You should see a red LED light up on the Raspberry Pi, which indicates that Raspberry Pi is connected to power. As it starts up (this is also calledbooting), you will see raspberries appear in the top left-hand corner of your screen.



AfterafewsecondstheRaspberryPiOSdesktopwill appear.

Finishthe setup

When you start your RaspberryPi for the first time, the Welcome to RaspberryPiapplication will pop up and guide you through the initial setup.



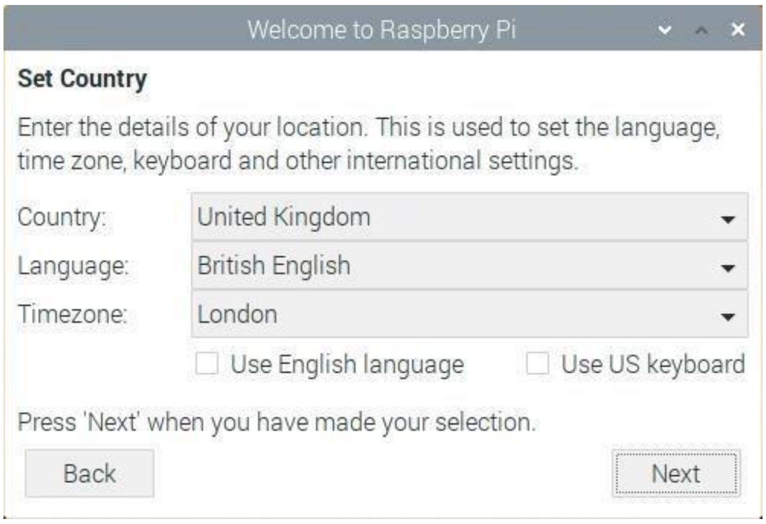
Welcome to Raspberry Pi

Welcome to the Raspberry Pi Desktop!

Before you start using it, there are a few things to set up.

Press 'Next' to get started.

- ClickNexttostart thesetup.
- Set yourCountry,Language,andTimezone,thenclickNextagain.



Welcome to Raspberry Pi

Set Country

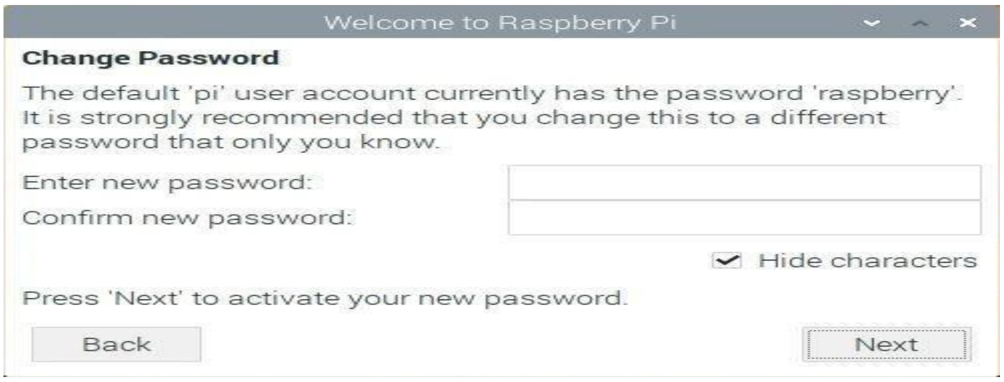
Enter the details of your location. This is used to set the language, time zone, keyboard and other international settings.

Country: United Kingdom
Language: British English
Timezone: London

☐ Use English language ☐ Use US keyboard

Press 'Next' when you have made your selection.

○ Enteranewpassword for yourRaspberryPiandclickNext.


Welcome to Raspberry Pi

Change Password

The default 'pi' user account currently has the password 'raspberry'. It is strongly recommended that you change this to a different password that only you know.

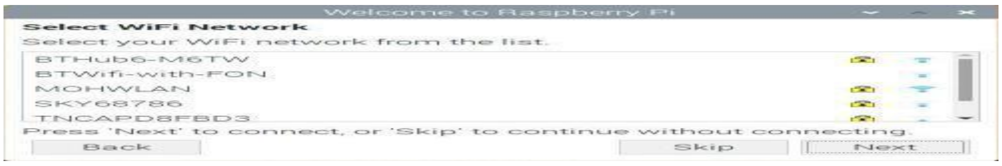
Enter new password:

Confirm new password:

✓ Hide characters

Press 'Next' to activate your new password.

- ConnecttoyourWiFinetworkbyselectingitsname,enteringthepassword,and clicking Next.


Welcome to Raspberry Pi

Select WiFi Network

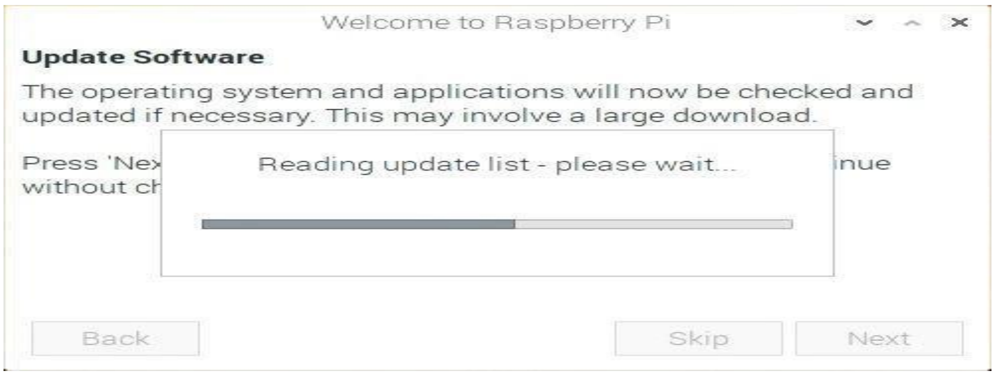
Select your WiFi network from the list.

BTHub6-M6TW
BTWifi-with-FON
MOHWLAN
SKY68786
TNCAPD8FBD3

Press 'Next' to connect, or 'Skip' to continue without connecting.

Note:ifyourRaspberryPimodeldoesn‟thavewirelessconnectivity,youwon‟tseethis screen.

- ClickNextlet thewizardcheck forupdatesto Raspbianand install them (this might take a little while).


Welcome to Raspberry Pi

Update Software

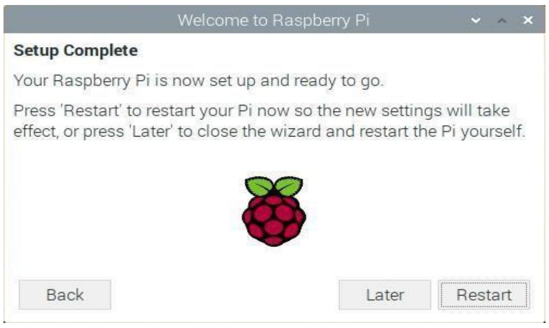
The operating system and applications will now be checked and updated if necessary. This may involve a large download.

Press 'Nex inue without ch

Reading update list - please wait...

- ClickDoneorRebootto finishthe setup.

Note:youwillonlyneedto rebootifthat‟snecessarytocompletean update.


Welcome to Raspberry Pi

Setup Complete

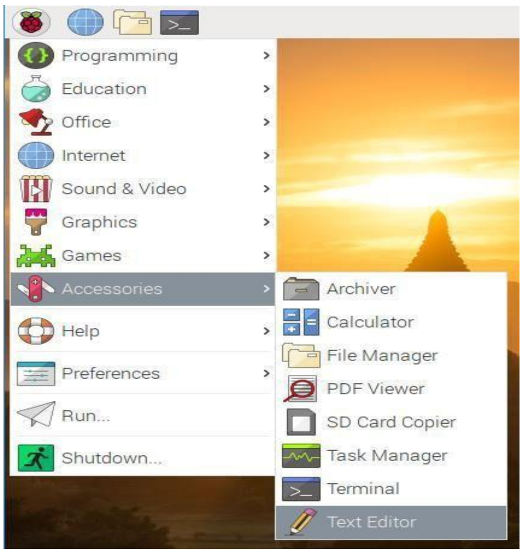
Your Raspberry Pi is now set up and ready to go.

Press 'Restart' to restart your Pi now so the new settings will take effect, or press 'Later' to close the wizard and restart the Pi yourself.

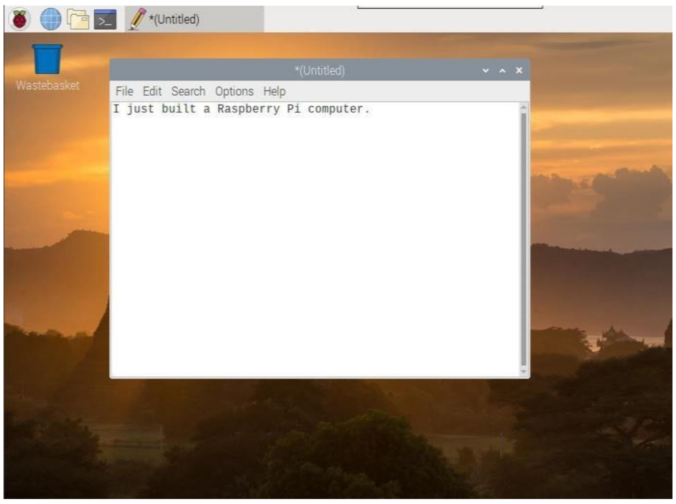
AtourofRaspberryPi

Nowit‟stimetotakeatourofyourRaspberryPi.

- Do you see the raspberry symbol in the top left-hand corner? That‟s where you access the menu: click on it to find lots of applications.
- ClickonAccessories,andthenclickon TextEditor.

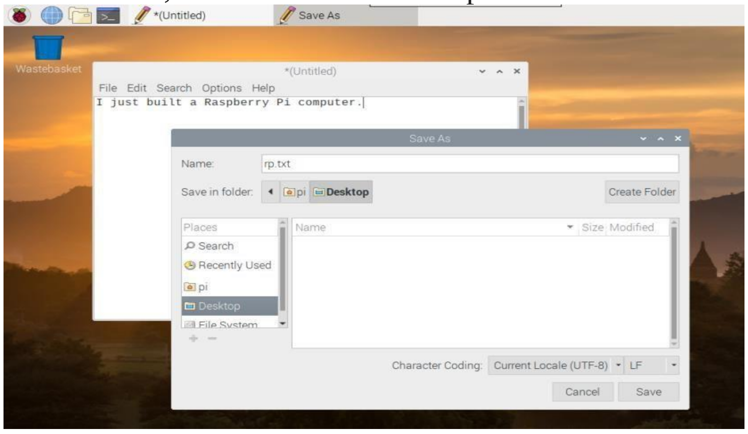


- TypeIjustbuilt aRaspberryPi computerin thewindow thatappears.



```
I just built a Raspberry Pi computer.
```

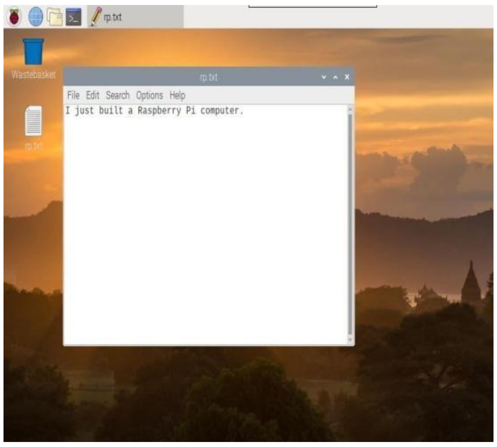
- ClickonFile,thenchooseSave,andthen clickonDesktop andsavethefileas .



```
I just built a Raspberry Pi computer.
```

Name: rp.txt

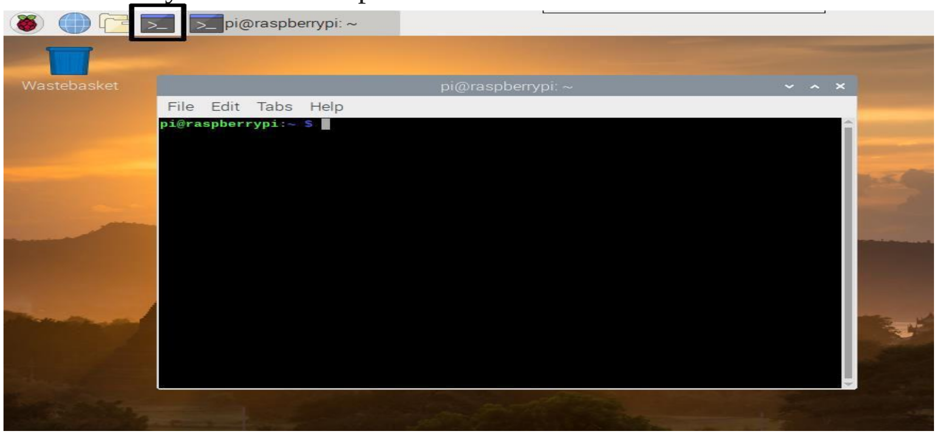
- Youshouldseeaniconnamed rp.txt appearonthedesktop.



```
I just built a Raspberry Pi computer.
```

Yourfilehasbeensaved to yourRaspberryPi‟sSD card.

