Modernized IoT

Experiment No: 1

Title: Study of Raspberry-Pi,Beagle board,Arduino,and different operating systems for Raspberry Pi/Beagle board/Arduino

Aim: Study of Raspberry-Pi, Beagleboard,Arduino,and different operating systems for Raspberry-Pi/Beagle board/Arduino. Understanding the process of OS installation on Raspberry-Pi/Beagle board/Arduino.

Introduction:

Internet of Things: - IoT is short for Internet of Things. The Internet of Things refers to the ever- growing network of physical objects that feature an IP address for internet connectivity, and the communication that occurs between these objects and other Internet-enabled devices and systems. The Internet of Things (IoT) refers to the use of intelligently connected devices and systems to leverage data gathered by embedded sensors and actuators in machines and other physical objects. In other words, the IoT(Internet of Things)can be called to any of the physical objects connected with network.

Examples of IoT:-

1) Apple Watch and HomeKit.

2) Smart Refrigerator.

3) Smart cars.

4) Google Glass.

5) Smart thermostats.

A) Raspberry-Pi:- The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote teaching of basic computer science in schools and in developing countries. It does not include peripherals (such as keyboards and mice). The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable device that enables people of all ages to explore computing ,and to learn how to program in languages like Scratch and Python. The Raspberry Pi is a credit-card-sized computer that costs between $5 and $35.It's available anywhere in the world, and can function as a proper desktop computer or be used to

build smart devices. A Raspberry Pi is a general-purpose computer, usually with a Linux operating system, and the ability to run multiple programs. Raspberry Pi is like the brain. Its primary advantage comes in processing higher level processing capability. It's a single board computer.

Fig.1:-Raspberry-Pi

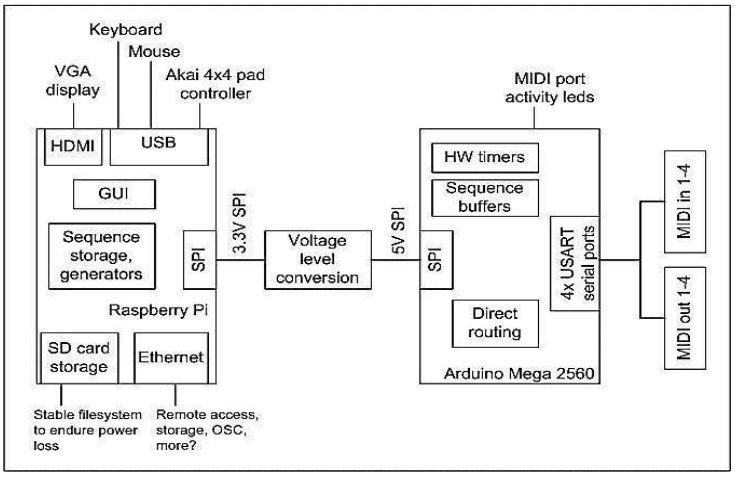


Fig.2:-Raspberry-PiArchitecture

Here are the various components on the RaspberryPi board:

- **ARMCPU/GPU**—This is a Broadcom BCM2835 System on a Chip(SoC) that's madeup of an ARM central processing unit (CPU) and a Video core4 graphics processing unit (GPU). The CPU handles all the computations that make a computer work (taking input, doing calculations and producing output), and the GPU handles graphics output.
- **GPIO** -- These are exposed general-purpose input/output connection points that will allow the real hardware hobbyists the opportunity to tinker.
- **RCA**—An RCA jack allows connection of analog TVs and other similar output devices.
- **Audioout**--This is a standard 3.55-millimeter jack for connection of audio output devices such as headphones or speakers. There is no audio in.
- **LEDs**--Light-Emitting Diodes,for your entire indicator light needs.
- **USB** -- This is a common connection port for peripheral devices of all types (including your mouse and keyboard).Model Ahasone, and ModelBhastwo.You can use a USB hub to expand the number of ports or plug your mouse into your keyboard if it has its own USB port.
- **HDMI**—This connector allows you to hook up a high-definition television or other compatible device using an HDMI cable.
- **Power**—This is a 5v MicroUSB power connector into which you can plug your compatible power supply.
- **SD card slot** -- This is a full-sized SD card slot. An SD card with an operating system (OS) installed is required for booting the device. They are available for purchase from the manufacturers, but you can also download an OS and save it to the card yourself if you have a Linux machine and the where withal.
- **Ethernet**—This connector allows for wired network access and is only available on the ModelB.

2

B) BeagleBoard:-The **BeagleBoard** is a low-power open-source single-board computer produced by Texas Instruments in association with Digi-Key and Newark element14. The Beagle Board was also designed with open-source software development in mind, and as a way of demonstrating the Texas Instrument's OMAP3530 system-on-a-chip. The board was developed by a small team of engineers as an educational board that could be used in colleges around the world to teach open-source hardware and software capabilities. It is also sold to the public under the Creative Commons share-alike license. The board was designed using Cadence Or CAD for schematics and Cadence Allegro for PCB manufacturing; no simulation software was used. **Beagle Bone** Blackisa low-cost, open source, community-supported development platform for ARM® Cortex™-A8 processor developers and hobbyists. Boot Linuxin under 10-seconds and get started on Sitara™AM335xARMCortex-A8processor development in less than 5minutes with just a single USB cable.

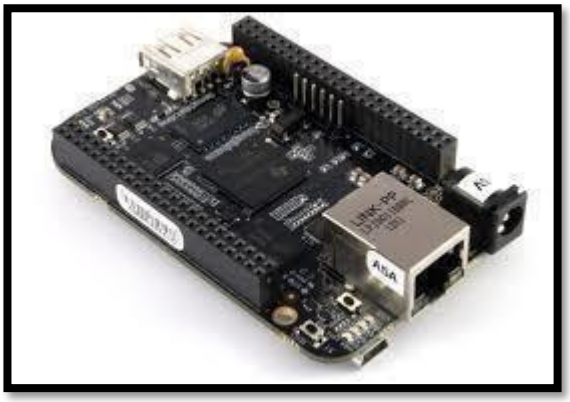


Fig.3:-Beagle Board Black

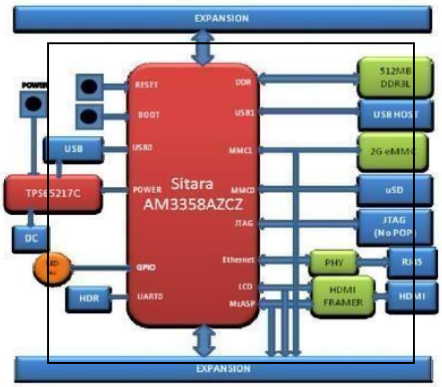


Fig.4:-Beagle Board Black architecture

HerearethevariouscomponentsontheBeagleboard: Processor:
AM335x 1GHz ARM® Cortex-A8

- 512MBDDR3RAM
- 4GB8-biteMMCon-boardflashstorage
- 3Dgraphicsaccelerator
- NEONfloating-pointaccelerator
- 2xPRU32-bitmicrocontrollers

Connectivity

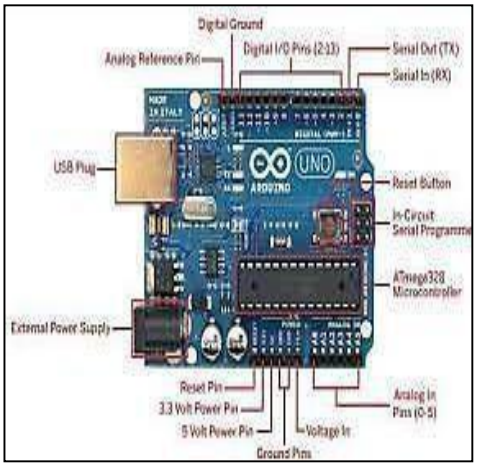
- USBclientforpower&communications
- USBhost
- Ethernet
- HDMI
- 2x46 pinheaders

SoftwareCompatibility

- Debian

- Android
- Ubuntu
- Cloud9IDEonNode.jsw/BoneScript library

C) Arduino: - Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical and digital world. Arduino board designs use a variety of microprocessorsandcontrollers.Theboardsareequippedwithsetsofdigitaland analoginput/output(I/O) pinsthatmaybe interfaced to various expansion boards or breadboards (shields) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languagesCandC++.Inadditiontousingtraditionalcompilertoolchains,the Arduinoprojectprovidesanintegrated developmentenvironment(IDE)based ontheProcessing languageproject.Arduinoisopen-source hardware.The hardwarereferencedesignsaredistributed underaCreativeCommonsAttributionShare-Alike2.5licenseandareavailable on the Arduino website. Layout and production files for some versionsofthehardwarearealsoavailable



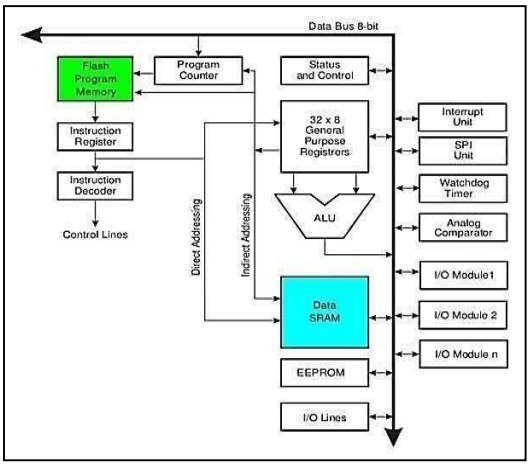


Fig.5:-ArduinoBoard

Fig.6:-ArduinoBoardArchitecture

HerearethevariouscomponentsontheArduinoboard:

Microcontrollers

ATmega328P(usedonmostrecentboards)

ATmega168(usedonmostArduinoDiecimilaandearlyDuemilanove)ATmega8 (used on some older board)

DigitalPins

Inaddition to the specific functions listed below,thedigital pins onan Arduino boardcanbe usedforgeneral purpose

input and output via the pinMode(), digitalRead(), and digitalWrite() commands. Each pin has an internal pull-up resistor whichcanbeturnedonandoffusingdigitalWrite()(w/ avalueofHIGHorLOW,respectively)whenthepinisconfigured as an input. The maximum current per pin is 40 mA.

AnalogPins

In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) usingtheanalogRead()function.Mostoftheanaloginputscanalsobeusedasdigitalpins:analoginput0asdigital pin 14through analog input 5 asdigital pin 19. Analoginputs 6and 7(present on the Mini and BT) cannot be used as digital pins.

PowerPins

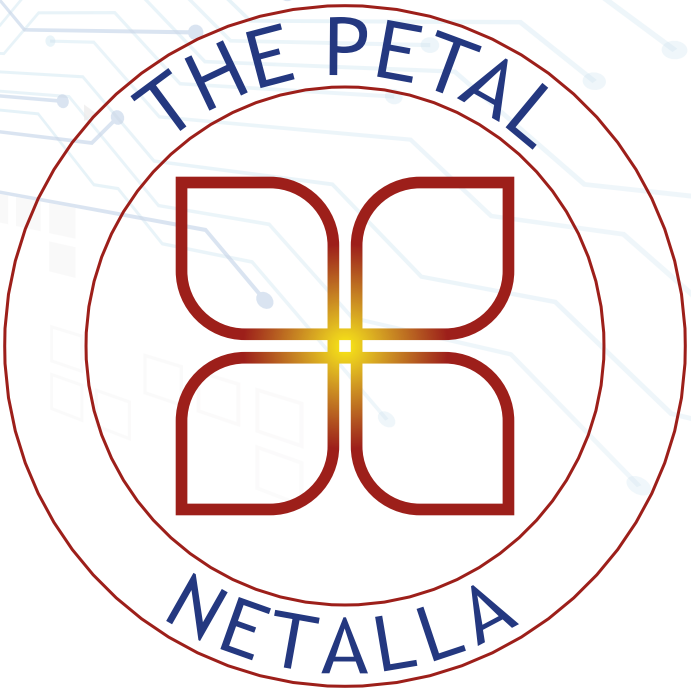
VIN(sometimes labelled "9V").Theinput voltagetothe Arduinoboard when it's using an externalpower source (asopposedto5voltsfromtheUSBconnectionorotherregulatedpowersource).Youcansupplyvoltage through this pin, or, if supplying voltage via the powerpack, access it through this pin. Note that different boards accept differentinputvoltagesranges,pleaseseethedocumentationforyourboard.AlsonotethattheLilypadhasnoVINpin and accepts only a regulated input.

OtherPins

- **AREF.**Referencevoltagefortheanaloginputs.UsedwithanalogReference().
- **Reset.** (Diecimila-only) Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.
- AnalogReferencepin(orange)
- DigitalGround(lightgreen)
- DigitalPins2-13(green)
- Digital Pins 0-1/Serial In/Out- TX/RX (dark green) - These pins cannot be used for digital i/o (**digitalRead**and **digitalWrite**) if you are also using serial communication (e.g., **Serial.begin**).
- ResetButton-S1(darkblue)
- In-circuitSerialProgrammer(blue-green)
- AnalogInPins0-5(lightblue)
- PowerandGroundPins(power:orange,grounds:lightorange)
- ExternalPowerSupplyIn(9-12VDC)-X1(pink)
- Toggles External PowerandUSB Power (place jumper on twopins closest to desiredsupply)- SV1 (purple)USB (usedfor uploadingsketchesto the boardandfor serialcommunication betweenthe boardand the computer; can be used to power the board)

D) ESP32 Development Board:

5


THE PETAL
NETALLA

The Petal Netalla IoT KIT

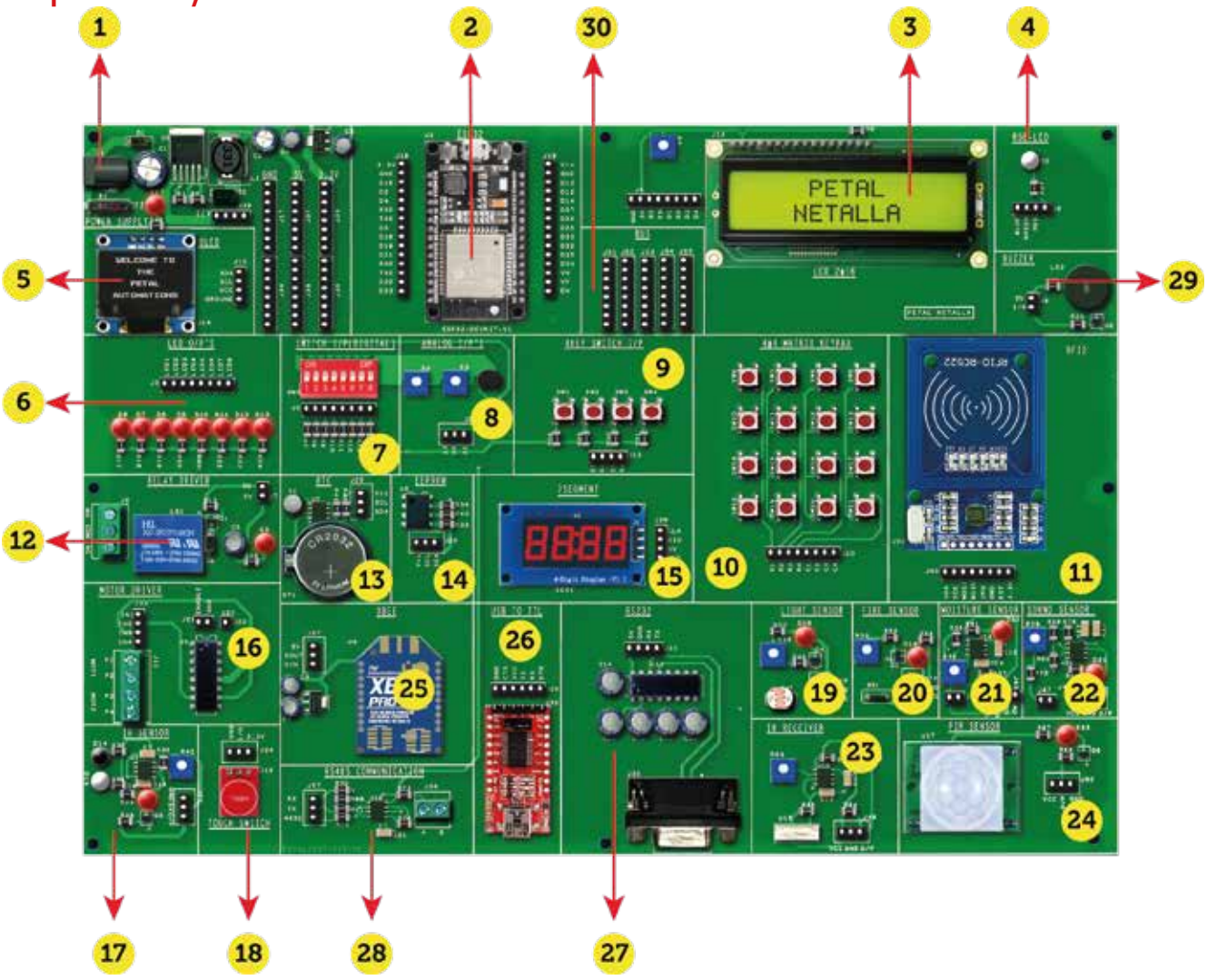
Innovate, Create, Transform: Get advanced high school and college students to take their first step in building internet-connected objects. Explore the Internet of Things with ESP32 Education.

Introduction

IoT kit teach students on how to **build internet-connected objects**, such as a home security alarm, a classroom counter, and urban farming device, with **step-by-step tutorials and examples**. All the activities adopt a learning-by-doing approach, through which students acquire knowledge by constructing fully functional solutions which teach them to write, deploy, and interact with board through our web application.

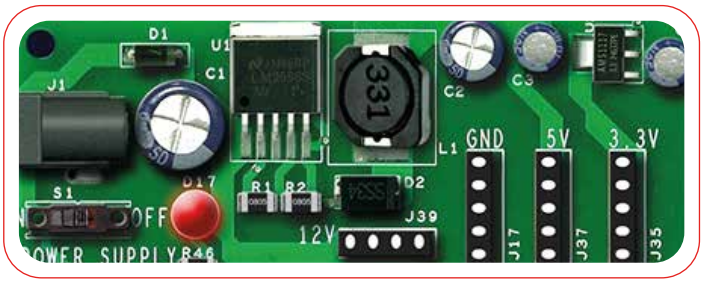
Create connections, decompose complex problems into simpler parts, allow students to innovate, and enhance their understanding of real-world technology with the Explore IoT Kit - an industry-standard IoT tool that will help prepare them for their future careers.

Complete Layout



1. Power Supply
2. ESP32
3. LCD (Liquid Crystal Display)
4. RGB LED
5. OLED Display
6. 8 Channel LED's
7. 8 Channel Switch's
8. Analog Inputs (2 channel variable resistors & LM35)
9. 4 Key Switch Input
10. 4 * 4 Matrix Keypad
11. RFID Reader
12. Relay Driver
13. RTC (Real Time Clock)
14. EEPROM
15. 7 Segment LED
16. Motor Driver
17. IR Sensor
18. Touch Switch
19. Light Sensor
20. Fire Sensor
21. Moisture Sensor
22. Sound Senor
23. IR Receiver
24. PIR Sensor
25. Xbee Adapter
26. USB to TTL
27. RS232 to USB
28. RS485
29. 5V Buzzer
30. BusConnection Port

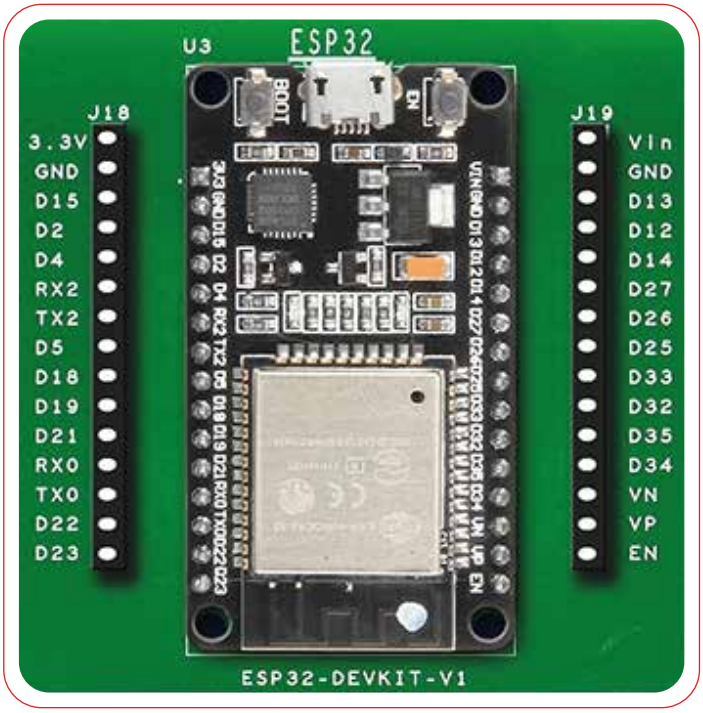
1. Power Supply



Here by 12V / 2A DC adaptor give with kit, and 3 different level following Voltages available for do multiple operations 12V, 5V and 3.3V.

2. ESP32

ESP32 is the elegant IoT Software Development kit, which has plug and play designed layout to make it so easy for the students, hobbyists, enthusiasts and professionals to get connected and focus more on Program and application development. It has developed with onboard IO's, Communication interfaces and peripherals. Every one easily design, experiment with and test circuits without using soldering methods. It's applied in many educational institutions.



MCU

- ESP32-D0WD-V3 embedded, Xtensa® dual core 32-bit LX6 microprocessor, upto 240 MHz
- 448 KB ROM for booting and core functions
- 520 KB SRAM for data and instructions
- 16 KB SRAM in RTC
- 4 MB SPI flash

Wi-Fi

- 802.11b / g / n
- Bitrate: 802.11n up to 150 Mbps
- A-MPDU and A-MSDU aggregation
- 0.4 µs guard interval support
- Center frequency range of operating channel: 2412 ~ 2484 MH

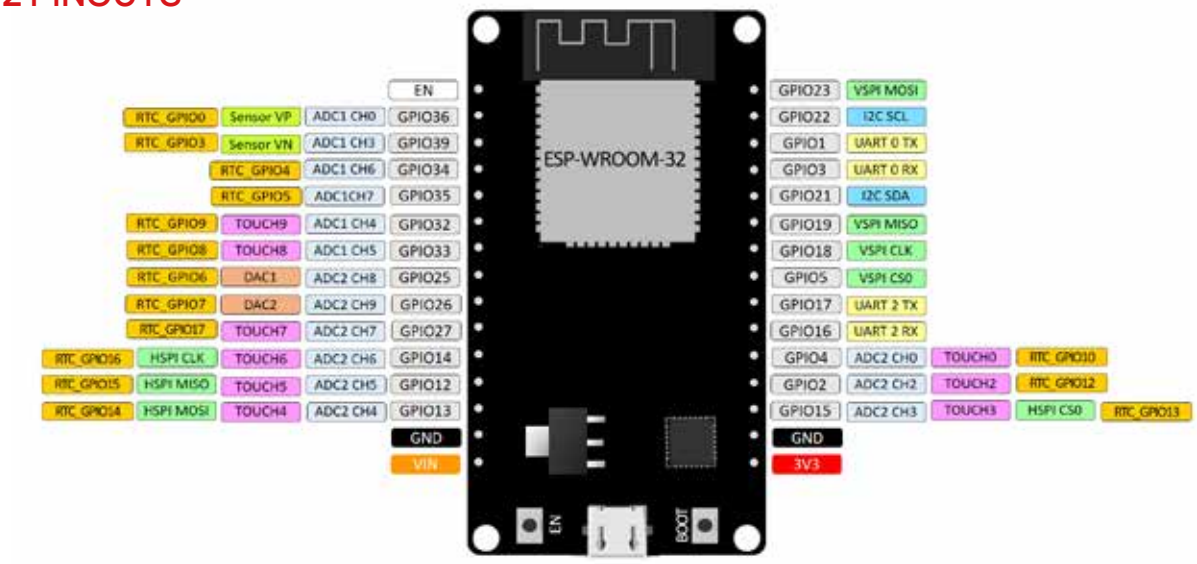
Bluetooth® / BLE

- Bluetooth V4.2 BR / EDR and Bluetooth LE specification
- Class-1, Class-2 and Class-3 transmitter
- AFH
- CVSD and SBC

Hardware

- **Interfaces:** UART, SPI, I2C, LED PWM, Motor PWM, IR, GPIO, Capacitive Touch Sensor, ADC, DAC, Two-Wire Automotive Interface, RGB LED, Sensors, Relay.
- **Communication Interface:** RS232, RS485 (Modbus RTU), USB, SPI, I2C, RF.
- **On Board Peripheral:** 8 Channel LED, 8 Channel Selection DIP Switch, RGB LED, 4x4 Matrix Keypad, 4 Key Menu Keypad, 3.3 to 5v Level Converter, RTC & EEPROM, DC Motor / Stepper Motor Driver, Relay, Buzzer, Temperature Sensor, 2 Channel Analog Pot, IR Sensor, Touch Sensor, Moisture Sensor, Sound Sensor, LDR, Fire Sensor, PIR Sensor, TSOP Sensor, OLED Display, 16 x 2 LCD Display,Seven Segment Display, RFID Reader,Xbee Adapter.