- Closethetexteditor byclickingtheXinthetopright-handcorneroftheй window.
- Returntothemenu,clickonShutdown,andthen clickon Reboot.
- WhenRaspberryPihas rebooted,your textfileshouldstill be thereon thedesktop.
- Raspberry Pi runs a version of an operating system called Linux (Windows andmacOS are other operatingsystems). This operatingsystem allows you to make things happen by typing in commands instead of clicking on menu options. To try this out, click on the Terminal symbol at the top of the screen:



```
pi@raspberrypi:~ $
```

Inthe windowthatappears,type:

```
ls
```

andthenpressEnteronthe keyboard.

Youcannowseealistofthefilesandfoldersinyour directory.

- Nowtypethiscommandto changedirectorytothe Desktop:

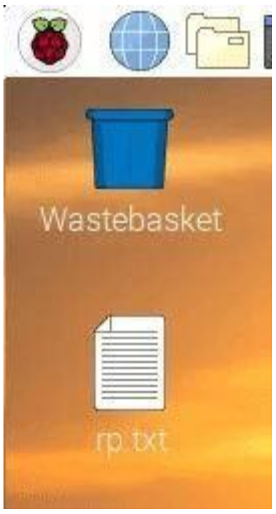
```
cdDesktop
```

YouhavetopresstheEnterkeyaftereverycommand. Then type:

```
ls
```

Canyouseethetextfileyou created?

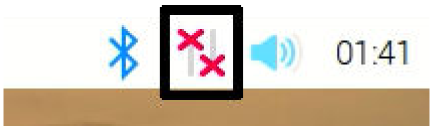
- Closetheterminal window byclickingon theX.
- Now drag to the Wastebasket on the desktop so the Raspberry Pi will be tidyfor the next person using it.



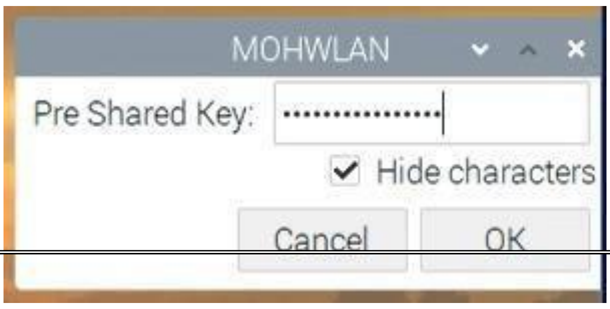
Browsingtheweb

Youmightwanttoconnect yourRaspberryPi totheinternet. If youdidn‟tpluginanethernet cable or connect to a WiFi network during the setup, then you can connect now.

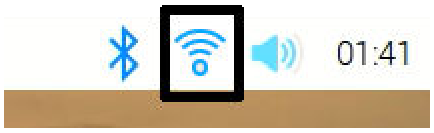
- Click the icon with red crosses in the top right-hand corner of the screen, and select your network from the drop-down menu. You may need to ask an adult which network you should choose.



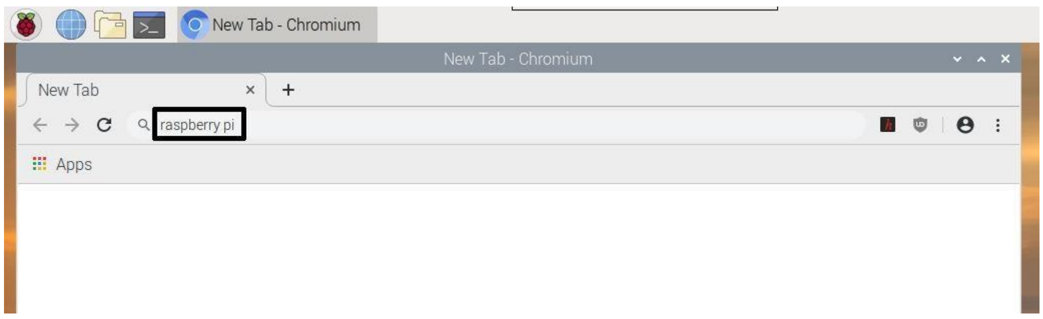
- Typeinthepasswordforyourwirelessnetwork,oraskanadulttotypeitforyou,then click OK.
- WhenyourPiisconnectedtotheinternet, youwillseeawirelessLANsymbolinsteadof the red



crosses.

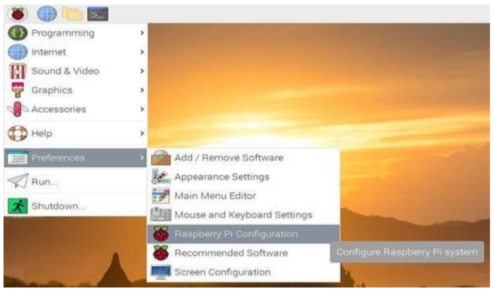


- Clickthewebbrowsericonandsearchfor .



Configuring your Raspberry Pi

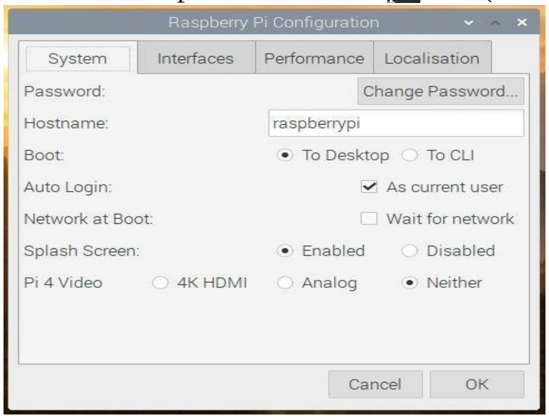
You can control most of your Raspberry Pi‟s settings, such as the password, through the Raspberry Pi Configuration application found in Preferences on the menu.



- System

In this tab you can change basic system settings of your Raspberry Pi.

- Password— set the password of the pi user (it is a good idea to change the password


Raspberry Pi Configuration

System | Interfaces | Performance | Localisation

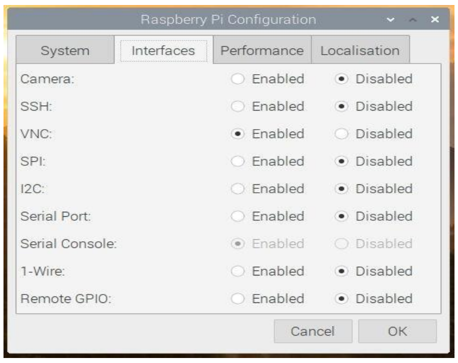
Password:		Change Password...	
Hostname:		raspberrypi	
Boot:		[✓ To Desktop]	To CLI
Auto Login:			[✓ As current user]
Network at Boot:			Wait for network
Splash Screen:		[✓ Enabled]	Disabled
Pi 4 Video	4K HDMI	Analog	[✓ Neither]

Cancel OK

from the factory default „raspberry")

- Boot—select to show the Desktop or CLI (command line interface)when your Raspberry Pi starts
- Auto Login—enabling this option will make the Raspberry Pi automatically login whenever it starts
- Network at Boot— selecting this option will cause your Raspberry Pi to wait until a network connection is available before starting
- Splash Screen— choose whether or not to show the splash (start up) screen when your Raspberry Pi boots
- Interfaces

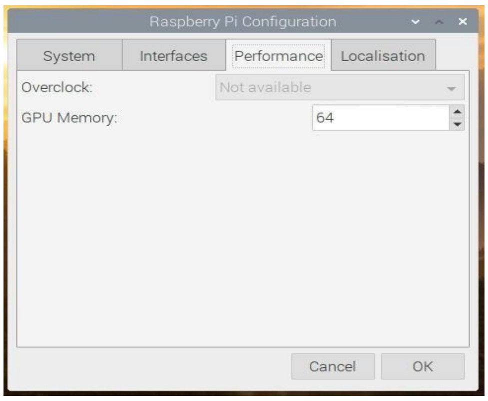
You can link devices and components to your Raspberry Pi using a lot of different types of connections. The Interfaces tab is where you turn these different connections on or off, so that your Raspberry Pi recognises that you"ve linked something to it via a particular type of connection.



Camera:	Enabled [✓ Disabled]
SSH:	Enabled [✓ Disabled]
VNC:	[✓ Enabled] Disabled
SPI:	Enabled [✓ Disabled]
I2C:	Enabled [✓ Disabled]
Serial Port:	Enabled [✓ Disabled]
Serial Console:	[✓ Enabled] Disabled
1-Wire:	Enabled [✓ Disabled]
Remote GPIO:	Enabled [✓ Disabled]

- Camera—enable the RaspberryPi Camera Module
- SSH— allow remote access to your Raspberry Pi from another computer using SSH
- VNC— allow remote access to the Raspberry Pi Desktop from another computer using VNC
- SPI—enable the SPIGPIOpins
- I2C—enabletheI2CGPIOpins
- Serial—enable the Serial(Rx,Tx)GPIOpins
- 1-Wire—enablethe1-WireGPIOpin
- RemoteGPIO—allow access to your Raspberry Pi"s GPIOpins from another computer
- Performance

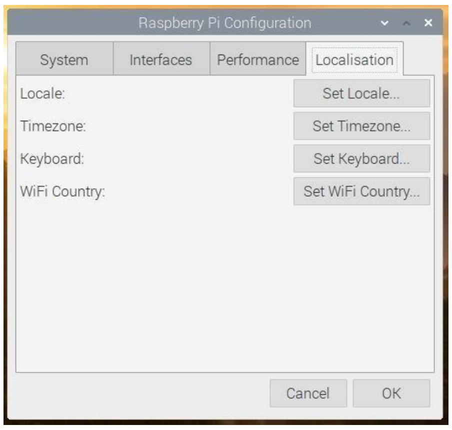
If you need to do so for a particular project you want to work on, you can change the performance settings of your Raspberry Pi in this tab.
Warning: Changing your Raspberry Pi"s performance settings may result in it behaving erratically or not working.



Overclock: Not available
GPU Memory: 64

- Overclock—changetheCPUspeedandvoltagetoincrease performance
- **GPUMemory**—changetheallocationofmemorygiventothe GPU

- Localisation



This tab allows you to change your Raspberry Pi settings to be specific to a country or location.

- Locale—setthelanguage,country,andcharactersetusedbyyourRaspberryPi
- Timezone— set thetime zone
- Keyboard—changeyourkeyboardlayout
- WiFi Country— settheWiFi country code

Ex. No. 8: Connect Rasberry Pi with your existing system components

ExportingDisplayOnToOtherSystems

Date: 28/08/24

Making use of available laptop/desktop displays as a display for the device using SSH client & X11 display server.

HowDoesitWork?

To connect a Raspberry Pi to a laptop display, you can simply use an Ethernet cable. The Raspberry Pi"s desktop GUI (Graphical User Interface) can be viewed through the laptop display using a 100 Mbps Ethernet connection between the two. There are many software programs available that can establish a connection between a Raspberry Pi and your laptop. We used VNC server software to connect the Pi to our laptop. Installing the VNC server on yourPiallows youto seetheRaspberryPi"sdesktopremotely,usingthemouseand keyboard as if you were sitting right in front of your Pi. It also means that you can put your Pianywhere else in your home and still control it. Also, the internet can be shared from your laptop"s WiFi over Ethernet. This also lets you access the internet on the Pi and connect it to your laptop display.

SettingupyourRaspberryPi

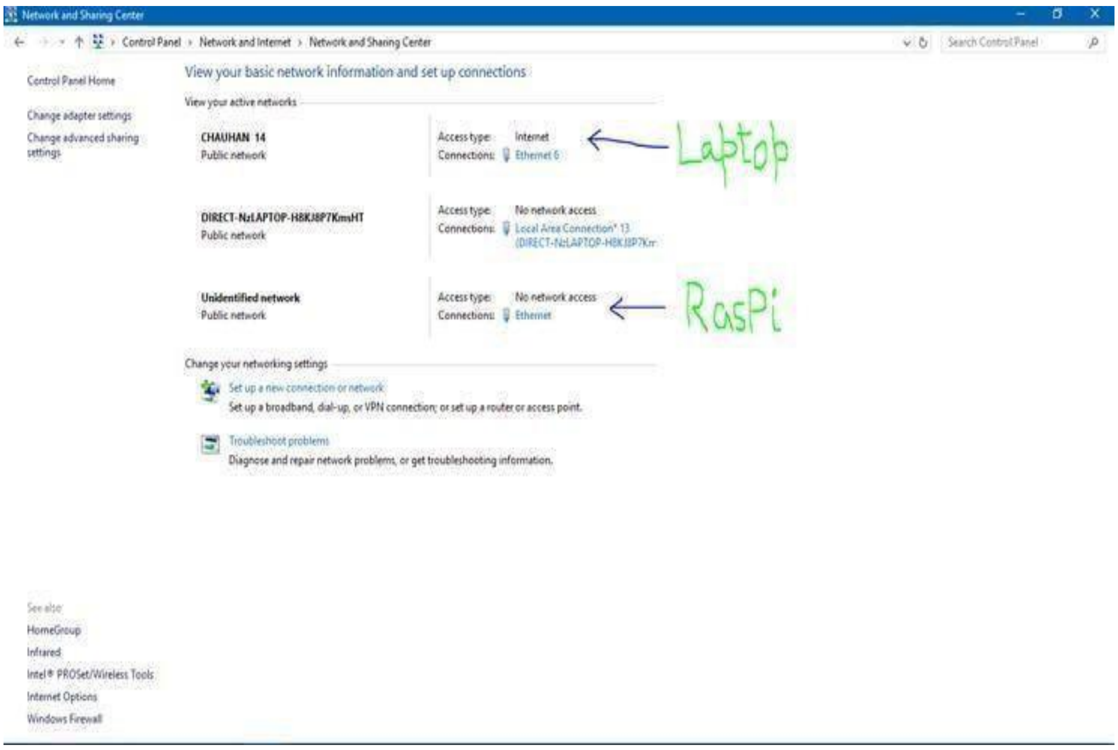
Before moving to connect your Raspberry Pi to your laptop display, you need an SD cardwith the OS preinstalled, or install **Raspbian**on a blank SD card. You will find lots of blogs and tutorials on preparing an SD card for the Raspberry Pi. If you are a beginner, you can simply click**here**and know more about this. This will show how to install the OS for the Raspberry Pi. You can also buy SD cards with the **Raspbian** and**NOOBs**operating systems preinstalled.*I would suggest you install* the*latest**full Raspbian OS image**from the official Raspberry Pi website as it is having VNC Server in the OS package.*

After setting up your SD Card, insert it into the Raspberry Pi. Next, connect your power adapter to the Raspberry Pi to power it. Also, connect your Raspberry Pi to the laptop via an Ethernet cable and connect a keyboard and mouse to it.