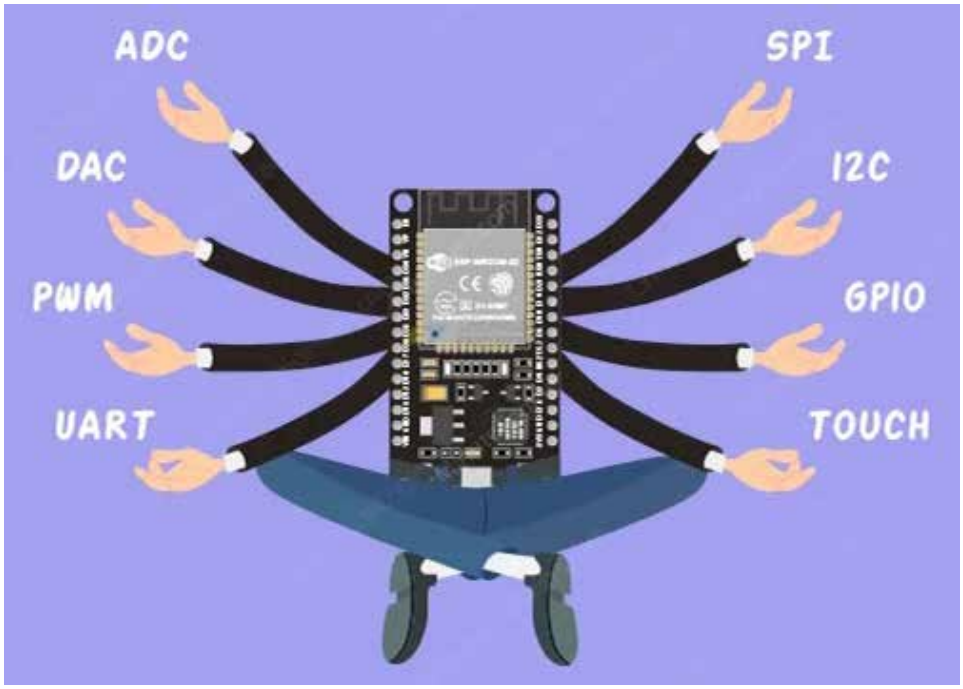
ESP32 PINOUTS



ESP32 Peripherals and I/O

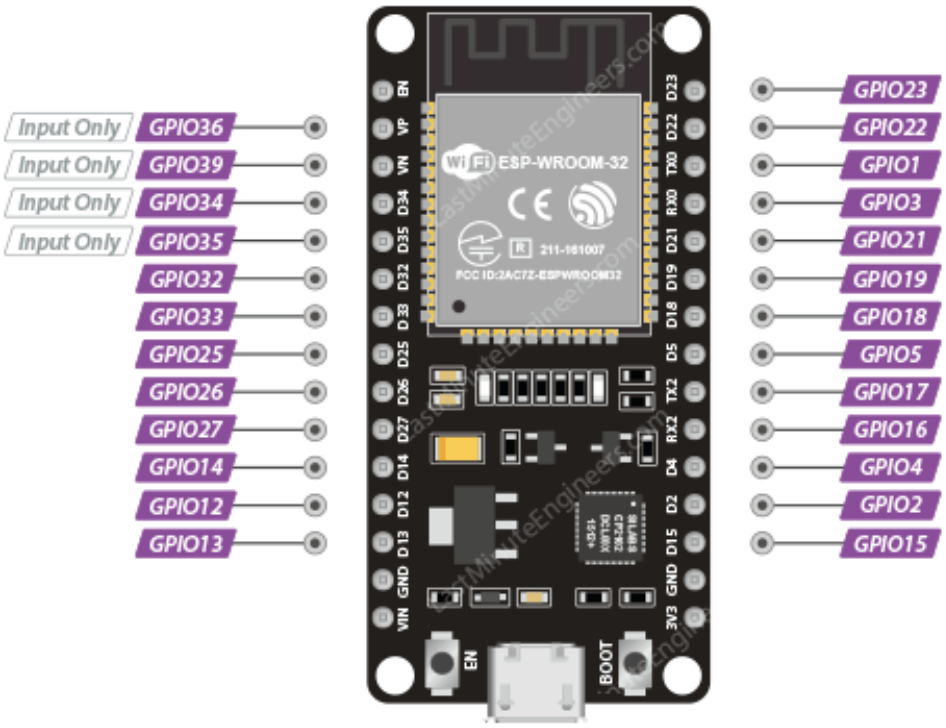
Although the ESP32 has total 48 GPIO pins, only 25 of them are broken out to the pin headers on both sides of the development board. These pins can be assigned to all sorts of peripheral duties, including

15 ADC Channels	15 channels of 12-bit SAR ADC's. The ADC range can be set, in firmware, to either 0-1V, 0-1.4V, 0-2V or 0-4V
2 UART Interfaces	2 UART interfaces. One is used to load code serially. They feature flow control and support IrDA too!
25 PWM Outputs	25 channels of PWM pins for dimming LEDs or controlling motors.
2 DAC Channels	8-bit DACs to produce true analog voltages
3 SPI & 1 I2C Interfaces	There are 3 SPI and 1 I2C interfaces to hook up all sorts of sensors and peripherals
9 Touch Pads	9 GPIOs feature capacitive touch sensing



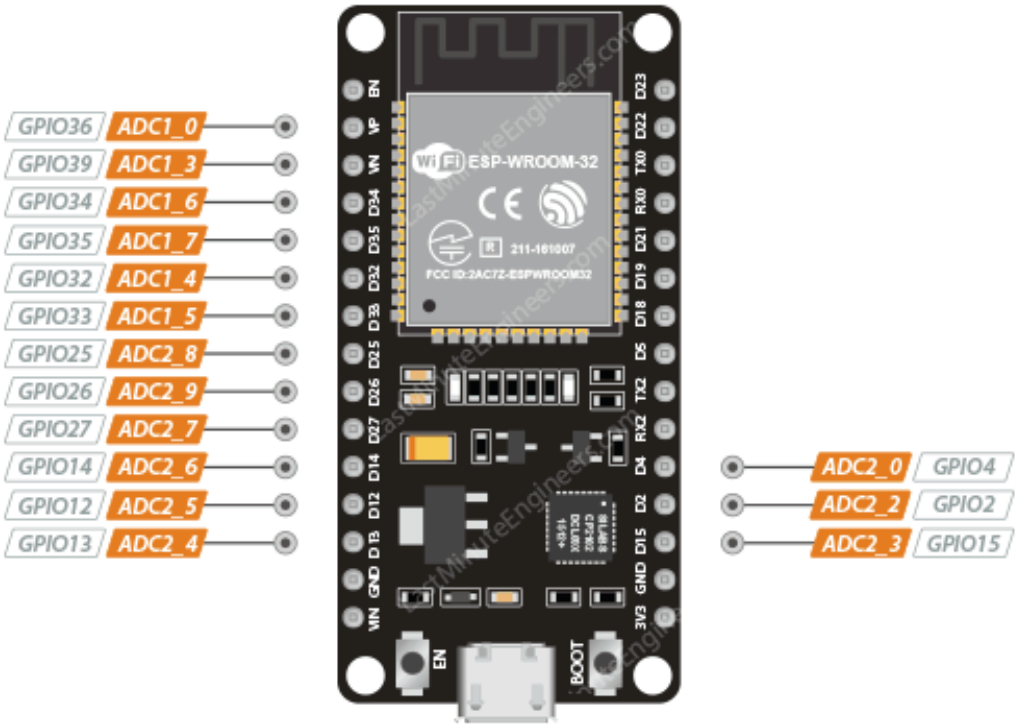
GPIO Pins

ESP32 development board has 25 GPIO pins which can be assigned to various functions programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance.



ADC Pins

The ESP32 has fifteen 12-bit ADC input channels. These are GPIOs that can be used to convert the analog voltage on the pin to a digital number.

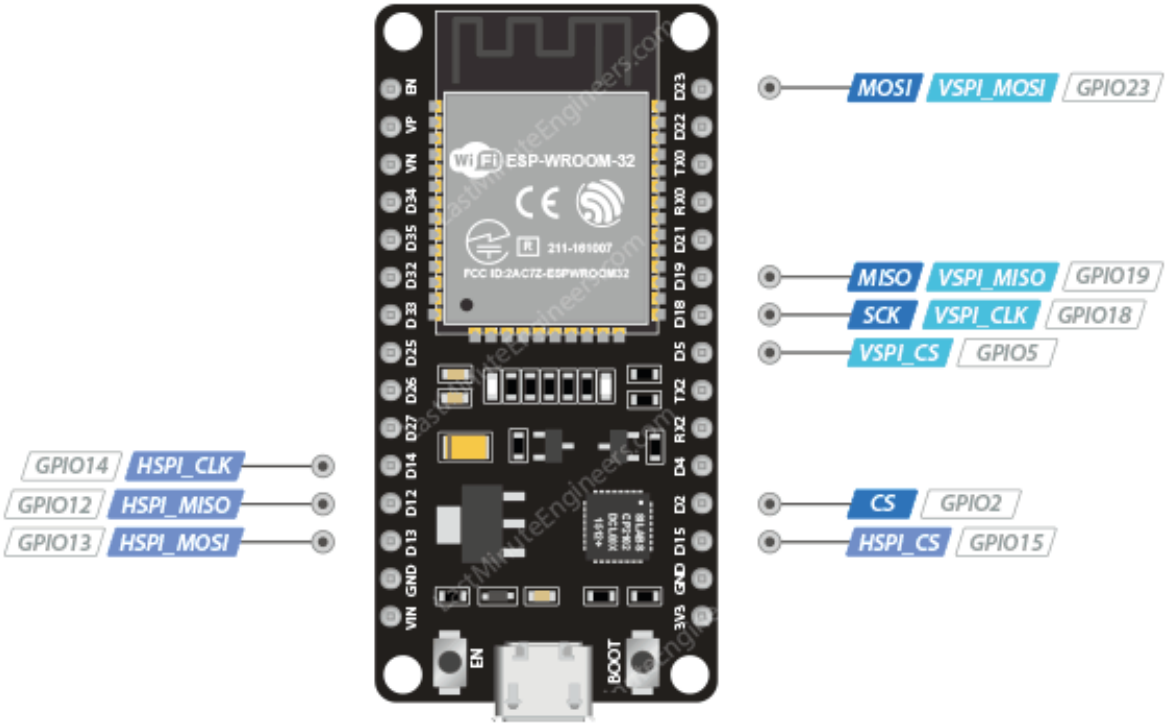


The ADC on the ESP32 is a 12-bit ADC meaning it has the ability to detect 4096 (212) discrete analog levels. In other words, it will map input voltages between 0 and the operating voltage 3.3V into integer values between 0 and 4095. For example, this yields a resolution between readings of: 3.3 volts / 4096 units or 0.0008 volts (0.8mV) per unit. You also have the ability to set the ADC resolution and ADC range of your channels in code.

SPI Pins

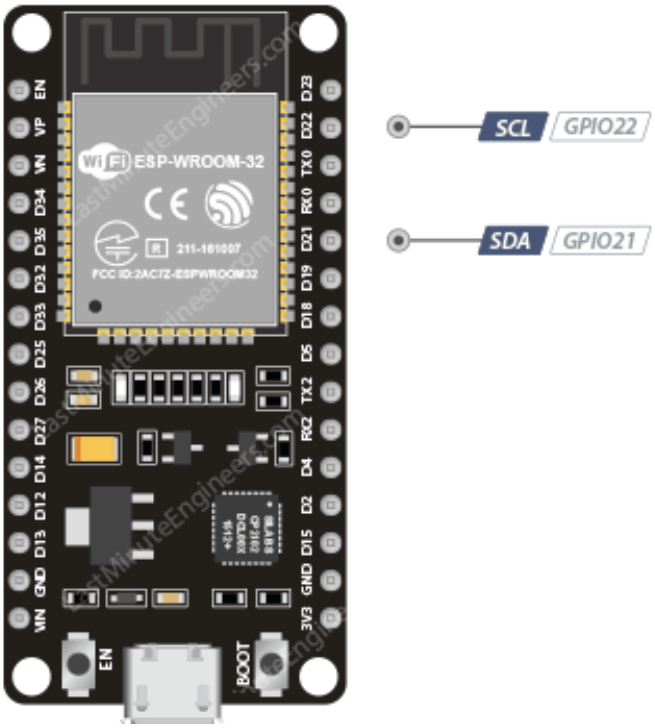
SPI Pins ESP32 features three SPIs (SPI, HSPI and VSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:

- 4 timing modes of the SPI format transfer
- Up to 80 MHz and the divided clocks of 80 MHz
- Up to 64-Byte FIFO



I2C Pins

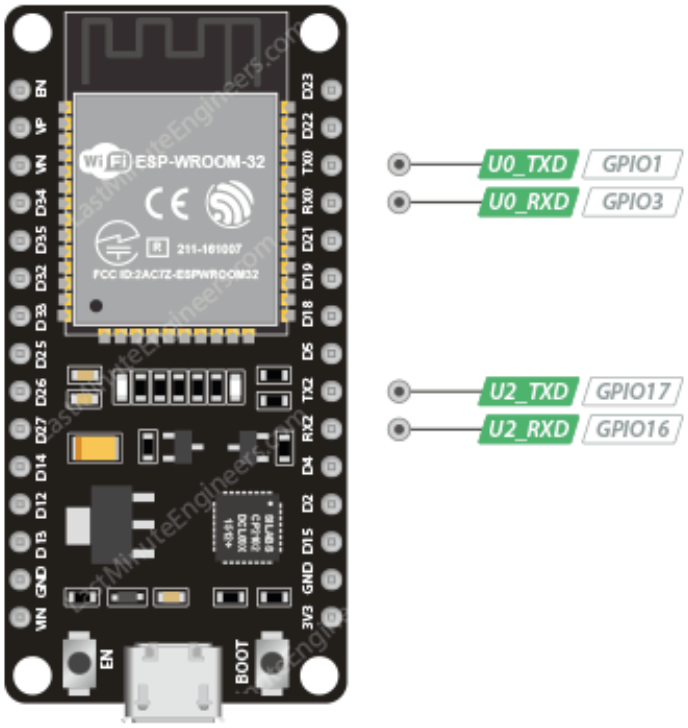
The ESP32 has a single I2C bus that allows you to connect up to 112 sensors and peripherals. The SDA and SCL pins are, by default, assigned to the following pins. However, you can bit-bang the I2C protocol on any GPIO pins with wire.begin (SDA, SCL) command.



UART Pins

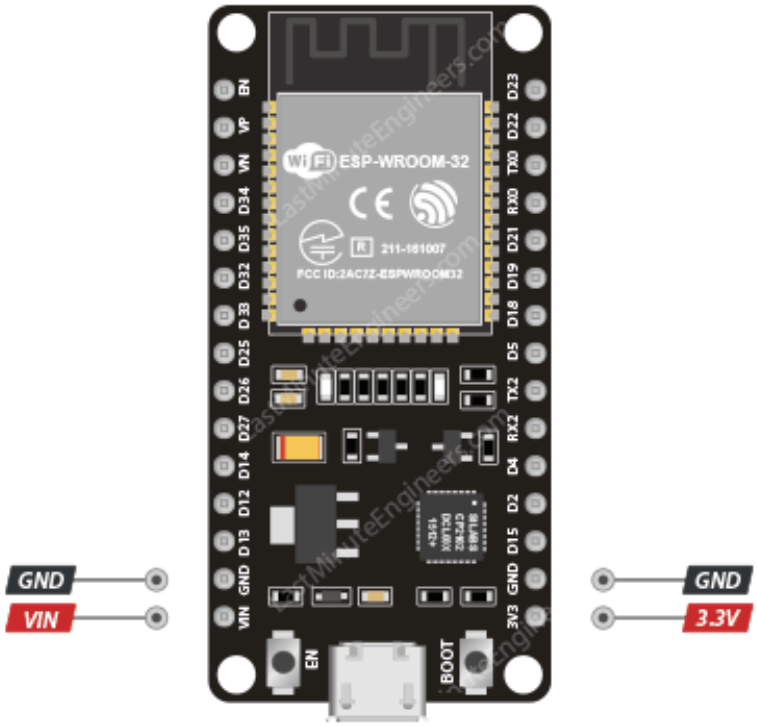
ESP32 has 2 UART interfaces, i.e. UART0 and UART2, which provide asynchronous communication (RS232 and RS485) and IrDA support and communicate at up to 5 Mbps.

- UART0 pins are connected to the USB-to-Serial converter and are used for flashing and debugging. Therefore it is not recommended to use the UART0 pins.
- UART2, on the other hand, are additional Serial1 pins, and are not connected to the USB-to-Serial converter. This means that you can use them to connect to UART-devices such as GPS, fingerprint sensor, etc.



Power Pins

There are two power pins viz. VIN pin & 3.3V pin. The VIN pin can be used to directly supply the ESP32 and its peripherals, if you have a regulated 5V voltage source. The 3.3V pin is the output of an on-board voltage regulator. This pin can be used to supply power to external components. GND is a ground pin of ESP32 development board.



3. LCD 2 * 16 Display

An LCD screen is an electronic display module that uses liquid crystal to produce a visible image. The 16X2 LCD display is a very basic module commonly used in DIYs and circuits. The 16 X 2 translates display 16 characters per line in 2 such lines. In this LCD each character is displayed in a 5 X 7 pixel matrix.

4. RGB LED

An RGB LED is a combination of 3 LEDs in just one package:

- 1 X RGB LED

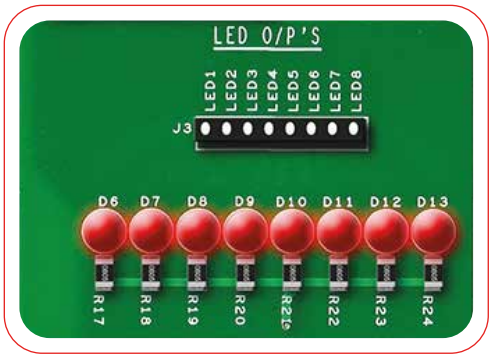
You can produce almost any color by combining those three colors.

5. OLED Display

OLED Display 0.96" module is an OLED monochrome 128 X 64 dot matrix display module with Grove 4pin I2C Interface. Comparing to LCD, OLED screens are more competitive, which has a number of advantages such as high brightness, self-emission, high contrast ratio, slim / thin outline, wide viewing angle, wide temperature range and low power consumption.

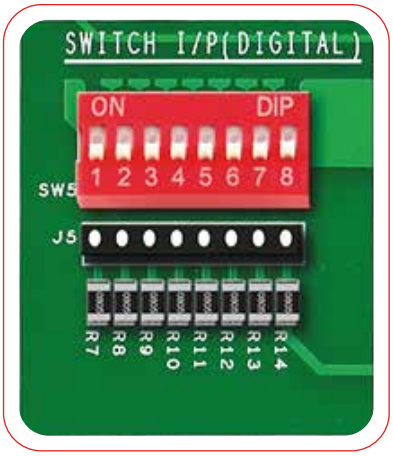
6. Eight Channel LED

Eight LED has available to check digital outputs.



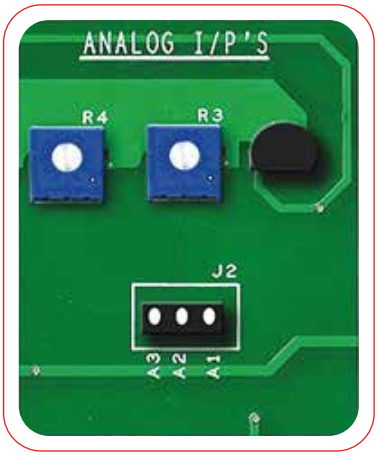
8. Analog Inputs

3 following Analog input devices are in this kit TrimPot1, TrimPot2 and LM35 temperature sensor given to learn how to read ADC values using The Petal Netalla IoT Kit.



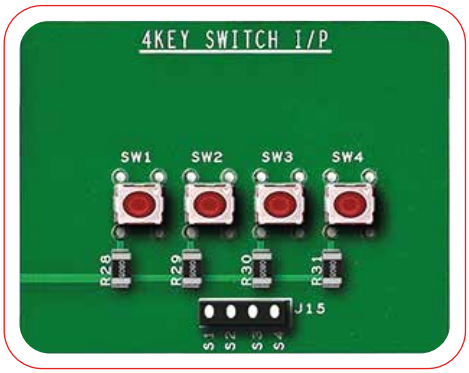
7. Eight Channel Dip Switch

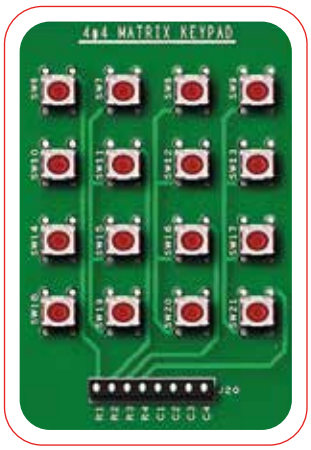
An eight channel dip switch is given to understand the digital logic manually by using the switch.



9. Four Keys Switch Inputs

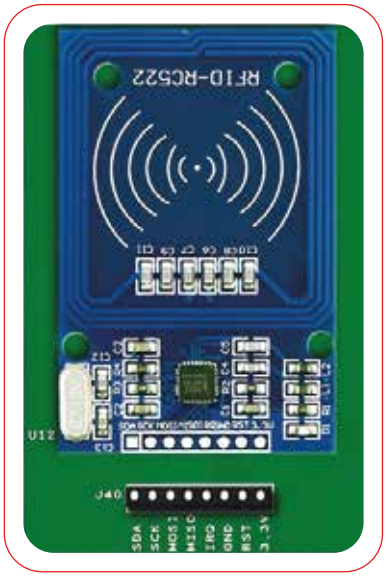
4 key switches use to learn about digital logic inputs, Menu key functions and direction control.





10. 4*4 Matrix Keypad

4*4 keypad uses to learn matrix operations and how to inset alphanumeric values.



11. RFID Reader / Writer

The RC522 RFID reader module is designed to create a 13.56MHz electromagnetic field and communicate with RFID tags (ISO 14443A standard tags). The reader can communicate with a microcontroller over a 4-pin SPI with a maximum data rate of 10 Mbps. It also supports communication over I2C and UART protocols. It is used in Smart login, Secured gate and door operating systems and smart rental applications.

12. Relay Driver

Relays are used for isolating a low-voltage circuit from a high-voltage circuit. They are used for controlling multiple circuits. They are also used as automatic change over. Microprocessors use relays to control a heavy electrical load. And it is used in Home automation, Industrial automation, and Automobile applications.

13. RTC module

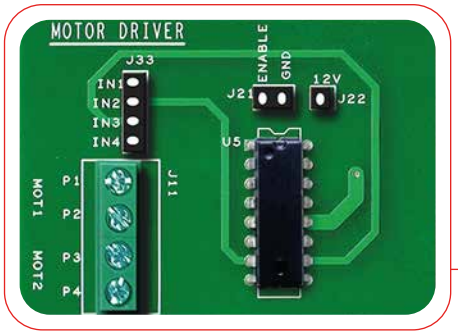
The purpose of an RTC or a real-time clock is to provide precise times and dates which can be used for various applications.

14. EEPROM

EEPROM is used to store external data.

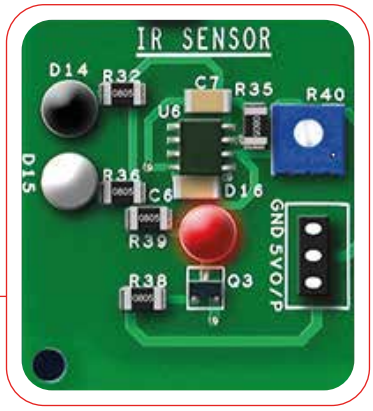
15. Four Digit Seven Segment Display

4 Digits 7 Segment Led Display Module works by I2C Bus you can control it using only 2 wires, leaving more pins available on your MicroController to connect other things. It is used in RTC displays, Traffic countdown displays, and Banking token systems.



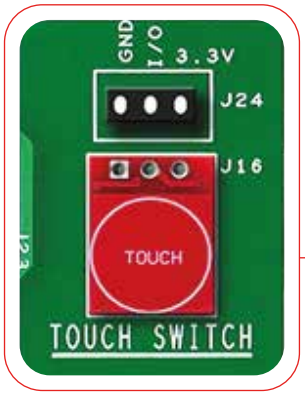
16. Motor Driver

L293D Motor Driver Module is a medium power motor driver it's provided to learn to drive DC Motors and Stepper Motors. It is used for DC motor and stepper motor driving control in Industrial automation, Motor based door operating system.



17. IR Sensor

IR sensor provides learn about obstacle detection by using The Petal Netalla IoT Kit. It is used for IR imaging devices, remote sensing, flame monitors, night vision devices, rail safety, etc



18. Touch Switch

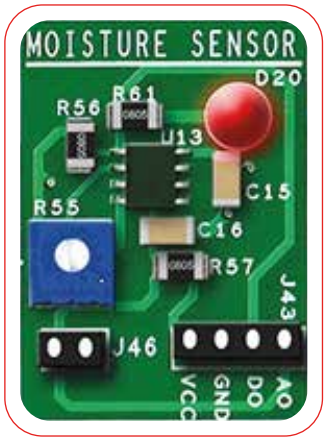
A capacitive touch sensor switch is a digital logic state output deliverable device, it is used to control load or devices through relay by using The Petal Netalla IoT Kit.