Note:*You need screen and a mouse after booting a new OS into Pi for the first time as by default, the SSH and VNC are disabled in Pi. Without SSH disabled, we cannot enable the PuTTY Configuration.*

SharingInternetOverEthernet

This step explains how you can share your laptop internet with the Raspberry Pi via Ethernet cable. In Windows: To share the internet with multiple users over Ethernet, go to **Network and Sharing Center.** Then click on the WiFi network:



Network and Sharing Center

Control Panel › Network and Internet › Network and Sharing Center

Search Control Panel

Control Panel Home

Change adapter settings

Change advanced sharing settings

View your basic network information and set up connections

View your active networks

CHAUHAN 14
Public network

Access type: Internet
Connections: Ethernet 6

DIRECT-NzLAPTOP-H8KJ8P7KmsHT
Public network

Access type: No network access
Connections: Local Area Connection* 13 (DIRECT-NzLAPTOP-H8KJ8P7Km

Unidentified network
Public network

Access type: No network access
Connections: Ethernet

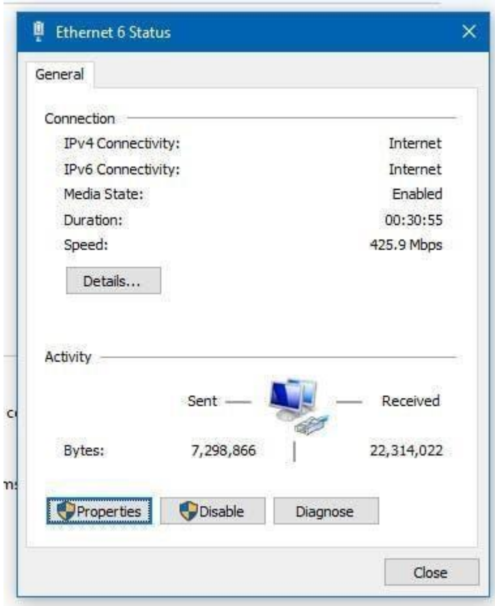
Change your networking settings

Set up a new connection or network
Set up a broadband, dial-up, or VPN connection; or set up a router or access point.

Troubleshoot problems
Diagnose and repair network problems, or get troubleshooting information.

See also
HomeGroup
Infrared
Intel® PROSet/Wireless Tools
Internet Options
Windows Firewall

Click on Properties (shown below), then go to **Sharing**and click on**"Allow other network users to connect".**Make surethat thenetworking connection is changed to theconnection of the Raspberry Pi. In my case, it is **Ethernet**:



Ethernet 6 Status

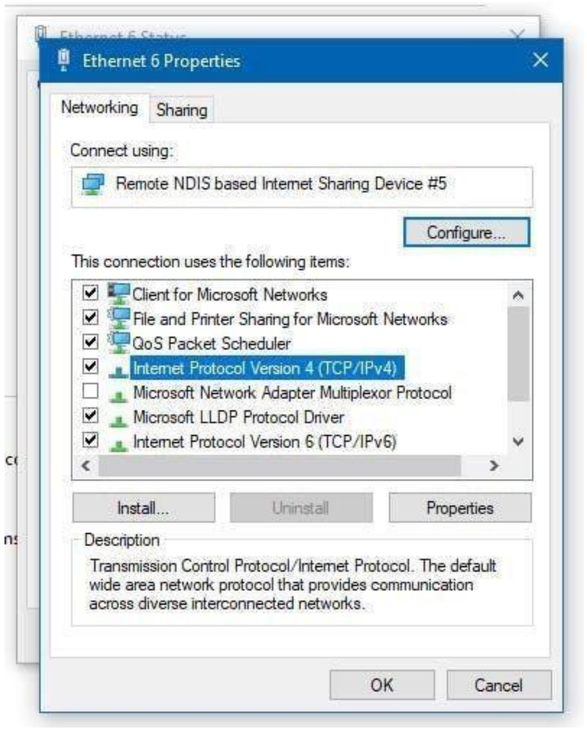
General

Connection

```
IPv4 Connectivity:        Internet
IPv6 Connectivity:        Internet
Media State:              Enabled
Duration:                 00:30:55
Speed:                    425.9 Mbps
```

Activity

```
            Sent                  Received
Bytes:      7,298,866             22,314,022
```



Ethernet 6 Properties

Networking | Sharing

Connect using:

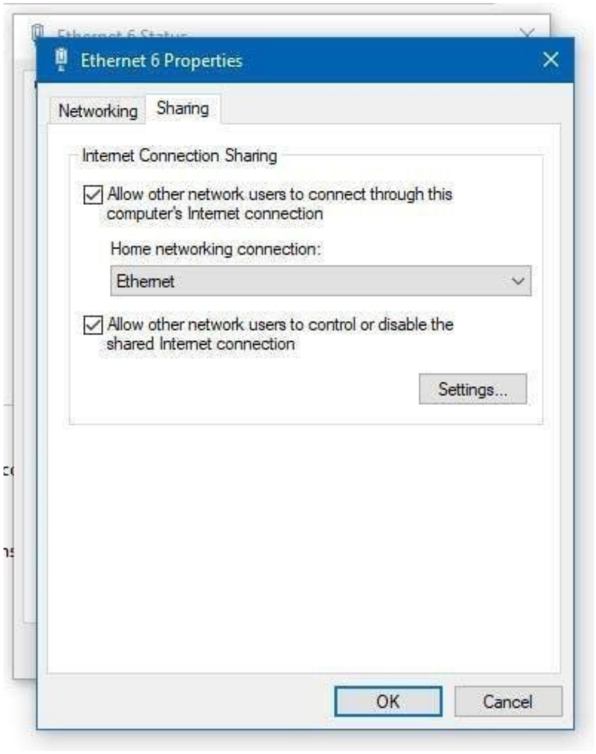
Remote NDIS based Internet Sharing Device #5

This connection uses the following items:

- [✓] Client for Microsoft Networks
- [✓] File and Printer Sharing for Microsoft Networks
- [✓] QoS Packet Scheduler
- [✓] Internet Protocol Version 4 (TCP/IPv4)
- [ ] Microsoft Network Adapter Multiplexor Protocol
- [✓] Microsoft LLDP Protocol Driver
- [✓] Internet Protocol Version 6 (TCP/IPv6)

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.



Ethernet 6 Properties

Networking | Sharing

Internet Connection Sharing

- [✓] Allow other network users to connect through this computer's Internet connection

Home networking connection: Ethernet

- [✓] Allow other network users to control or disable the shared Internet connection

FindingIPforPuTTYConfiguration

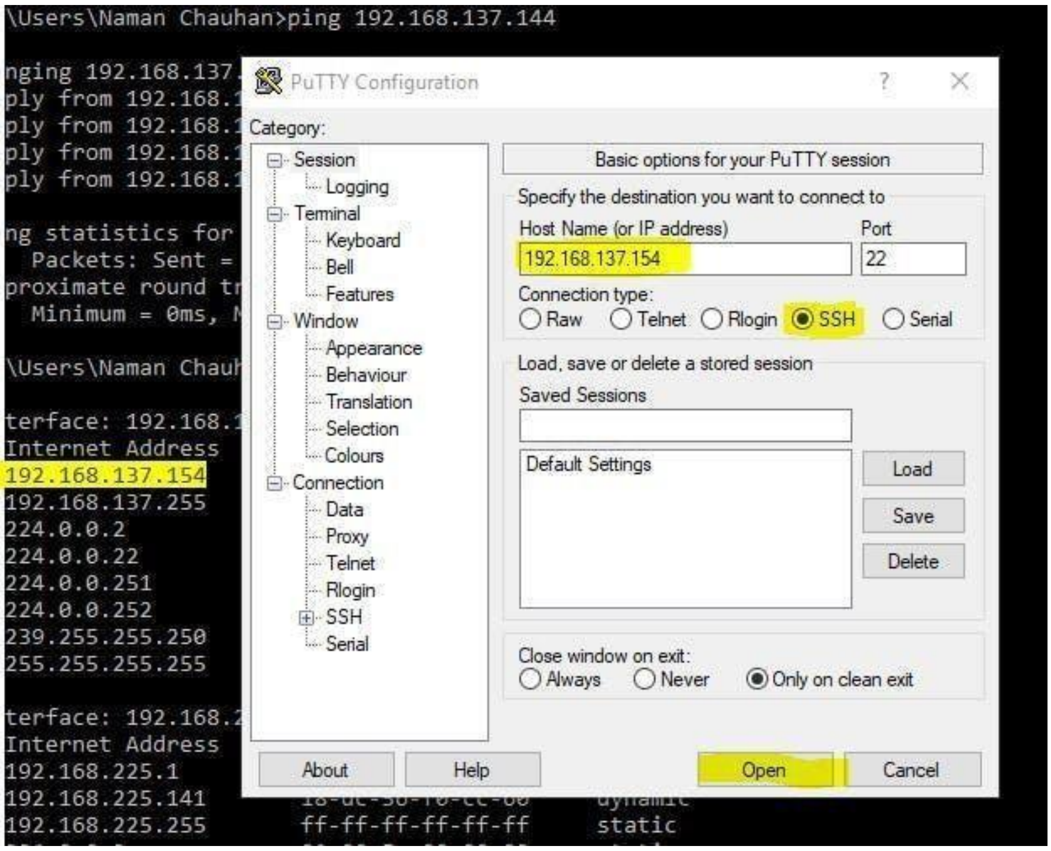
By default, the laptop will give a *dynamic IP*to the Raspberry Pi. Thus, we have to find out the IP address of Pi now.

As shown above, the IP assigned to my Pi is **192.168.137.144**. To check the IP assigned tothe connected Ethernet device, do the following. Considering that the IP assigned to your Piis **192.168.137.144** and the subnet mask is **255.255.255.0** :

- Openthecommandprompt.
- Pingat**raspberrypi.mshome.net**.
- Stopthepingafter5seconds.

Here,itis**192.168.137.154**.Notethissomewhere.

PuTTYConfigurationandVNConRaspberryPi



```
\Users\Naman Chauhan>ping 192.168.137.144

nging 192.168.137.
ply from 192.168.1
ply from 192.168.1
ply from 192.168.1
ply from 192.168.1

ng statistics for
  Packets: Sent =
proximate round tr
  Minimum = 0ms, M

\Users\Naman Chauh

terface: 192.168.1
Internet Address
192.168.137.154
192.168.137.255
224.0.0.2
224.0.0.22
224.0.0.251
224.0.0.252
239.255.255.250
255.255.255.255

terface: 192.168.2
Internet Address
192.168.225.1
192.168.225.141
192.168.225.255        ff-ff-ff-ff-ff-ff     static
```

PuTTY Configuration

Category: Session, Logging, Terminal, Keyboard, Bell, Features, Window, Appearance, Behaviour, Translation, Selection, Colours, Connection, Data, Proxy, Telnet, Rlogin, SSH, Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

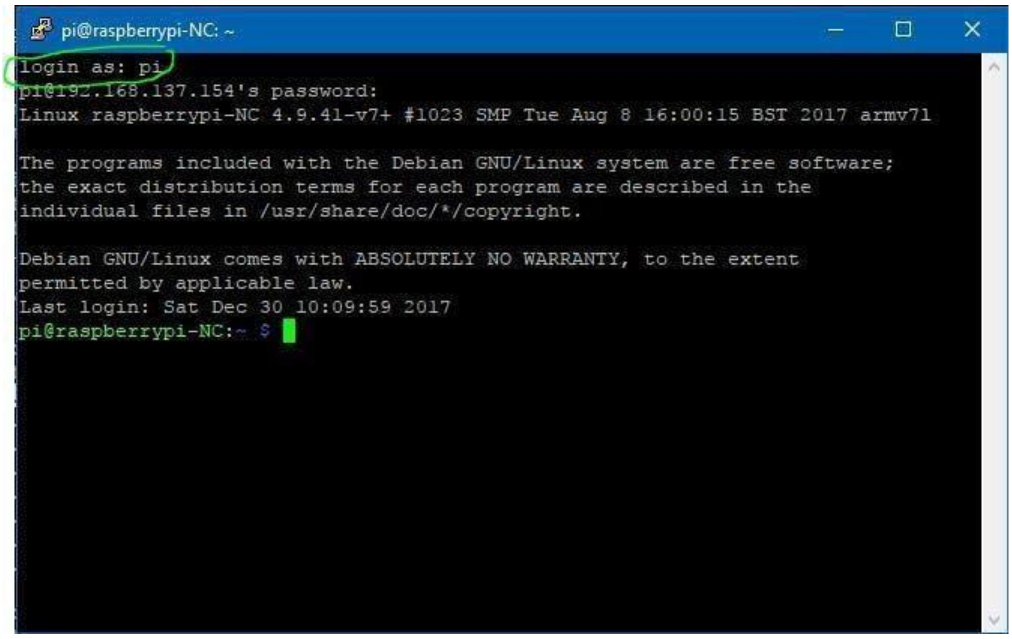
Host Name (or IP address): 192.168.137.154   Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin [✓ SSH] ○ Serial

Load, save or delete a stored session

Saved Sessions: Default Settings

Close window on exit: ○ Always ○ Never [✓ Only on clean exit]



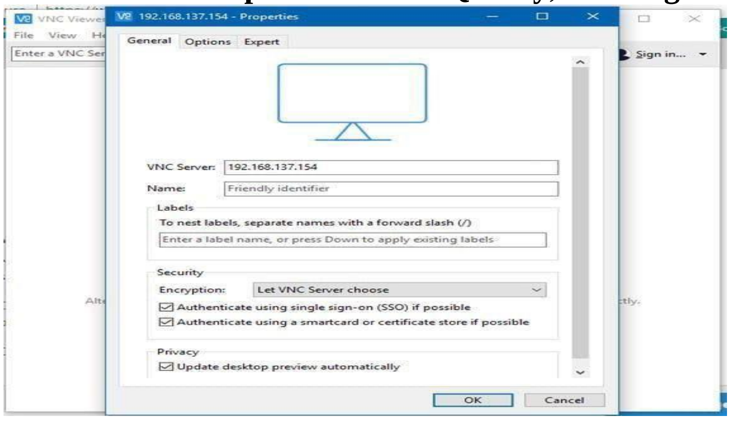
```
pi@raspberrypi-NC: ~
login as: pi
pi@192.168.137.154's password:
Linux raspberrypi-NC 4.9.41-v7+ #1023 SMP Tue Aug 8 16:00:15 BST 2017 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Dec 30 10:09:59 2017
pi@raspberrypi-NC:~ $
```

- IntheHostName,entertheIPAddresswenoted fromthecommandline.
- EnsurethattheConnectionTypeisSSH.
- HitEnterorclickonOpen toproceed.
- Now, anew window will open. It looks like anormal terminal window ofthecomputer but it is *Raspberry Pi's terminalwindow* accessible on your laptop.
- Itisdisplay-loginas:
- Enterpiastheusername.
- Enterthepasswordyou setfortheRaspberryPi.Thedefaultpasswordisraspberry
- If the password is correct, the Pi will load and you will access the terminal window of thePi.
- Now,youneedtostarttheVNCServer.Enterafterthe $sign-sudovncserver:1
- ThisistoinitializetheVNCServerontheRaspberryPi.

VNCServerandVNCVieweron Laptop

Now, Raspberry isready to connect using VNC. We just need to install the VNC server onthe laptop.

- Download**VNC Client**and install it. Now, download the**VNC Viewer**and install it on the laptop.
- OpentheVNCServerandtheVNCViewer now.
- IntheVNCViewer,clickon**File**>**NewConnection**.
- Enter**IPAddress**andin**Options**>**PictureQuality**,select**High.**


192.168.137.154 - Properties

General | Options | Expert

VNC Server: 192.168.137.154
Name: Friendly identifier

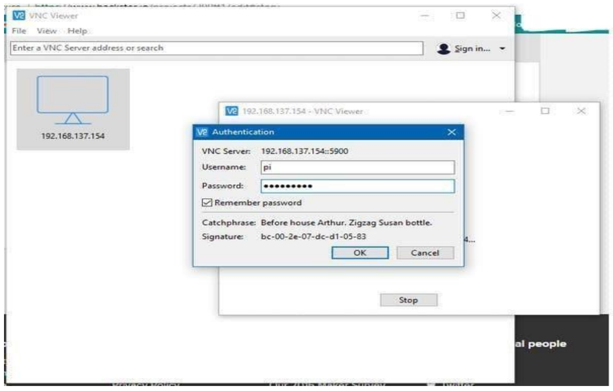
Labels
To nest labels, separate names with a forward slash (/)
Enter a label name, or press Down to apply existing labels

Security
Encryption: Let VNC Server choose
☑ Authenticate using single sign-on (SSO) if possible
☑ Authenticate using a smartcard or certificate store if possible

Privacy
☑ Update desktop preview automatically

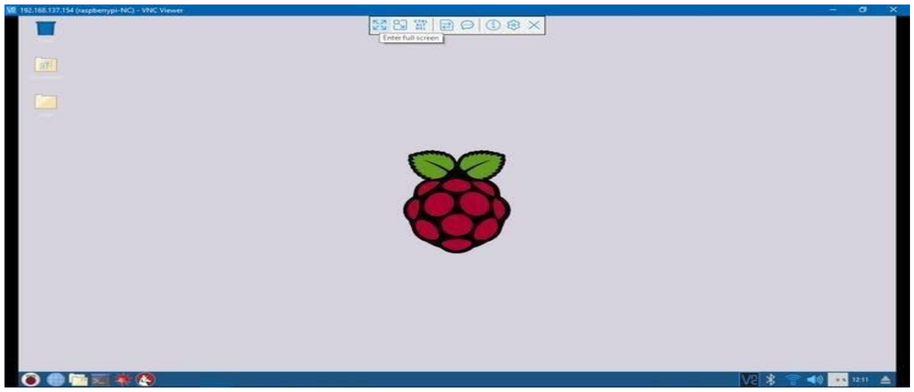
OK | Cancel

- ClickOK.Now,doubleclickontheIP Address.
- Enter**pi**inUsernameandyourPi'spassword (defaultis **raspberry**).
- Clickon**RememberPassword** sothat youdon'thavetoenterthisnext time.
- Clickon **OK**.



VNC Server: 192.168.137.154::5900
Username: pi
Password: ●●●●●●●●●●
☑ Remember password
Catchphrase: Before house Arthur. Zigzag Susan bottle.
Signature: bc-00-2e-07-dc-d1-05-83
OK Cancel

As you hit enter and all the things are correct, the Raspberry Pi Desktop will load in a new window. You can go into a full-screen mode by clicking on the options available above onthe window.



Ex. No:9 **Give overview of Zetta**
Date: 03/09/24

What is Zetta?

Zetta is an open source platform for IoT on which we can build APIs for device interaction. The platform is built on Node.js. People who are familiar with Node.js can easily get started with Zetta but, for beginners, a basic understanding of Node.js is required.

The Zetta platform and its characteristics

- Zetta is an open source platform, so anyone can use it free of cost. If you are passionate about Node.js (*https://nodejs.org/*), then you can contribute to this open source project. Basically, it"s a tool that will help to generate APIs which we can use to communicate between devices.
- Node.js is basically a server-side JavaScript. Developers can define devices as state machines using JavaScript. It is also cross-platform and is easily deployable in multiple cloud platforms.
- Zetta is an API driven platform. Every call is API based so that we can use these APIs for any other purpose like sending data to other analytics platforms.
- Zetta exposes Websocket endpoints to stream real-time events. This model of merging Hypermedia with Websocket streaming is acknowledged as Reactive Hypermedia.
- It can support almost all device protocols, and mediate them to HTTP. Connections between servers are also persistent, so that we can use the seamless services between servers in the cloud.
- We can create stateless applications in Zetta servers. Applications can be useful to connect devices, run different queries and interact between them. We can also write queries and triggers so that whenever new devices are attached, we will be notified.

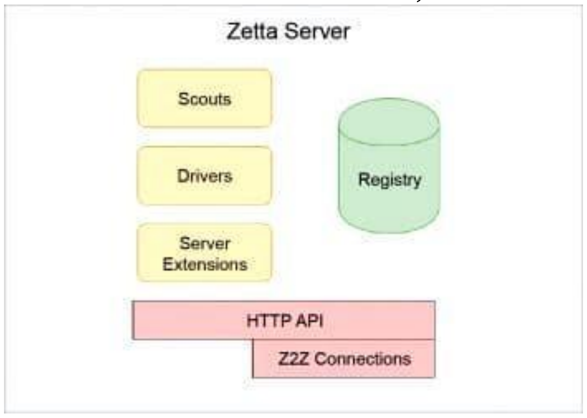


Figure 1: Zetta architecture

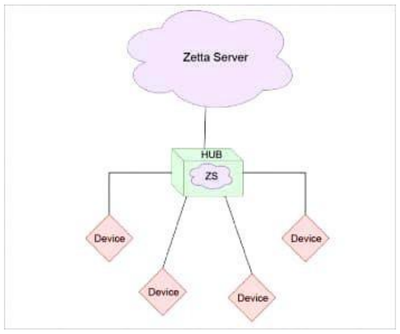


Figure 2: Zetta deployment

Zetta architecture

The Zetta server: The Zetta server is the main component of Zetta, which contains different sub-components like drivers, scouts, server extensions and apps. A Zetta server will run on a hardware hub such as BeagleBone Black, Raspberry Pi or Intel Edison. The server manages interactions between all the sub-components in order to communicate with devices and generate APIs, by which consumers can interact.

Scouts: Scouts provide a discovery mechanism for devices on the networks or to those which require system resources to understand and communicate with a specific type of protocol.

Scouts help Zetta to search for devices based on a particular protocol. They also fetch specific information about devices and whether those devices have already interacted with Zetta or not. They maintain security credentials and related details while communicating.

Drivers: Drivers are used to represent the devices in a state machine format. They are used for modelling devices and physical interaction between devices. Device models are used to generate different API calls.

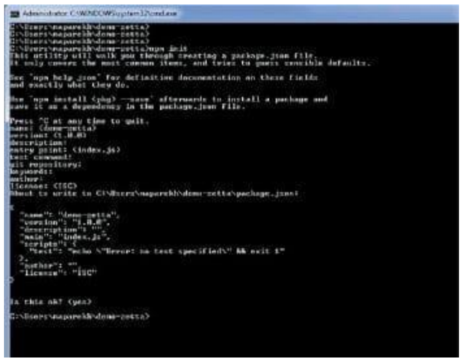
Server extensions: These are used for extending functionalities. They are in a pluggable mode, and deal with API management, adding additional securities, etc.

Registry: This is a database for the Zetta server, which stores information about the devices connected to the server. It is a persistence layer.

Secure linking: We can establish secure and encrypted tunnelling between different servers while communicating. This takes care of firewalls and network settings.

```
Microsoft Windows [illegible]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.
[illegible]
```

Figure 3: Node.js version



```
C:\Users\maparekh\demo-zetta>
C:\Users\maparekh\demo-zetta>
C:\Users\maparekh\demo-zetta>
C:\Users\maparekh\demo-zetta>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (demo-zetta)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\maparekh\demo-zetta\package.json:

{
  "name": "demo-zetta",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this ok? (yes)

C:\Users\maparekh\demo-zetta>
```

Figure 4: Creating the Node.js project

Apps: Apps that are used for different interactions between devices or to fetch and process some data are created in JavaScript. Apps can be created based on sensor streams or changes in devices. They can be used to track certain kinds of events that happen in systems.

Zetta deployment

Now let us explore the deployment of Zetta.

1. The Zetta server runs on a hardware hub, which can be Raspberry Pi, Intel Edison or BeagleBone Black.
2. The hub is connected to devices, and they communicate via HTTP to the specific protocols used in the deployment.
3. Another similar server runs in the cloud, which has the same Node.js packages that are available on the Zetta server in the hub. Both the servers are connected.

4. Zetta provides an API at the cloud endpoint so that consumers can use it.

Hardware requirements: Zetta runs with approximately six connected devices per hub, which is the ideal scenario suggested by it. Hardware requirements are dependent upon the number of devices, the load of each device and the type of data flowing between them. The ideal minimum requirement is a 500MHz CPU, 500MB RAM and storage of 1GB-2GB. Zetta requires 500MB to be able to run. It supports all common operating systems with 32-bit and 64-bit versions.

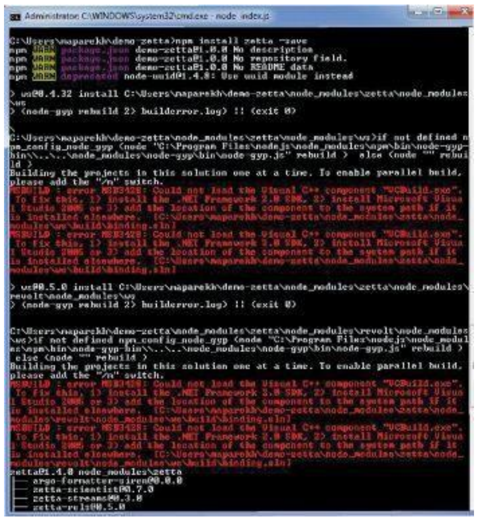
Zetta installation and demo project

Now let‟s take a look at installing Zetta and a „Hello world‟ version of the Zetta sample project.

Node.js

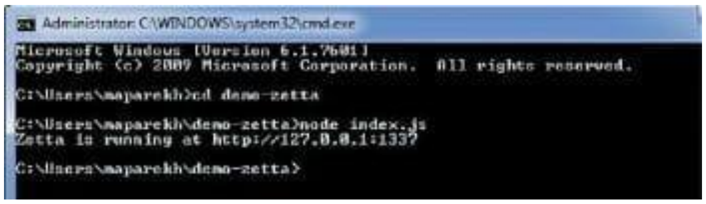
This is built on Chrome‟s JavaScript runtime for building scalable and faster applications. It uses an event-driven, non-blocking IO model. It is popular and very efficient for real-time applications running across distributed systems. Basically, it‟s a server-side JavaScript.