19. Light Sensor

Light-dependent resistor(LDR) is used as a light sensor, It is used to control lighting devices in absence of light. It is used for the street light control system.

20. Flame Sensor

A thermistor is a type of resistor whose resistance is strongly dependent on temperature Here by Thermistor is used as a fire sensor to learn heat and fire detection and temperature measurement by The Petal Netalla IoT Kit.

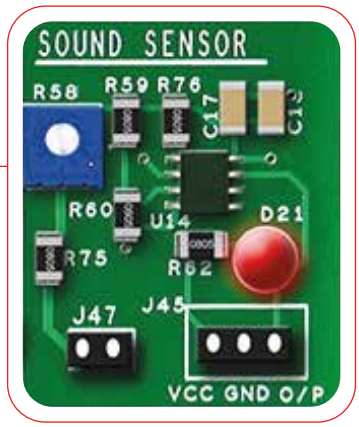


21. Moisture Sensor

Soil moisture sensors measure the volumetric water content in the soil, it is used to learn about smart agriculture farming by The Petal Netalla IoT Kit.

22. Sound Sensor

A condenser Microphone is used as a sound sensor, it's used to detect the sound. It used for Monitoring systems such as burglar alarms and door alarm.

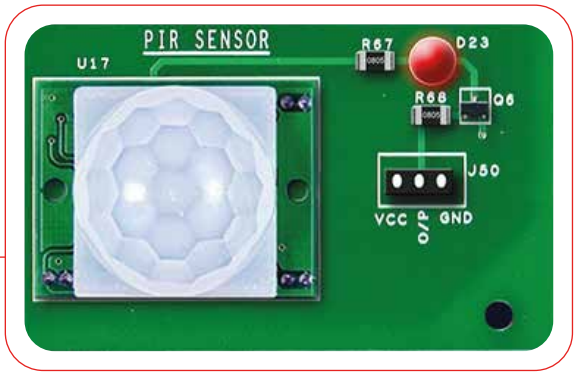


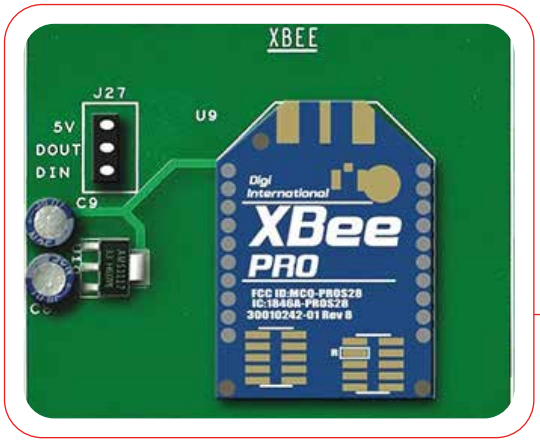
23. IR Receiver

The TSOP Sensor is a miniaturized receiver for infrared remote-control systems, it uses to read frequency from different infrared remotes and I can control multiple loads and devices by The Petal Netalla IoT Kit.

24. Human Detection Sensor

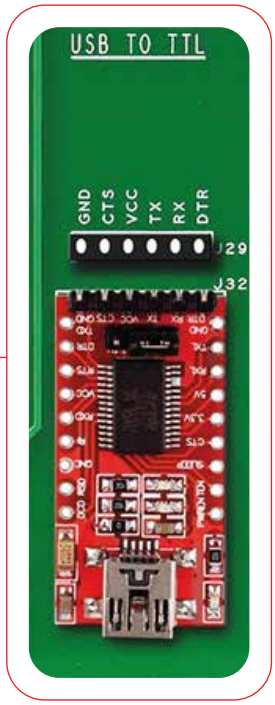
PIR sensors allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensors range. It is used for the Automated door operating system, Automated lighting system.





25. Xbee Adaptor

Xbee modems are one of the easiest ways to create a wireless point-to-point or mesh network, their uses for data transformation, BMS, CCMS and EMS.

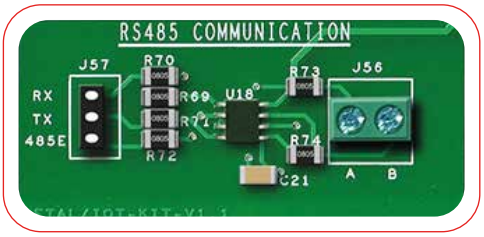


26. USB to TTL Converter

FTDI USB to TTL serial converter modules are used for general serial applications. It is popularly used for communication to and from microcontroller development boards and other MCU, which do not have USB interfaces.

27. RS232 Converter

RS232 is one of the standard protocols in telecommunication which is used for serial communication of data. It is the process of connecting signals between data terminal equipment (DTE). For example, file servers, routers and application servers, such as a modem.

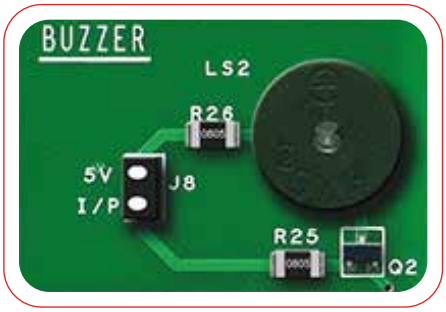


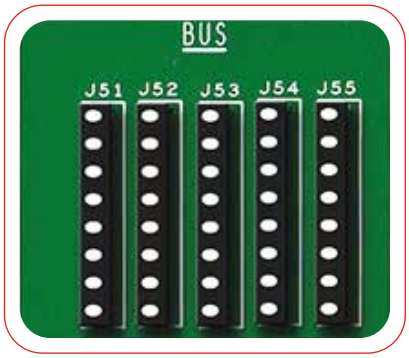
28. RS485 Converter

This is a module that allows the TTL interface of the microcontroller to be transferred to the RS485 module. It is generally used for industrial automation.

29. Buzzer

A buzzer or beeper is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric, Typical uses of buzzers include alarm devices and timers.



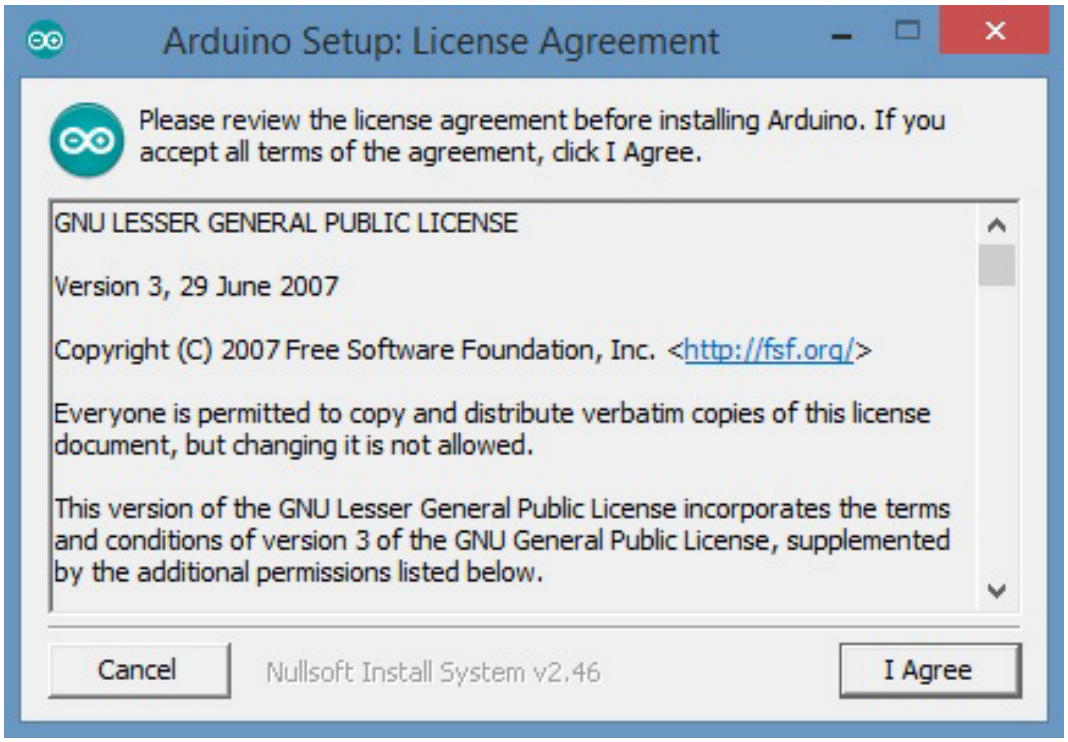


30. BUS Connection Port

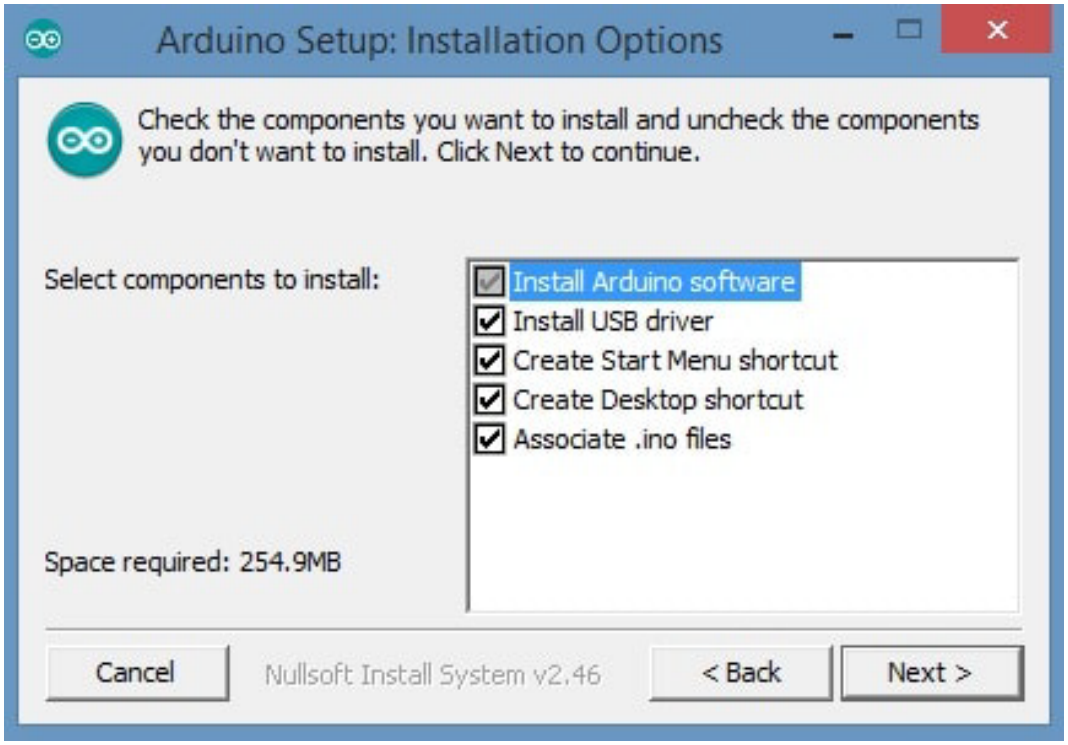
Bus connection ports are used to spread the single pinout by multi-way.