Go to the official site *https://nodejs.org* and from the *Downloads* section, download the appropriate installer file based on the operating system. More details about creating projects in Node.js are described in the following steps.



```
Administrator: C:\WINDOWS\system32\cmd.exe - node index.js

C:\Users\maparekh\demo-zetta>npm install zetta -save
npm WARN package.json demo-zetta@1.0.0 No description
npm WARN package.json demo-zetta@1.0.0 No repository field.
npm WARN package.json demo-zetta@1.0.0 No README data
npm WARN deprecated node-uuid@1.4.8: Use uuid module instead

> ws@0.4.32 install C:\Users\maparekh\demo-zetta\node_modules\zetta\node_modules
\ws
> (node-gyp rebuild 2> builderror.log) || (exit 0)

C:\Users\maparekh\demo-zetta\node_modules\zetta\node_modules\ws>if not defined n
pm_config_node_gyp (node "C:\Program Files\nodejs\node_modules\npm\bin\node-gyp-
bin\\..\..\node_modules\node-gyp\bin\node-gyp.js" rebuild )  else (node "" rebui
ld )
Building the projects in this solution one at a time. To enable parallel build,
please add the "/m" switch.
MSBUILD : error MSB3428: Could not load the Visual C++ component "VCBuild.exe".
 To fix this, 1) install the .NET Framework 2.0 SDK, 2) install Microsoft Visua
l Studio 2005 or 3) add the location of the component to the system path if it
is installed elsewhere.  [C:\Users\maparekh\demo-zetta\node_modules\zetta\node_
modules\ws\build\binding.sln]
MSBUILD : error MSB3428: Could not load the Visual C++ component "VCBuild.exe".
 To fix this, 1) install the .NET Framework 2.0 SDK, 2) install Microsoft Visua
l Studio 2005 or 3) add the location of the component to the system path if it
is installed elsewhere.  [C:\Users\maparekh\demo-zetta\node_modules\zetta\node_
modules\ws\build\binding.sln]

> ws@0.5.0 install C:\Users\maparekh\demo-zetta\node_modules\zetta\node_modules\
revolt\node_modules\ws
> (node-gyp rebuild 2> builderror.log) || (exit 0)

C:\Users\maparekh\demo-zetta\node_modules\zetta\node_modules\revolt\node_modules
\ws>if not defined npm_config_node_gyp (node "C:\Program Files\nodejs\node_modul
es\npm\bin\node-gyp-bin\\..\..\node_modules\node-gyp\bin\node-gyp.js" rebuild )
 else (node "" rebuild )
Building the projects in this solution one at a time. To enable parallel build,
please add the "/m" switch.
MSBUILD : error MSB3428: Could not load the Visual C++ component "VCBuild.exe".
 To fix this, 1) install the .NET Framework 2.0 SDK, 2) install Microsoft Visua
l Studio 2005 or 3) add the location of the component to the system path if it
is installed elsewhere.  [C:\Users\maparekh\demo-zetta\node_modules\zetta\node_
modules\revolt\node_modules\ws\build\binding.sln]
MSBUILD : error MSB3428: Could not load the Visual C++ component "VCBuild.exe".
 To fix this, 1) install the .NET Framework 2.0 SDK, 2) install Microsoft Visua
l Studio 2005 or 3) add the location of the component to the system path if it
is installed elsewhere.  [C:\Users\maparekh\demo-zetta\node_modules\zetta\node_
modules\revolt\node_modules\ws\build\binding.sln]
zetta@1.4.0 node_modules\zetta
├── argo-formatter-siren@0.0.0
├── zetta-scientist@0.7.0
├── zetta-streams@0.3.0
├── zetta-rels@0.5.0
```

Figure 5: Installing the Zetta Node.js module



Administrator: C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\maparekh>cd demo-zetta

C:\Users\maparekh\demo-zetta>node index.js
Zetta is running at http://127.0.0.1:1337

C:\Users\maparekh\demo-zetta>
```

Figure 6: Node.js Zetta server status

Zetta installation

For Zetta installation, the first thing required is Node.js. As discussed, download the Node.js installer on to your system. This will install both Node.js and npm (node package manager). So we don‟t need to install anything separately; it‟s a complete package. We can verify the versions by using the commands shown in Figure 3.

Creating a Zetta project

1. Create a new directory to save the project, e.g., *demo-zetta.*
2. Now *cd* to that directory. Here, it‟s *cd demo-zetta*.
3. To create a new Node.js project, run the command given below:
npm init
4. You will be asked for basic information about the project. By default, it will choose the value if you press *Enter*. If you want to change the value, then do so and press *Enter* several times and finish the installation. Basically, it will create a *package.json* file, which contains meta data about the project and its dependencies.
5. Now we will install the Zetta Node.js module. Here, the *-save* option adds Zetta to the *package.json* dependencies list.
npm install zetta -save

After all these steps, we have a basic Zetta project, which contains a *package.json* file and a *node_modules* directory.
Next, let‟s configure the Zetta server.

Zetta server configuration

[
We can install the Zetta server locally as a Zetta hub or in the cloud. We can link both the servers to access it from everywhere. Here we will install it locally for demo purposes.
1. Go to the *demo-zetta* directory.
2. Create a new file called *index.js*, and copy the code given below into it:

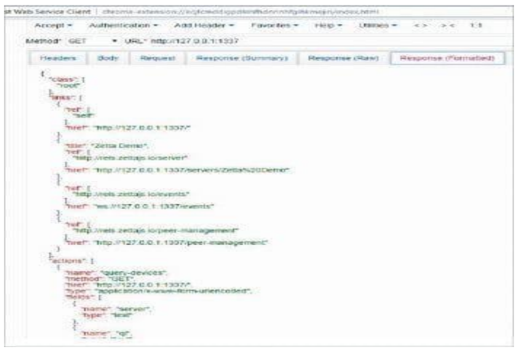
```
var zetta = require(„zetta‟);
zetta()
.name(„Zetta Demo‟)
.listen(1337, function(){
console.log(„Zetta is running at http://127.0.0.1:1337‟);
});
```

3. Now save and close the file.
After performing the steps given above, we have configured a basic Zetta server hub.

Starting the server

In the *demo-zetta* directory, enter the command given below:
node index.js

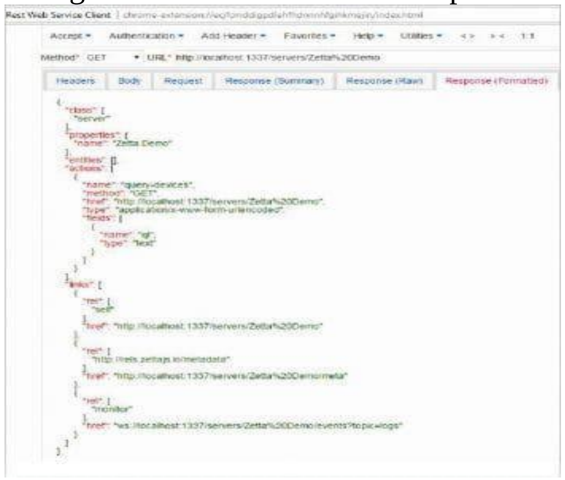
Figure 6 demonstrates the output of the above command. It shows the status of a running server.



Method: GET URL: http://127.0.0.1:1337

```
{
  "class": [
    "root"
  ],
  "links": [
    {
      "rel": [
        "self"
      ],
      "href": "http://127.0.0.1:1337/"
    },
    {
      "title": "Zetta Demo",
      "rel": [
        "http://rels.zettajs.io/server"
      ],
      "href": "http://127.0.0.1:1337/servers/Zetta%20Demo"
    },
    {
      "rel": [
        "http://rels.zettajs.io/events"
      ],
      "href": "ws://127.0.0.1:1337/events"
    },
    {
      "rel": [
        "http://rels.zettajs.io/peer-management"
      ],
      "href": "http://127.0.0.1:1337/peer-management"
    }
  ],
  "actions": [
    {
      "name": "query-devices",
      "method": "GET",
      "href": "http://127.0.0.1:1337/",
      "type": "application/x-www-form-urlencoded",
      "fields": [
        {
          "name": "server",
          "type": "text"
        },
        {
          "name": "ql",
```

Figure 7: The Zetta API call response



Method: GET URL: http://localhost:1337/servers/Zetta%20Demo

```
{
  "class": [
    "server"
  ],
  "properties": {
    "name": "Zetta Demo"
  },
  "entities": [],
  "actions": [
    {
      "name": "query-devices",
      "method": "GET",
      "href": "http://localhost:1337/servers/Zetta%20Demo",
      "type": "application/x-www-form-urlencoded",
      "fields": [
        {
          "name": "ql",
          "type": "text"
        }
      ]
    }
  ],
  "links": [
    {
      "rel": [
        "self"
      ],
      "href": "http://localhost:1337/servers/Zetta%20Demo"
    },
    {
      "rel": [
        "http://rels.zettajs.io/metadata"
      ],
      "href": "http://localhost:1337/servers/Zetta%20Demo/meta"
    },
    {
      "rel": [
        "monitor"
      ],
      "href": "ws://localhost:1337/servers/Zetta%20Demo/events?topic=logs"
    }
  ]
}
```

Figure 8: Demo Zetta server API response

Calling the Zetta API

Now, let‟s call the Zetta API. We need to call the server‟s root URL for that. Here we can use the *curl* command with *http://127.0.0.1:1337* or any REST client tools.
In the URL section of the REST client, enter *http://127.0.0.1:1337* and submit the request.
Now, in the response (formatted) section, you can see the response (see Figure 7). Check it for more information.
The Zetta server returns a JSON object that describes the root class of the API. The response demonstrates the current API state and links to resources given by the API. This is the basic API, which is not doing anything much as we don‟t have devices attached. Once we add the devices, the API will show more information.
Zetta API follows the Siren hypermedia specification. For more information on that, you can visit *https://github.com/kevinswiber/siren*.
Zetta API is a built-in feature of Zetta, which automatically generates APIs for devices. We can deploy these APIs in the cloud, which allows any authorised user to communicate with these devices from anywhere.

Ex. No. 10: Interfacing Temperature Humidity Sensor with Node MCU Esp8266

Date: 10/09/24

Required Hardware

1. **Node MCU ESP 8266**
2. **DHT 11**
3. **Bread Board**
4. **Micro USB Cable**
5. **LED**

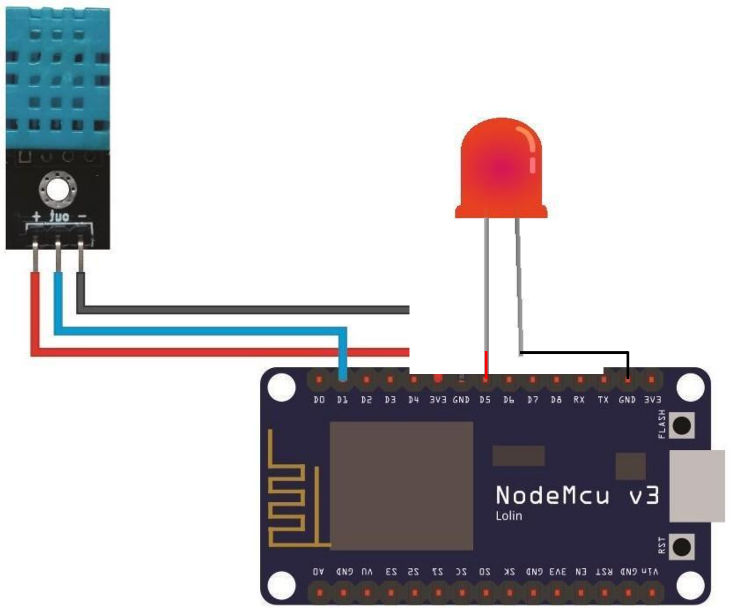
Software

1. **Arduino IDE**

Hardware Connection

1. **Connect DHT11 VCC pin with 3.3v pin in Node MCU**
2. **Connect DHT11 GND pin with GND in Node MCU**
3. **Connect DHT11 D0 pin with A0 in Node MCU**
4. **Connect LED +ve pin with D5**
5. **Connect LED -ve pin with**

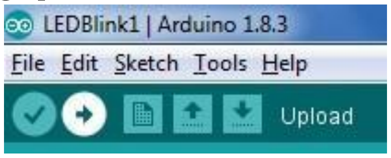
GNDSchematic



+ out -
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
FLASH
NodeMcu v3
Lolin
RST
A0 GND VU S3 S2 S1 SC SO SK GND 3V3 EN RST GND Vin

Arduino Script

Now write following code and upload it into the Board using upload button



```
#include "DHT.h"
#include <ESP8266WiFi.h>
//.................................................................
#define DHTTYPE DHT11 //--> Defines the type of DHT sensor used (DHT11, DHT21,
and DHT22), in this project the sensor used is DHT11.
const int DHTPin = D1; //--> The pin used for the DHT11 sensor is Pin D1 = GPIO5
DHT dht(DHTPin, DHTTYPE); //--> Initialize DHT sensor, DHT dht(Pin_used,
Type_of_DHT_Sensor);

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  delay(500);
  pinMode(D5,OUTPUT);
  dht.begin();  //--> Start reading DHT11 sensors
  delay(500);
}

void loop() {
  int h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();