Installing the **ARDUINO IDE**

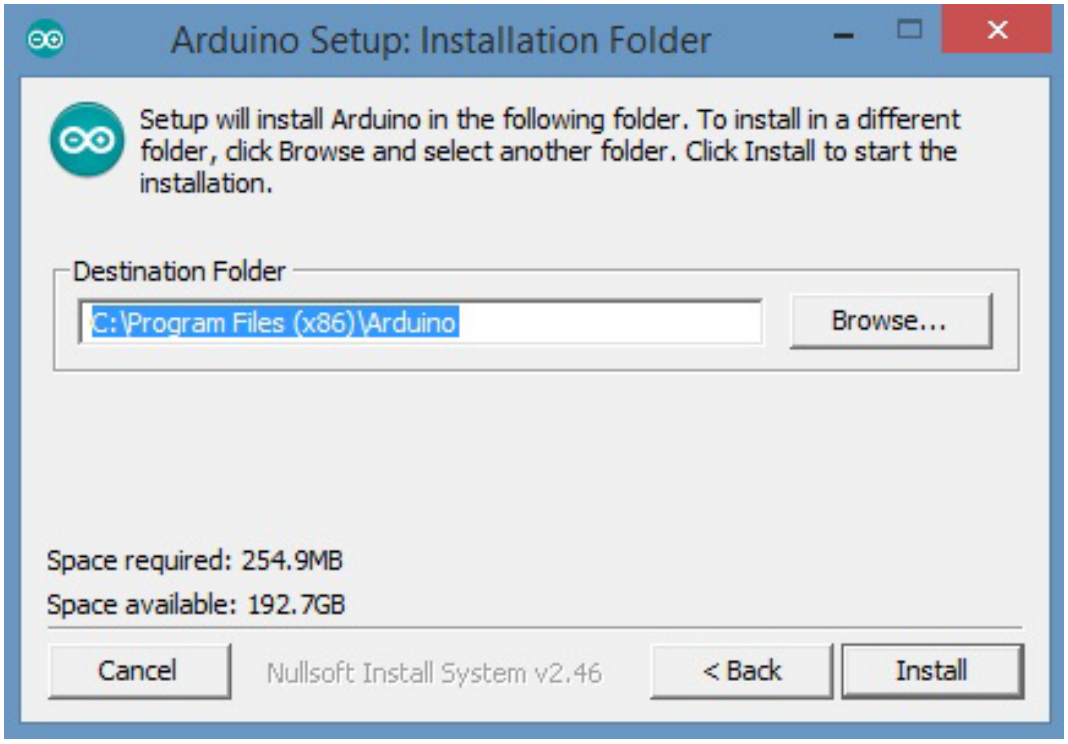
1. Visit http://www.arduino.cc/en/main/software to download the latest Arduino IDE version for your computer's operating system. There are versions for Windows, Mac, and Linux systems. At the download page, click on the "Windows Installer" option for the easiest installation
2. Save the .exe file to your hard drive
3. Open the .exe file
4. Click the button to agree the licensing agreement.



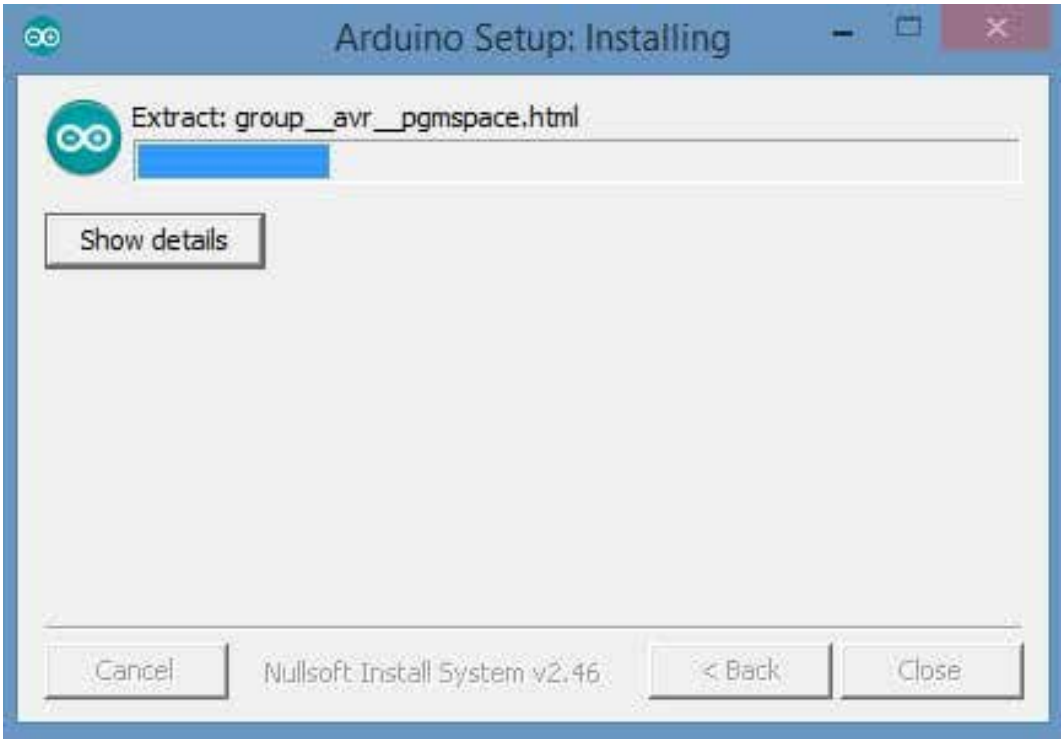
5. Decide which components to install, and then click "Next"



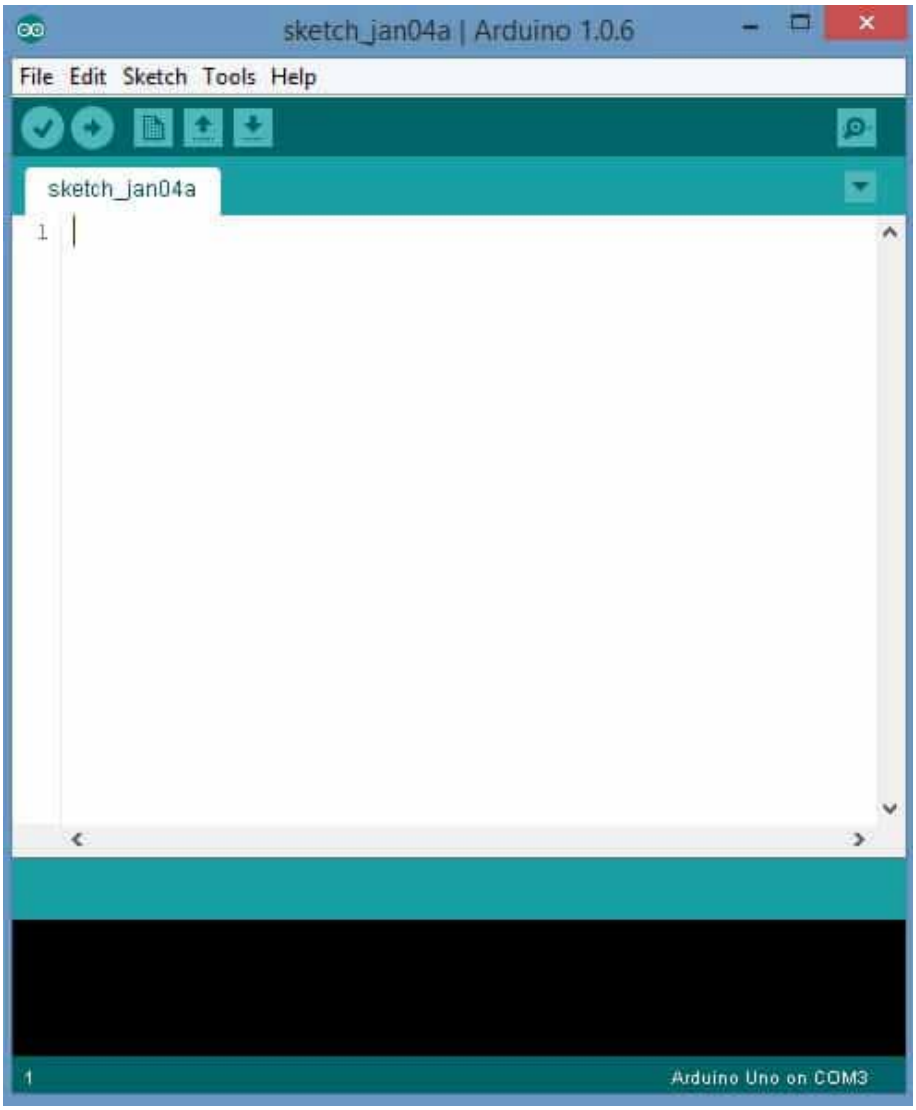
6. Select which folder to install the program to, then click "Install"



7. Wait for the program to finish installing, and then click "Close"



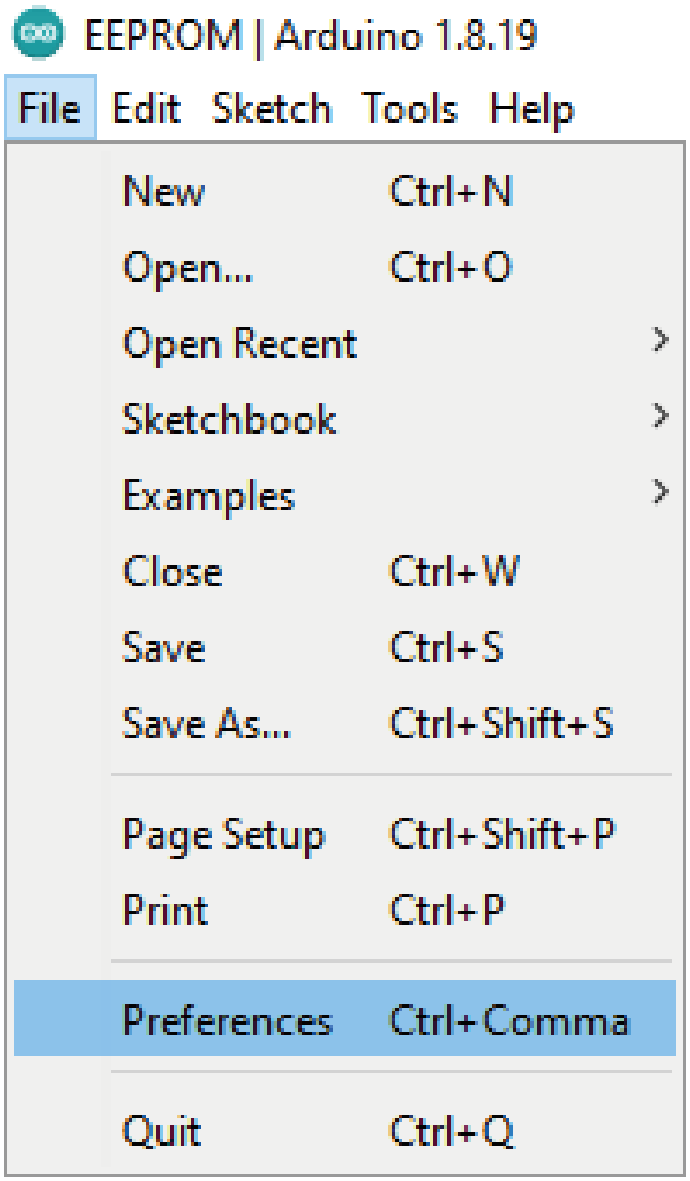
8. Now find the Arduino shortcut on your Desktop and click on it.
   The IDE will open up and you'll see the code editor.



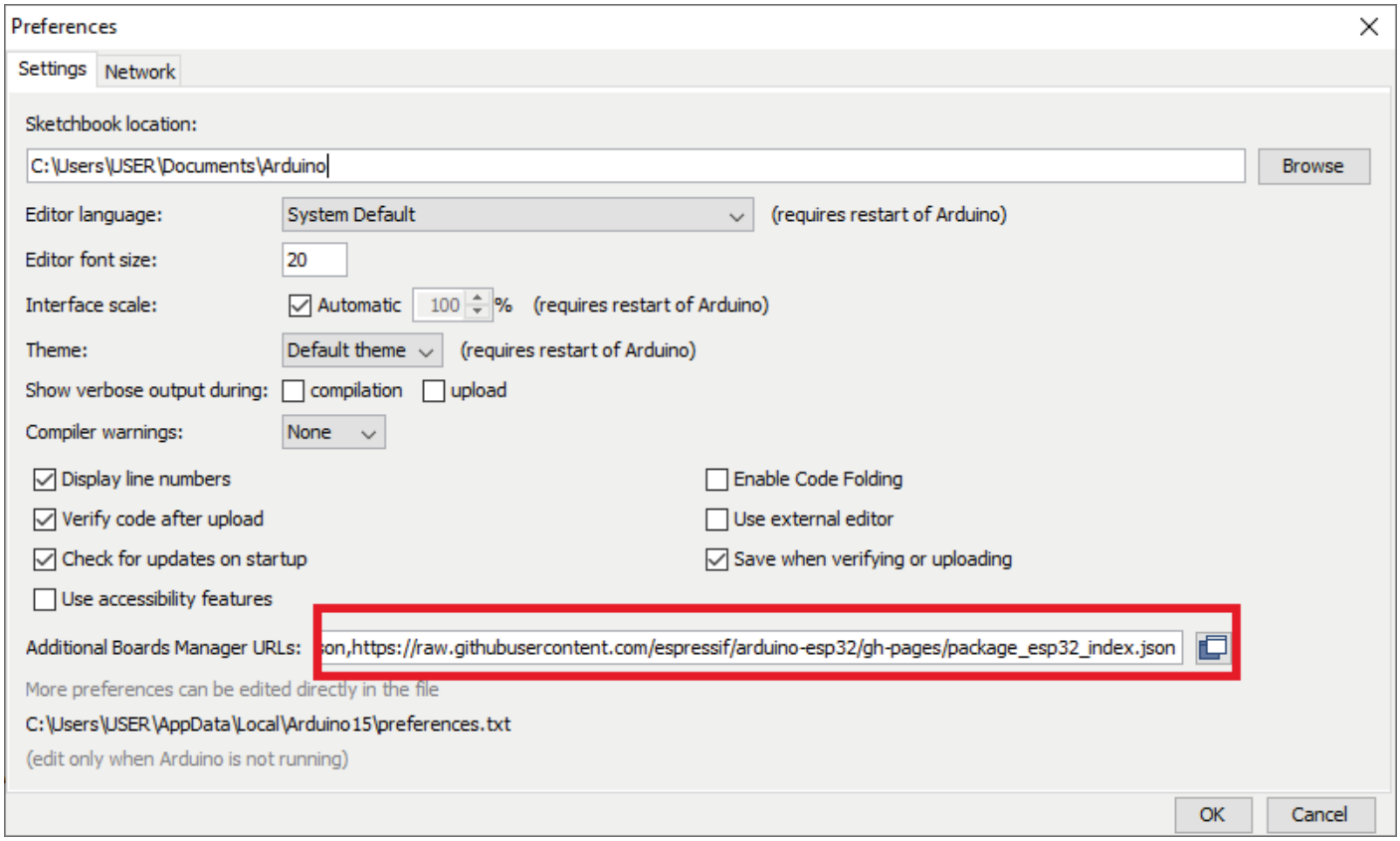
Installing ESP32 Add-on in **ARDUINO IDE**

To install the ESP32 board in your Arduino IDE, follow these next instructions:

1. In your Arduino IDE, go to File› Preferences



2. Enter the following into the "Additional Board Manager URLs" field
   https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
   Then, click the "OK" button.