  String Temp = "Temperature : " + String(t) + " °C";
  String Humi = "Humidity : " + String(h) + " %";
  Serial.println(Temp);
  Serial.println(Humi);
  Serial.println("============================");
  delay(1000);

  if(t > 32)
  {
    digitalWrite(D5,HIGH);
  }
  else
  {
    digitalWrite(D5,LOW);  }  }
```

Ex. No.: 11: Interfacing MQ 4 GAS Sensor with Node MCU Esp8266

Datee: 18/09/24

Required Hardware

1. **Node MCU ESP 8266**
2. **MQ4 Gas Sensor**
3. **Bread Board**
4. **Micro USB Cable**
5. **LED or Buzzer**

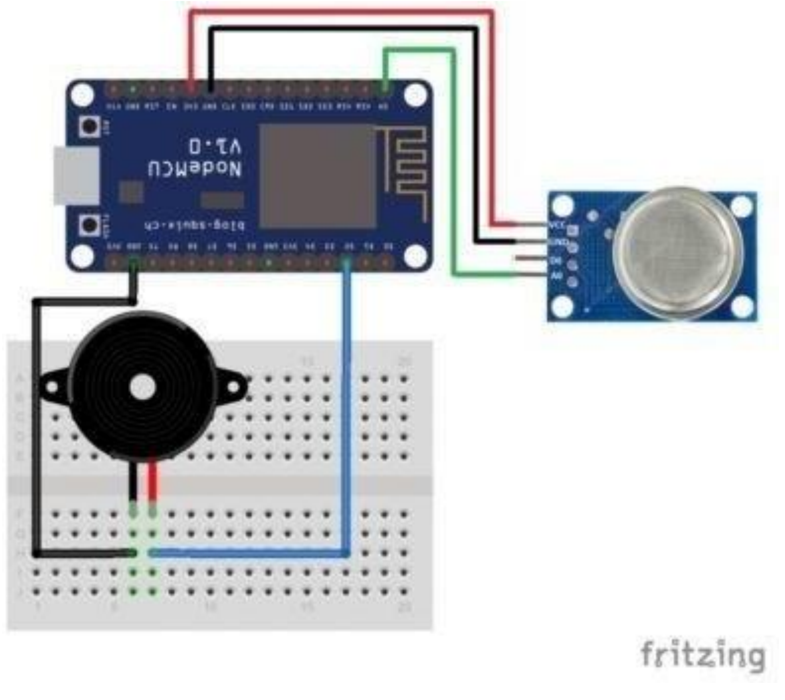
Software

1. **Arduino IDE**

Hardware Connection

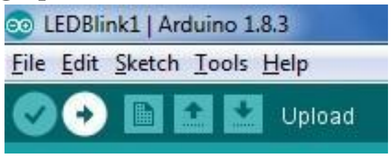
1. **Connect MQ4 VCC pin with 3.3v pin in Node MCU**
2. **Connect MQ4 GND pin with GND in Node MCU**
3. **Connect MQ4 A0 pin with A0 in Node MCU**
4. **Connect Buzzer +ve pin with D2**
5. **Connect Buzzer -ve pin with GND**

Schematic



Arduino Script

Now write following code and upload it into the Board using upload button



```
#include <ESP8266WiFi.h>
#define smokeA0 A0
#define Alarm D2

// Your threshold value
int sensorThres = 500;

void setup() {
  pinMode(Alarm, OUTPUT);
  pinMode(smokeA0, INPUT);
  Serial.begin(9600);
}

void loop() {
  // Checks if it has reached the threshold value
  if (Sensor > 700)
  {
    digitalWrite(Alarm, HIGH);
  }
  else
  {
    digitalWrite(Alarm, LOW);


  }
  delay(100);
}
```

Ex. No. :12 Interfacing MQ 4 GAS Sensor with Node MCU Esp8266

Date: 25/09/24

Required Hardware

1. **Node MCU ESP 8266**
2. **Pulse Sensor**
3. **Bread Board**
4. **Micro USB Cable**
5. **LED or Buzzer**

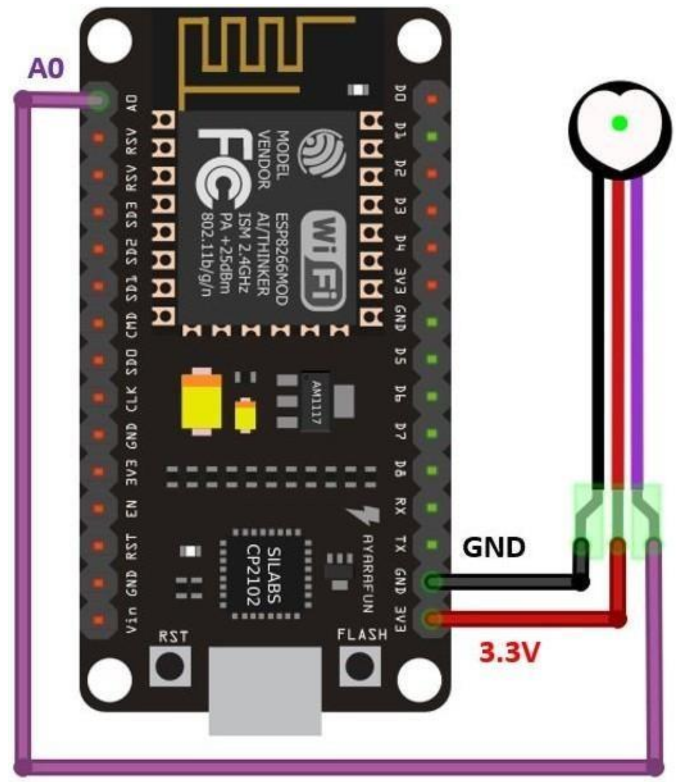
Software

1. Arduino IDE

Hardware Connection

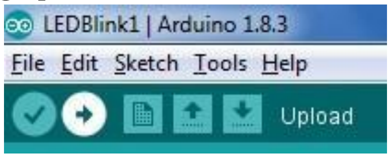
1. **Connect Pulse Sensor VCC pin with 3.3v pin in Node MCU**
2. **Connect Pulse Sensor GND pin with GND in Node MCU**
3. **Connect Pulse Sensor A0 pin with A0 in Node MCU**
4. **LED +ve pin with D5**
5. **LED -ve pin with GND**

Schematic



Arduino Script

Now write following code and upload it into the Board using upload button



```
//  Variables
int PulseSensorPurplePin = 0;        // Pulse Sensor PURPLE WIRE connected to
ANALOG PIN 0
int LED13 = 13;  //  The on-board Arduion LED

int Signal;                // holds the incoming raw data. Signal value can range from 0-1024
int Threshold = 550;            // Determine which Signal to "count as a beat", and which to
ingore.

// The SetUp Function:
void setup() {
  pinMode(LED13,OUTPUT);         // pin that will blink to your heartbeat!
   Serial.begin(9600);         // Set's up Serial Communication at certain speed.

}

// The Main Loop Function
void loop() {

  Signal = analogRead(PulseSensorPurplePin);  // Read the PulseSensor's value.
                                              // Assign this value to the "Signal" variable.

   Serial.println(Signal);                    // Send the Signal value to Serial Plotter.

   if(Signal > Threshold){                          // If the signal is above "550", then "turn-on"
Arduino's on-Board LED.
     digitalWrite(LED13,HIGH);
   } else {
     digitalWrite(LED13,LOW);                //  Else, the sigal must be below "550", so "turn-
off" this LED.
   }
delay(10);
```

Ex. No.: 13 **Interfacing Relay and Control LIGHT with Node MCU Esp8266**

Date: 01/10/24

Required Hardware

1. **Node MCU ESP 8266**
2. **LDR resistor**
3. **Realy**
4. **Bread Board**
5. **Micro USB Cable**

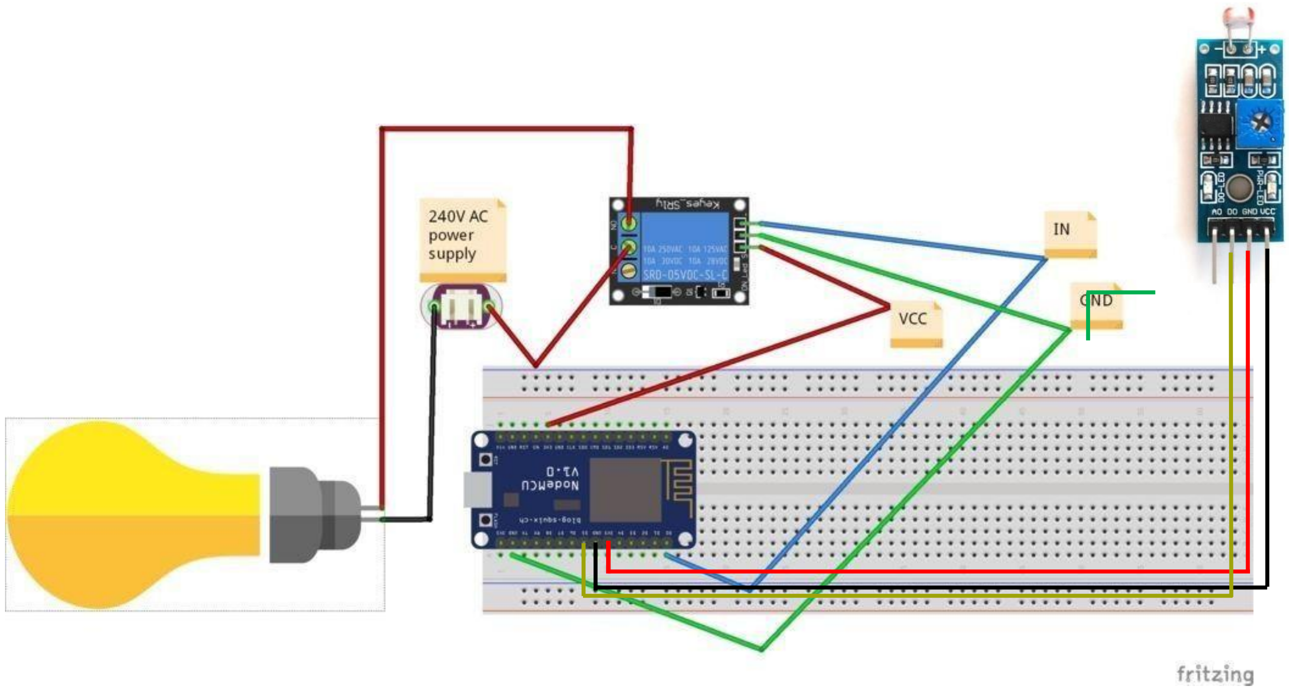
Software

1. **Arduino IDE**

Hardware Connection

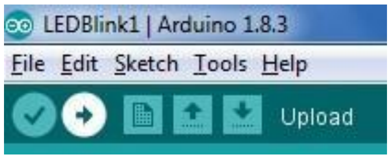
1. **Connect LDR VCC pin with 3.3v pin in Node MCU**
2. **Connect LDR GND pin with GND in Node MCU**
3. **Connect LDR D0 pin with D5 in Node MCU**
4. **Connect Relay VCC pin with 3.3v**
5. **Connect Relay GND pin with GND**
6. **Connect Relay IN pin with D0**

Schematic



Arduino Script

Now write following code and upload it into the Board using upload button



```
void setup() {
// initialize serial communication at 9600 bits per second:
Serial.begin(9600);
pinMode(D5, INTPUT); //LDR SENSOR
pinMode(D0, OUTPUT); //RELAY SWITCH

}

void loop() {
// read the input on analog pin 0:
int sensorValue = digitalRead(D5);

// print out the value you read:
Serial.println(sensorValue);

If(sensorValue < 200)
{
        digitalWrite(D0, HIGH);
}
else
{
        digitalWrite(D0, LOW);
}



}
```

Ex. No. 14: Create a Web server using Node MCU Esp8266

Date: 08/10/24

Required Hardware

1. **Node MCU**
2. **LED**
3. **Bread Board**
4. **Micro USB Cable**

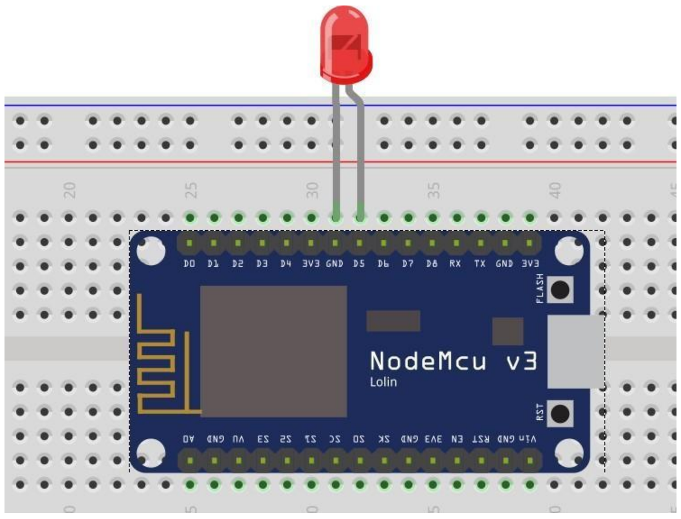
Software

1. Arduino IDE

Hardware Connection

1. **Connect LED +ve pin with D5 pin**
2. **Connect LED -ve pin with GND**

Schematic



Procedure

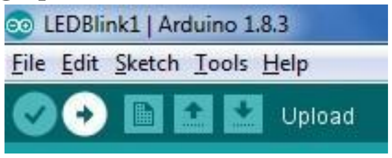
1. Connect Node MCU with WiFi modem or hotspot with SSID and Password.
2. Start serial in 9600 baurd rate.
3. Connect LED with D5 pin with 3.3v and Ground pin with GND.
4. Create Button with HTML
5. Write Arduino sketch and upload code into node mcu.

HTML CODE

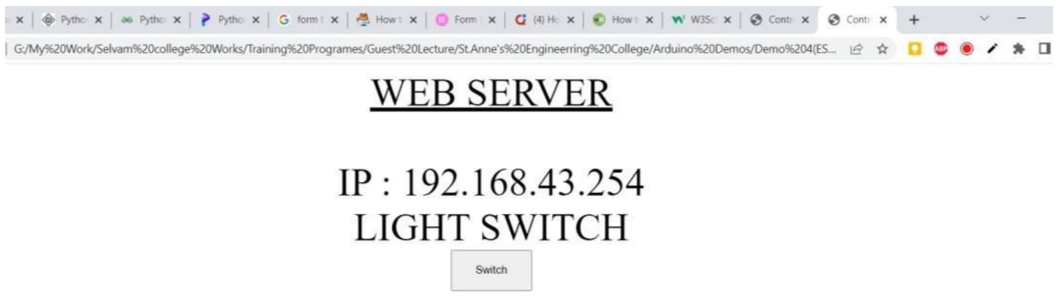
```
<html>

<head>

<title>Controll Device over Internet</title>

<style>

.button {
height: 50px;
width: 100px;
}

</style>

   </head>

<body>

<center>

<font size=20>

   <u>WEB SERVER </u><br><br>
   IP : 192.168.43.254<br>
   LIGHT SWITCH </font>

<form action="http://192.168.43.254/toggle">

   <input type="button" class="button" value="Switch" />

</form>

</body>

</html>
```

Arduino Script

Now write following code and upload it into the Board using upload button



```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

#define LED D5
ESP8266WebServer server;

char* ssid = "selmahome";
char* password = "26301985";

void setup() {
  // put your setup code here, to run once:
  pinMode(LED,OUTPUT);
  WiFi.begin(ssid,password);
  Serial.begin(9600);
  while(WiFi.status()!=WL_CONNECTED)
  {
    Serial.print(".");
    delay(500);
  }
  Serial.println("");
  Serial.println("IP Address");
  Serial.println(WiFi.localIP());


  server.on("/",[](){server.send(200,"text/plain","Hello World");});
  server.on("/toggle",toggleLED);
  server.begin();
}


void loop() {
  // put your main code here, to run repeatedly:
  server.handleClient();
}
void toggleLED()
{
  digitalWrite(LED,!digitalRead(LED));
  server.send(204,"");
}
```


G:/My%20Work/Selvam%20college%20Works/Training%20Programes/Guest%20Lecture/St.Anne's%20Engineerring%20College/Arduino%20Demos/Demo%204(ES...

WEB SERVER

IP : 192.168.43.254

LIGHT SWITCH

Ex. No. 15: Interfacing Temperature Humidity Sensor with Node MCU Esp8266

Date: 09/10/24

Required Hardware

1. **Node MCU ESP 8266**
2. **DHT 11**
3. **Bread Board**
4. **Micro USB Cable**
5. **LED**

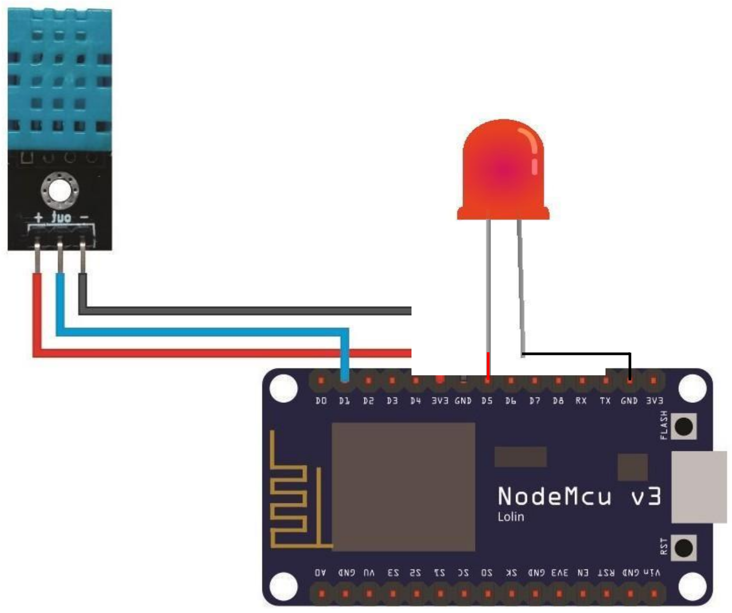
Software

1. **Arduino IDE**

Hardware Connection

1. **Connect DHT11 VCC pin with 3.3v pin in Node MCU**
2. **Connect DHT11 GND pin with GND in Node MCU**
3. **Connect DHT11 D0 pin with D1 in Node MCU**
4. **Connect LED +ve pin with D5**
5. **Connect LED -ve pin with GND**

Schematic

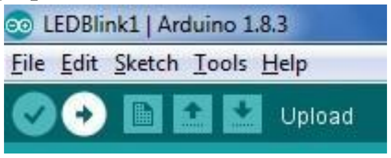

+ out −
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
FLASH
NodeMcu v3
Lolin
RST
A0 GND VU S3 S2 S1 SC S0 SK GND 3V3 EN RST GND Vin

Procedure to connect with cayenne

1. Create your own account if your a new user using the following link and login your account.
2. From the screen select All Devices
3. Select Generic ESP8266
4. Take note of the MQTT username, passwords and client ID
5. Connect the sensor to your ESP module following the attached schematics.
6. Upload the code by updating MQTT Username and Password.

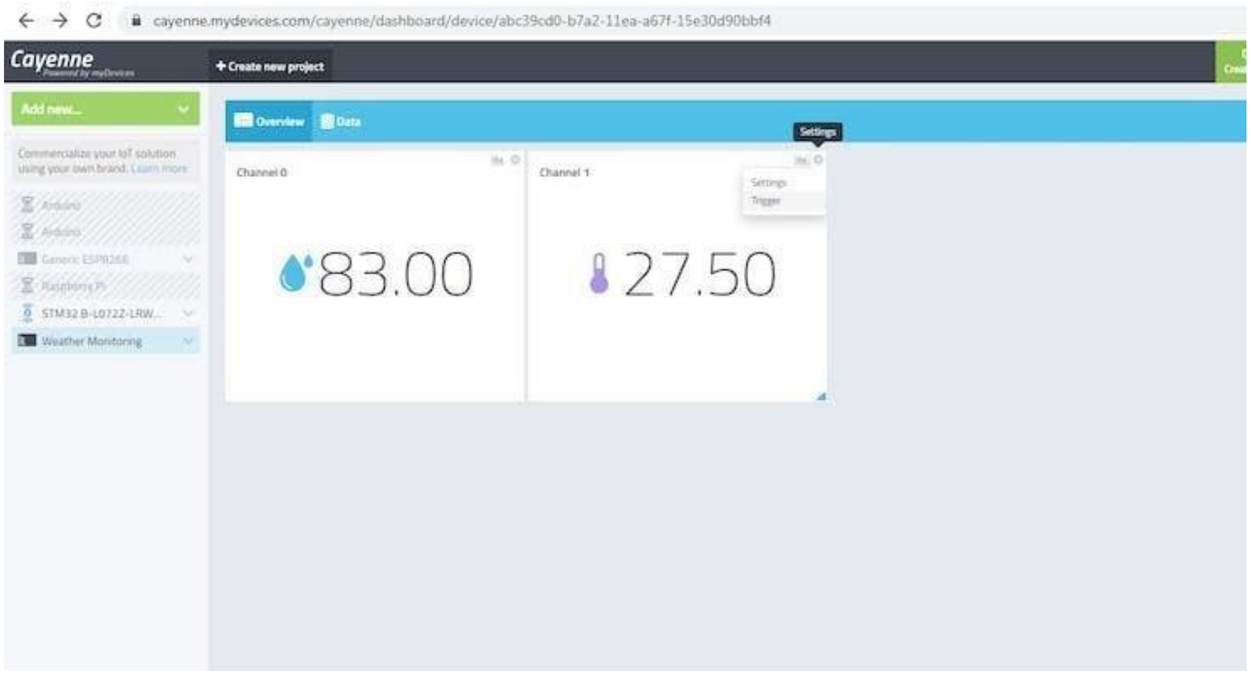
Arduino Script

1. Now write following code and upload it into the Board using upload button



```
/*
Cayenne DHT11 Sensor Project
*/

#define CAYENNE_PRINT Serial  // Comment this out to disable prints and save space
#define DHTPIN 0          // D1

// Uncomment whatever type you're using! In this project we used DHT11
#define DHTTYPE DHT11     // DHT 11
#include <CayenneMQTTESP8266.h>
#include <DHT.h>

// WiFi network info.
char ssid[] = "xxxxx";
char wifiPassword[] = "xxxxx";

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "dfsfdfdfs";
char password[] = "sdsdsd";
char clientID[] = "dsdsdsd";

DHT dht(DHTPIN, DHTTYPE);

//Variables for DHT11 values
float h;
float t;
bool humidity = false;
bool Temperature = false;


void setup()
{
        Serial.begin(9600);
  Serial.print("Setup");
        Cayenne.begin(username, password, clientID, ssid, wifiPassword);
  humidity = false;
  Temperature = false;
}

void loop()
{
            //Run Cayenne Functions
  Cayenne.loop();
}

CAYENNE_OUT(V0){
  //Serial.println("humidity");

    //Read humidity (percent)
  h = dht.readHumidity();
  delay(1000);
  Serial.print("humidity: ");
  Serial.println(h);
  //Set Humidity to true so we know when to sleep
  humidity = true;
  //Write to Cayenne Dashboard`
  Cayenne.virtualWrite(V0, h);
}

CAYENNE_OUT(V1){
  //Serial.println("temperature");

    //Read temperature as Celsius
    t = dht.readTemperature();

   delay(1000);

  Serial.print("temperature: ");
  Serial.println(t);

  //Set Temperature to true so we know when to sleep
  //Temperature = true;