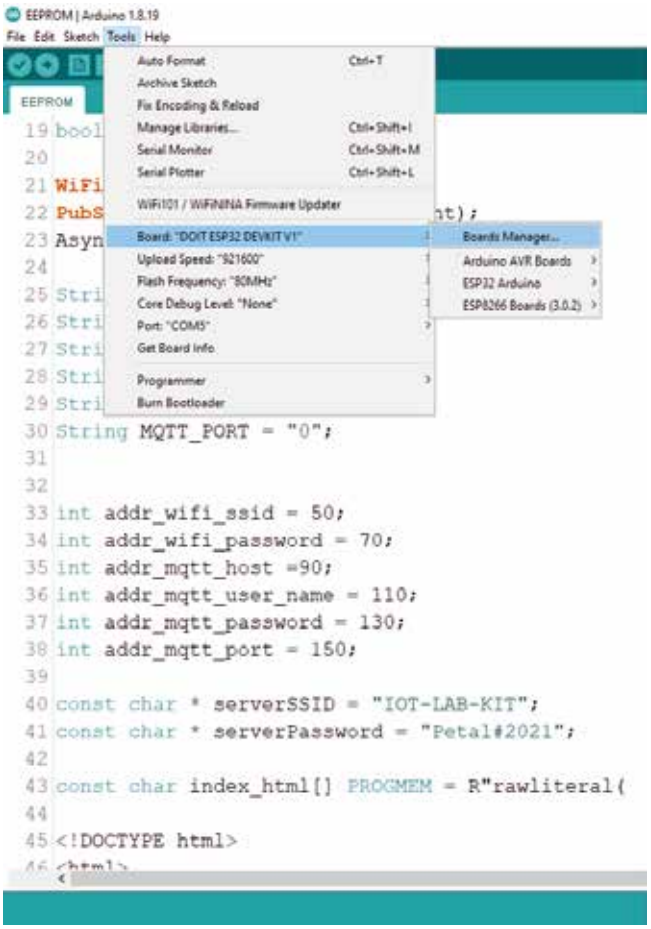


Note: if you already have the ESP8266 boards URL, you can separate the URLs with a comma as follows:

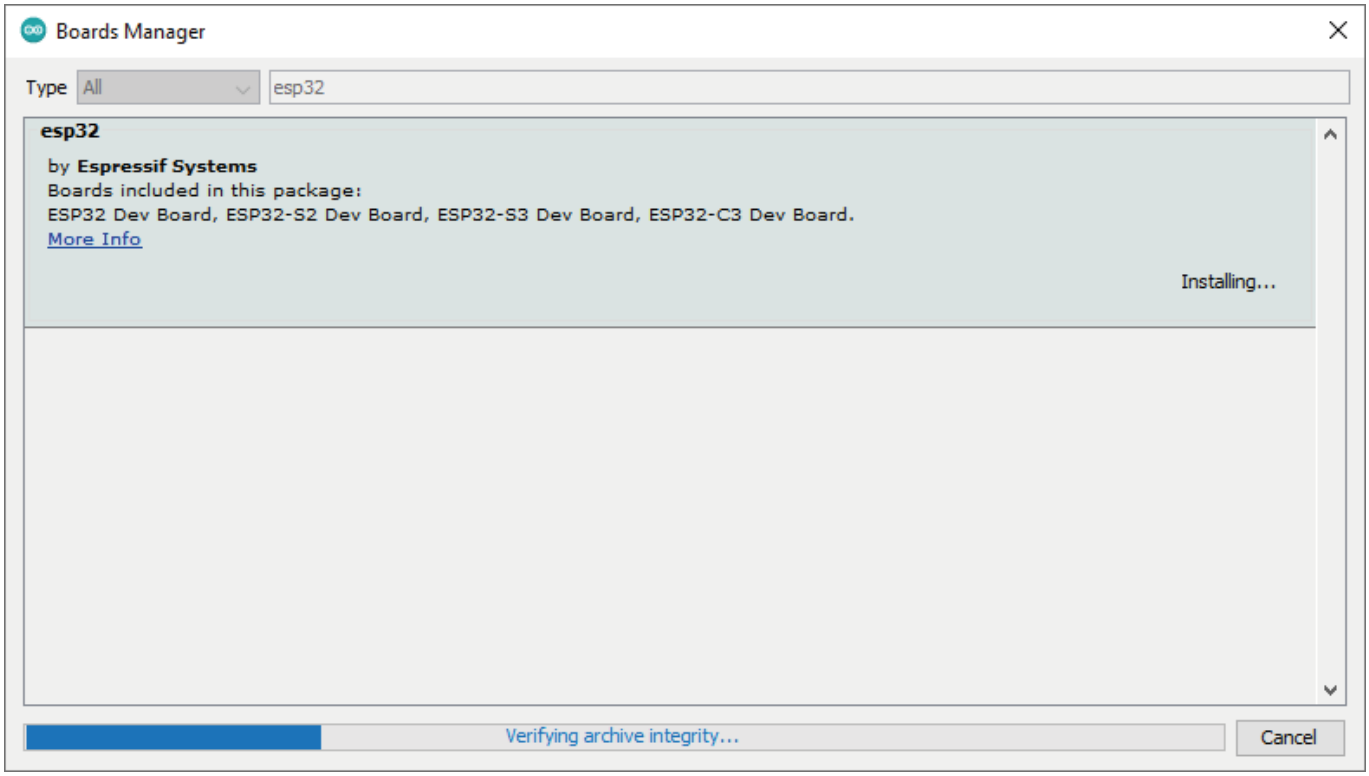
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

http://arduino.esp8266.com/stable/package_esp8266com_index.json

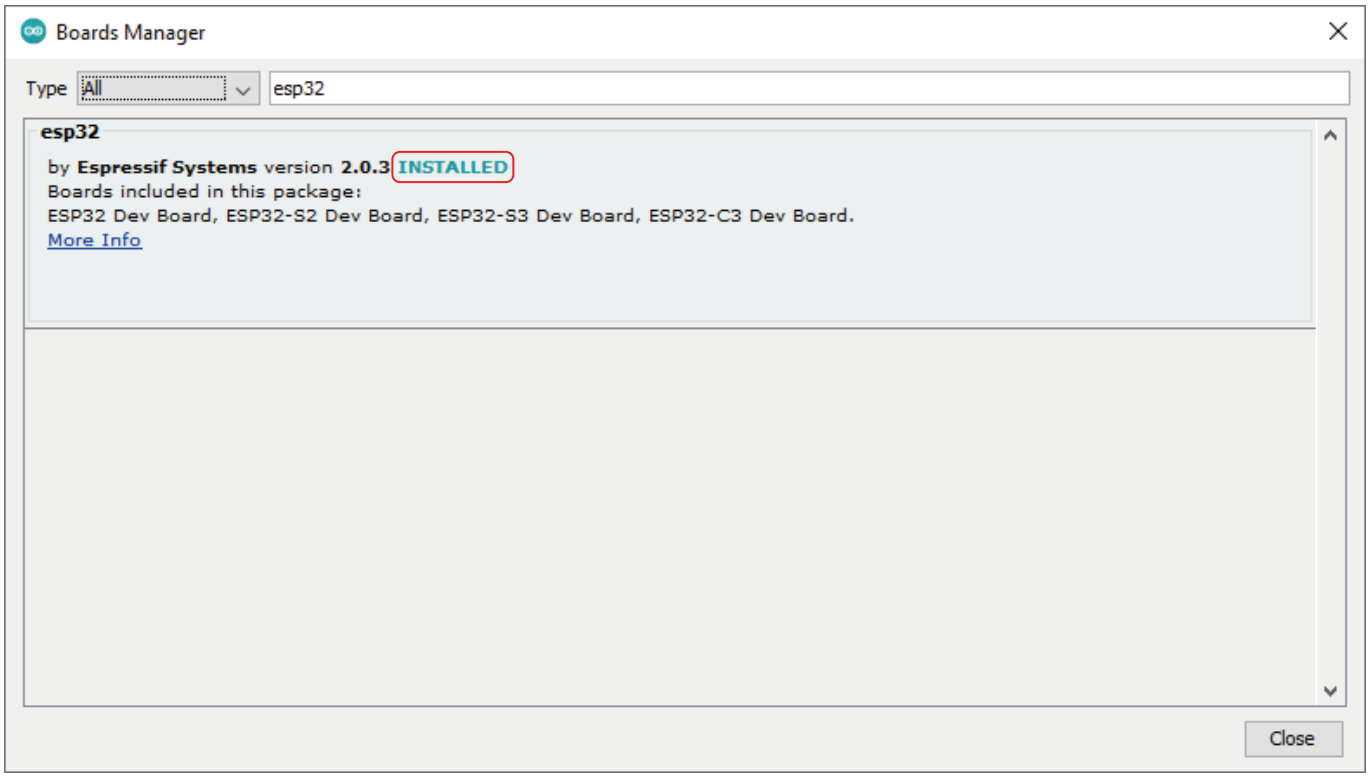
3. Open the Boards Manager. Go to Tools > Board > Boards Manager...



4. Search for ESP32 and press install button for the "ESP32 by Espressif Systems"



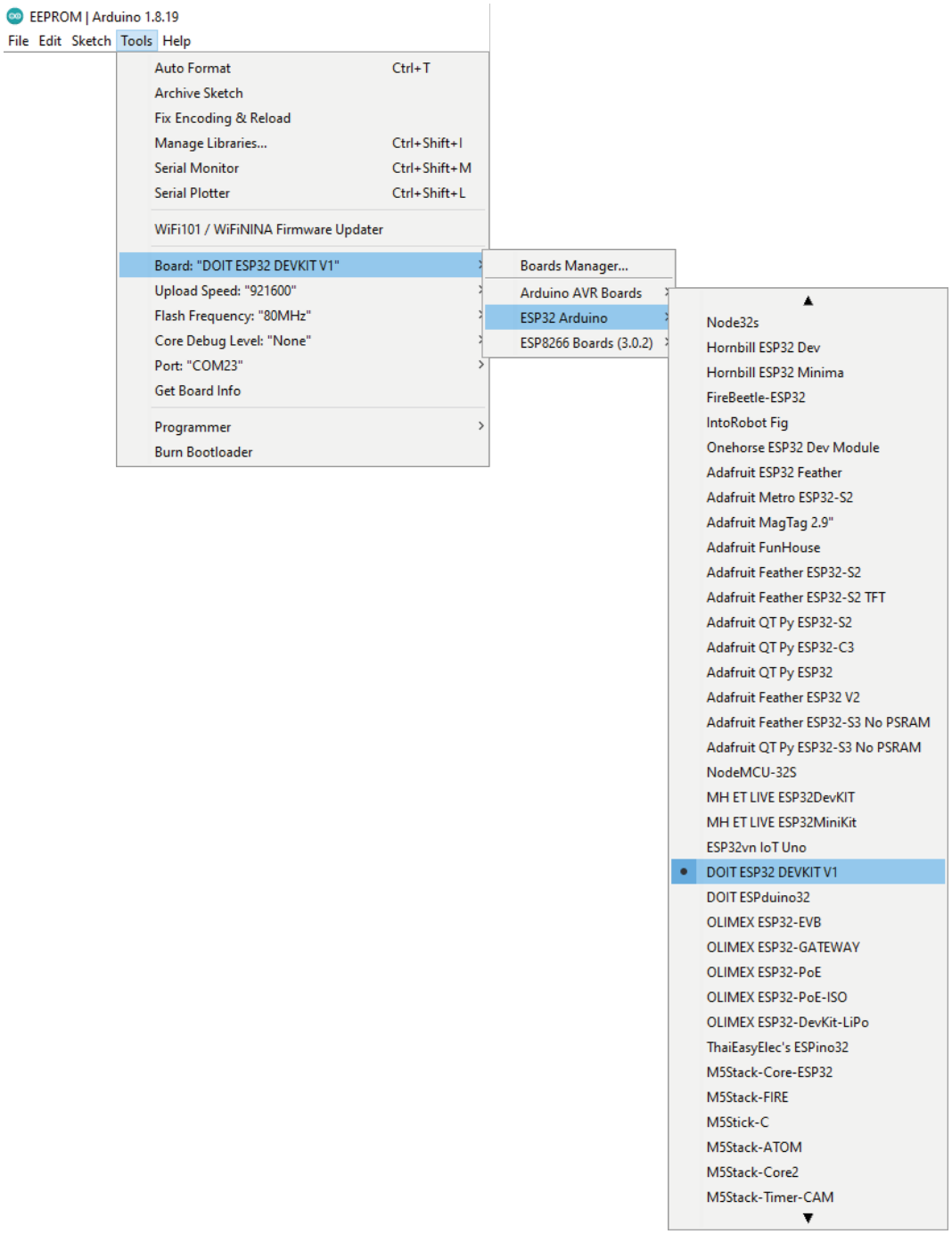
5. That's it. It should be installed after a few seconds.