  //Write to Cayenne Dashboard
  Cayenne.virtualWrite(V1, t);
}
```

Screenshot


cayenne.mydevices.com/cayenne/dashboard/device/abc39cd0-b7a2-11ea-a67f-15e30d90bbf4
Cayenne
Create new project
Add new...
Commercialize your IoT solution using your own brand. Learn more
Arduino
Arduino
Generic ESP8266
Raspberry Pi
STM32 B-L072Z-LRW...
Weather Monitoring
Overview
Data
Settings
Channel 0
Channel 1
Settings
Trigger
83.00
27.50

Ex. No. 16: Interfacing Relay and Control LIGHT with Blynk Cloud AppRequired Hardware

Date: 15/10/24

Requirements:

1. **Node MCU ESP 8266**
2. **Realy**
3. **Bread Board**
4. **Micro USB Cable**

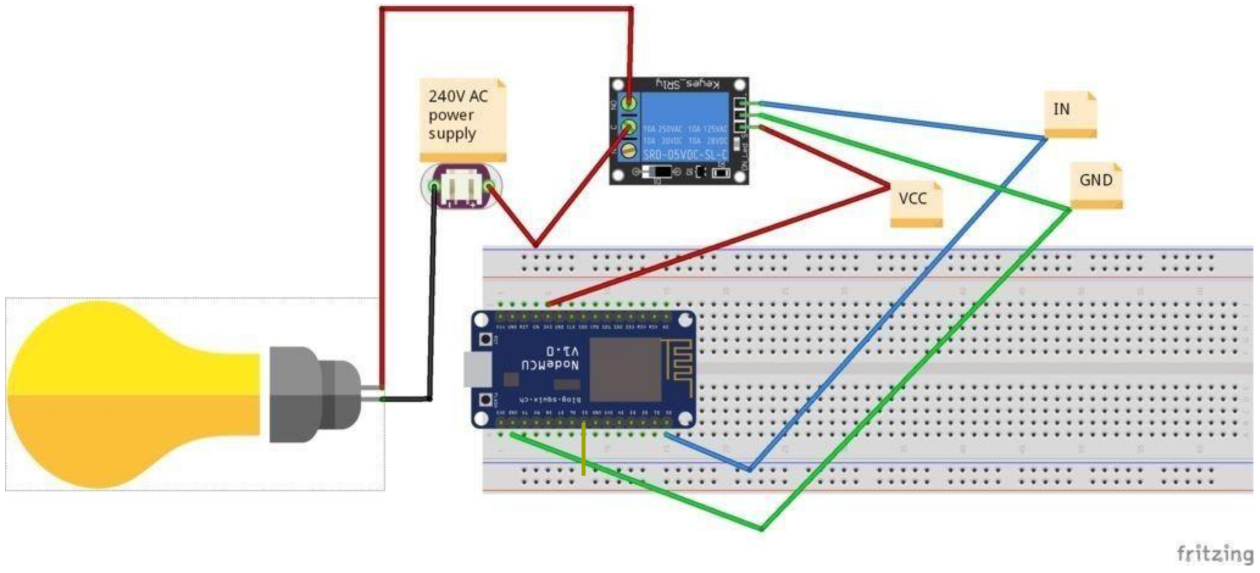
Software

Arduino IDE

Hardware Connection

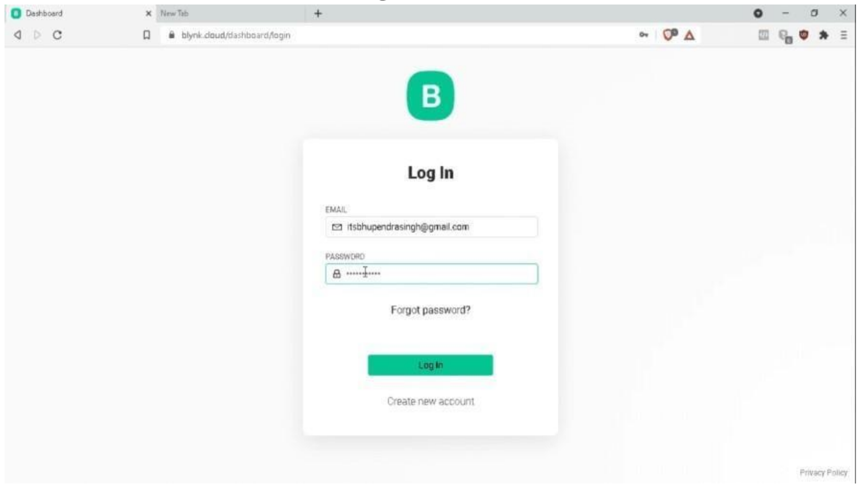
1. **Connect LDR D0 pin with D5 in Node MCU**
2. **Connect Relay VCC pin with 3.3v**
3. **Connect Relay GND pin with GND**
4. **Connect Relay IN pin with D0**

Schematic

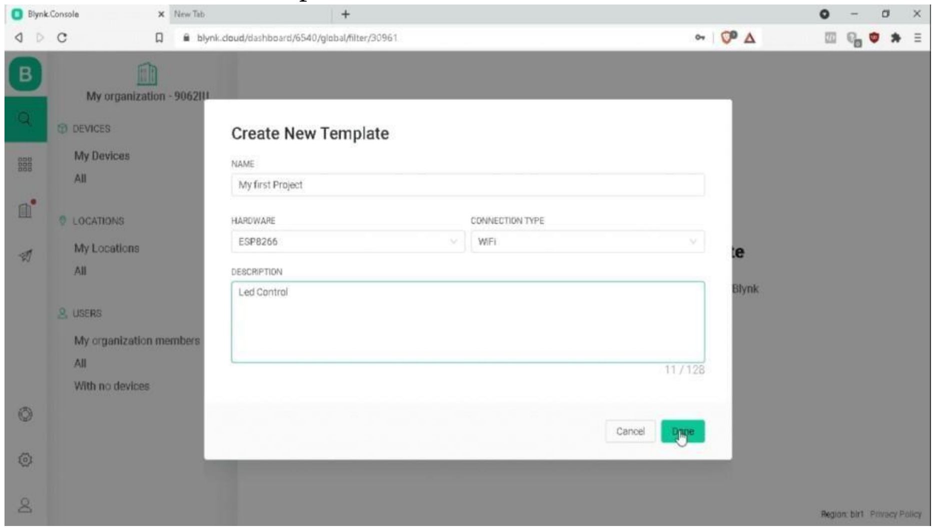


<u>Procedure to Connect with Blynk</u>

1. Open the browser and open the https://blynk.cloud/.
2. Create an new account and log in with



3. A new dashboard will be opened.
4. Here click on a new template, write the name of the project.
5. Select hardware like esp8266. ESP8266 means NodeMCU Series.


Create New Template

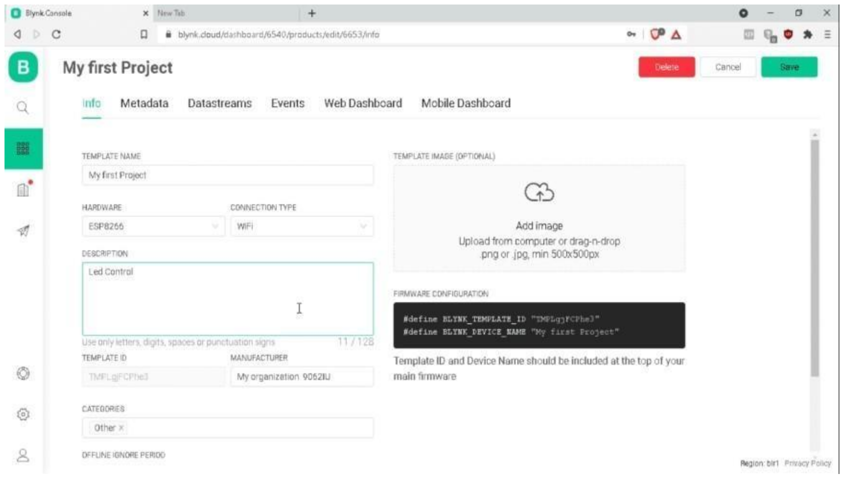
NAME: My first Project

HARDWARE: ESP8266

CONNECTION TYPE: WiFi

DESCRIPTION: Led Control

6. The project dashboard will be opened.



My first Project

TEMPLATE NAME: My first Project

HARDWARE: ESP8266

CONNECTION TYPE: WiFi

DESCRIPTION: Led Control

TEMPLATE ID: TMPLgjFCPhe3

MANUFACTURER: My organization 9062IU

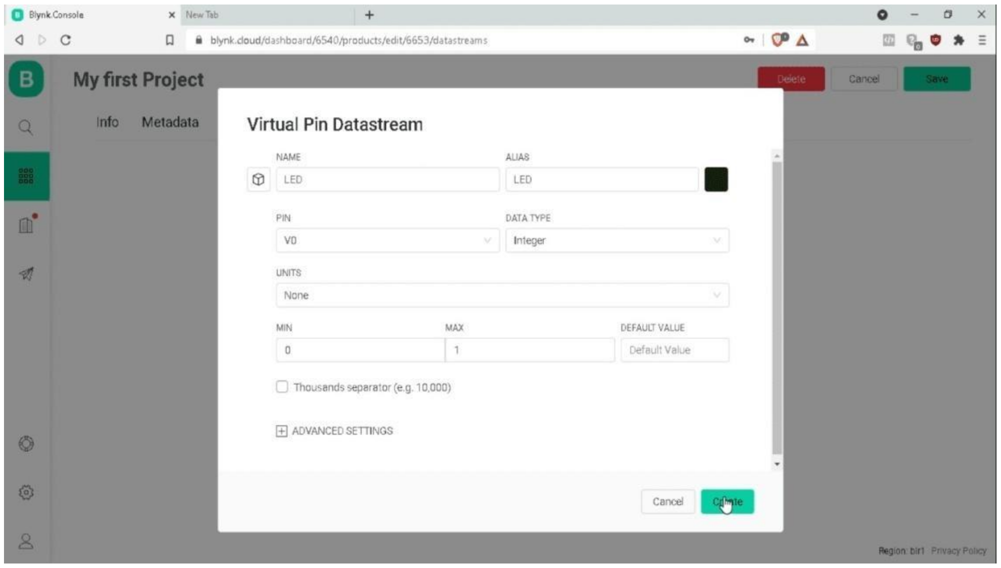
CATEGORIES: Other

FIRMWARE CONFIGURATION

```
#define BLYNK_TEMPLATE_ID "TMPLgjFCPhe3"
#define BLYNK_DEVICE_NAME "My first Project"
```

Template ID and Device Name should be included at the top of your main firmware

7. Copy the Template ID and Device name and write it to Arduino program.
8. Click on the new datastream, and select the virtual pin. Now write the name of virtual pin-like led. Select the V0 pin as our virtual pin and the integer as data type. After it, click to create. So here virtual pin has been created.



Virtual Pin Datastream

NAME: LED

ALIAS: LED

PIN: V0

DATA TYPE: Integer

UNITS: None

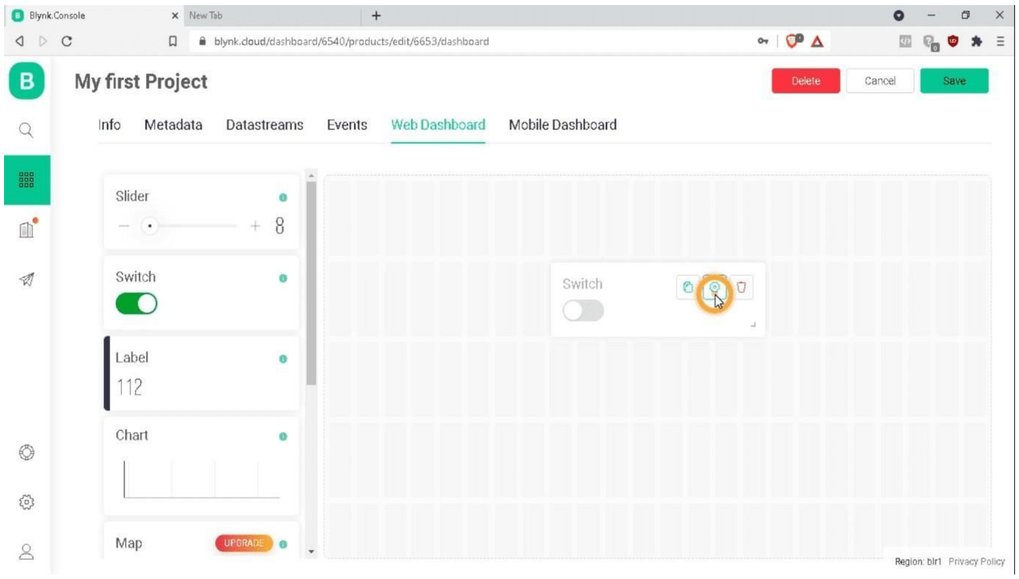
MIN: 0

MAX: 1

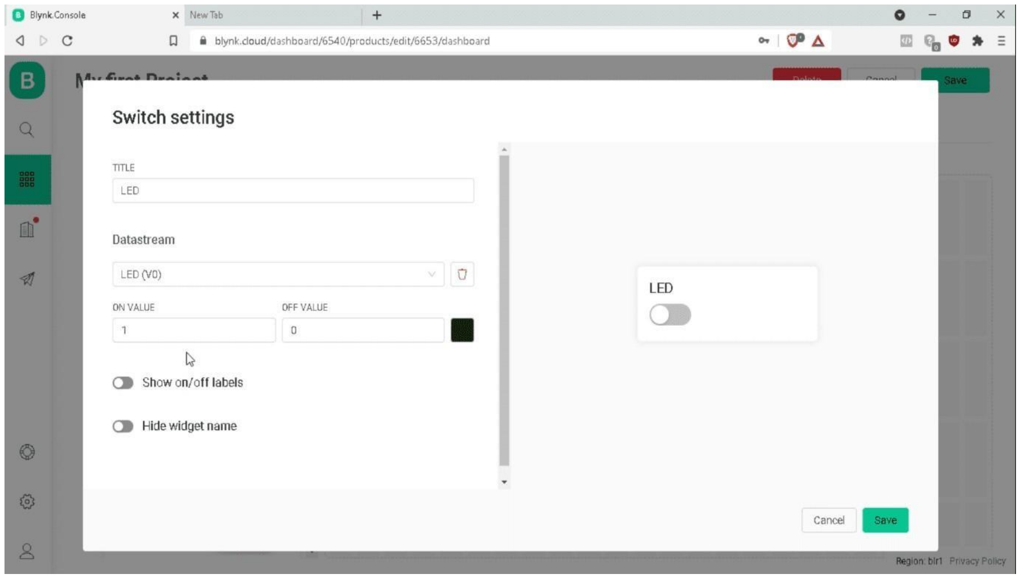
DEFAULT VALUE: Default Value

Thousands separator (e.g. 10,000)

9. Now In the web dashboard, drag and drop the switch widget to the dashboard and click on the setting.



10. Here choose the datastream source as LED (V0). Click on the Save button.


Blynk.Console
New Tab
blynk.cloud/dashboard/6540/products/edit/6653/dashboard
Switch settings
Cancel
Save
Region: blr1 Privacy Policy

TITLE
LED

Datastream
LED (V0)

ON VALUE
1

OFF VALUE
0

Show on/off labels

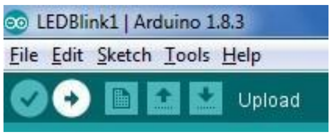
Hide widget name

LED

11. Now write the Arduino code and paste the credentials copied from blynk cloud.

Arduino Script

Now write following code and upload it into the Board using upload button



```
/* Comment this out to disable prints and save space */#define BLYNK_PRINT Serial

/* Fill-in your Template ID (only if using Blynk.Cloud) */
//#define BLYNK_TEMPLATE_ID
"YourTemplateID"#define
BLYNK_TEMPLATE_ID
"TMPLuOe0uKyw" #define
BLYNK_TEMPLATE_NAME "Smart
Light"
#define BLYNK_AUTH_TOKEN "YeBRZXt0NFTNyp-qSjKdp_jw_KC52KTp"

#include
<ESP8266WiFi.h>
#include
<BlynkSimpleEsp826
6.h>

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

// Your WiFi credentials.
// Set password to "" for
open networks.char ssid[]
= "xxxxxxxx";
char pass[] = "xxxxxxxx";

void setup()
{
  // Debug console Serial.begin(9600);


  Blynk.begin(auth, ssid, pass);
  // You can also specify server:
  //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 80);
  //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);
}
void loop()
{
  Blynk.run();
}
```