Testing the Installation

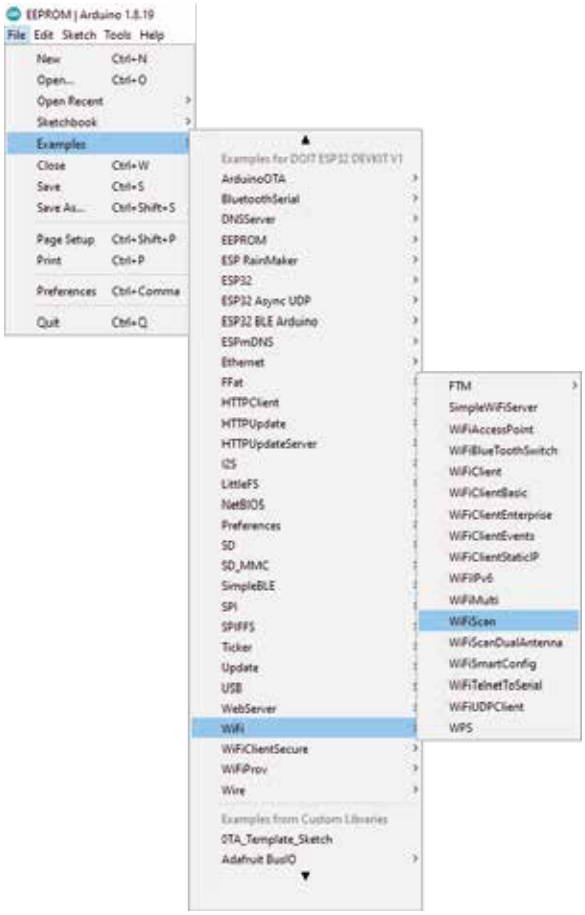
Plug the ESP32 board to your computer. With your Arduino IDE open, follow these steps.

1. Select your Board in Tools > Board menu (it's the DOIT ESP32 DEVKIT V1)

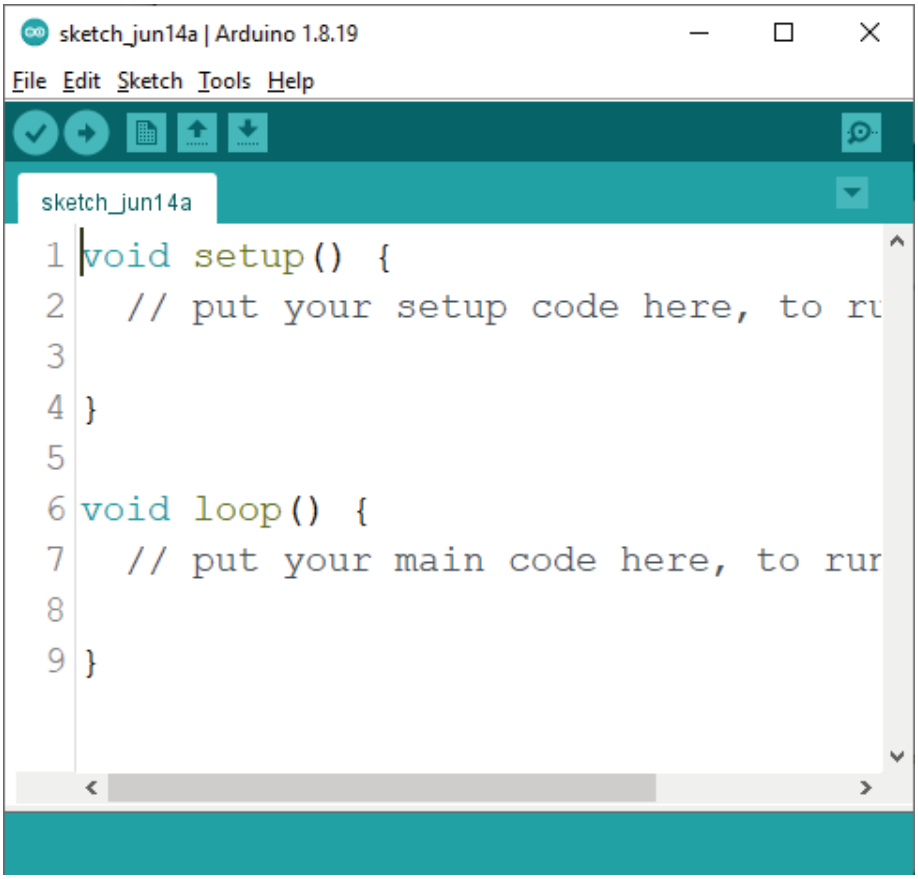


2. Select the Port (if you don't see the COM Port in your Arduino IDE, you need to install the CP210x USB to UART Bridge VCP Drivers)

3. Open the following example under File › Examples › Wi-Fi (ESP32) › WiFiScan



4. A new sketch opens in your Arduino IDE:



5. Press the Upload button in the Arduino IDE. Wait a few seconds while the code compiles and uploads to your board.

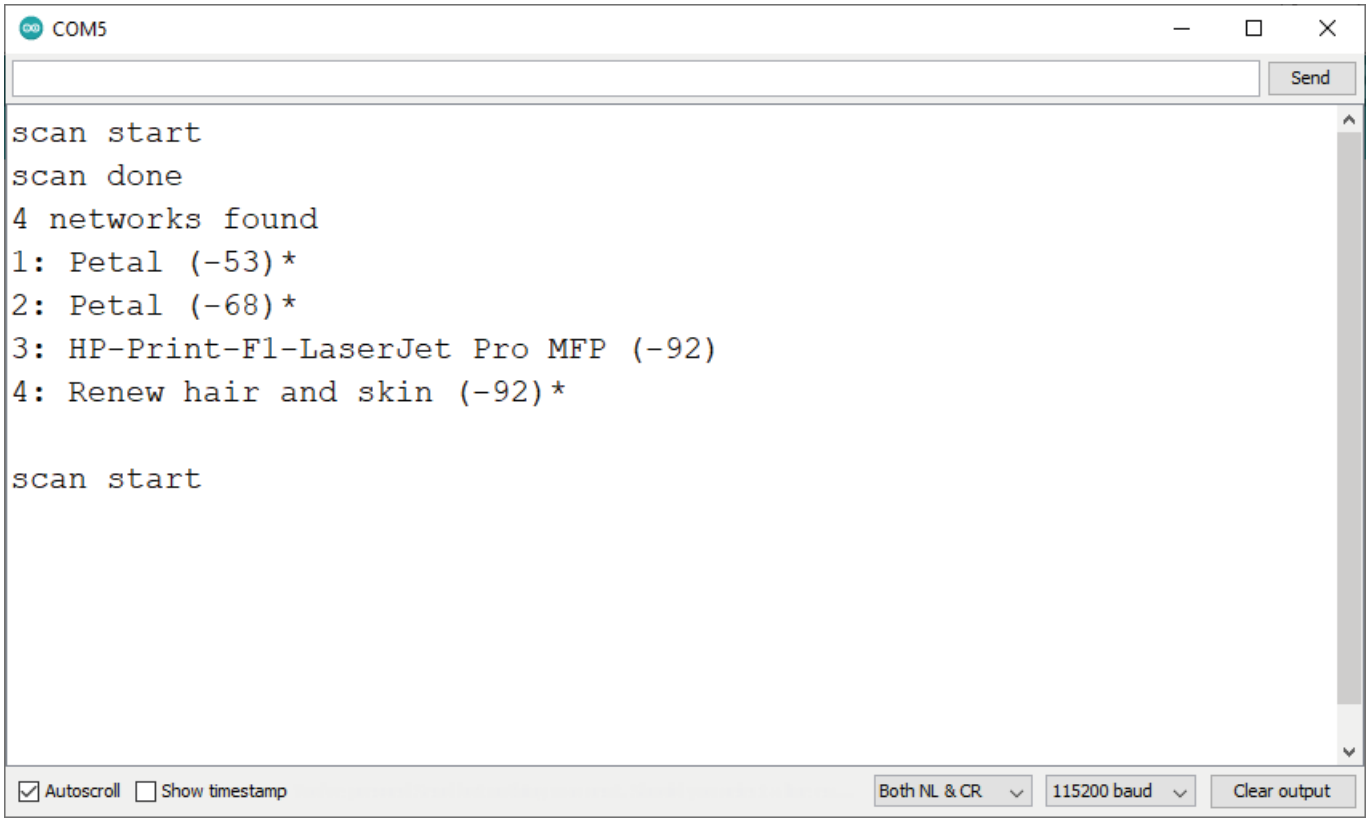
6. If everything went as expected, you should see a "Done uploading" message.

```
Done uploading.
Writing at 0x000a54b9... (96 %)
Writing at 0x000aaddf... (100 %)
Wrote 644608 bytes (416939 compressed) at 0x00010000 in 7.0 seconds (effective 740.6 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

7. Open the Arduino IDE Serial Monitor at a baud rate of 115200

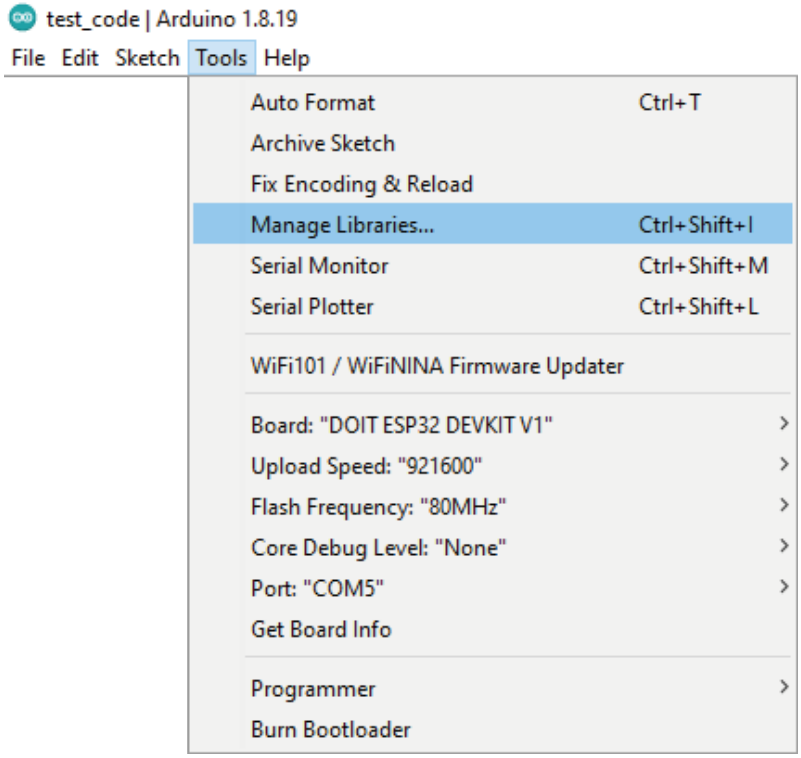
8. Press the ESP32 on-board Enable button and you should see the networks available near your ESP32



Install library on Arduino IDE

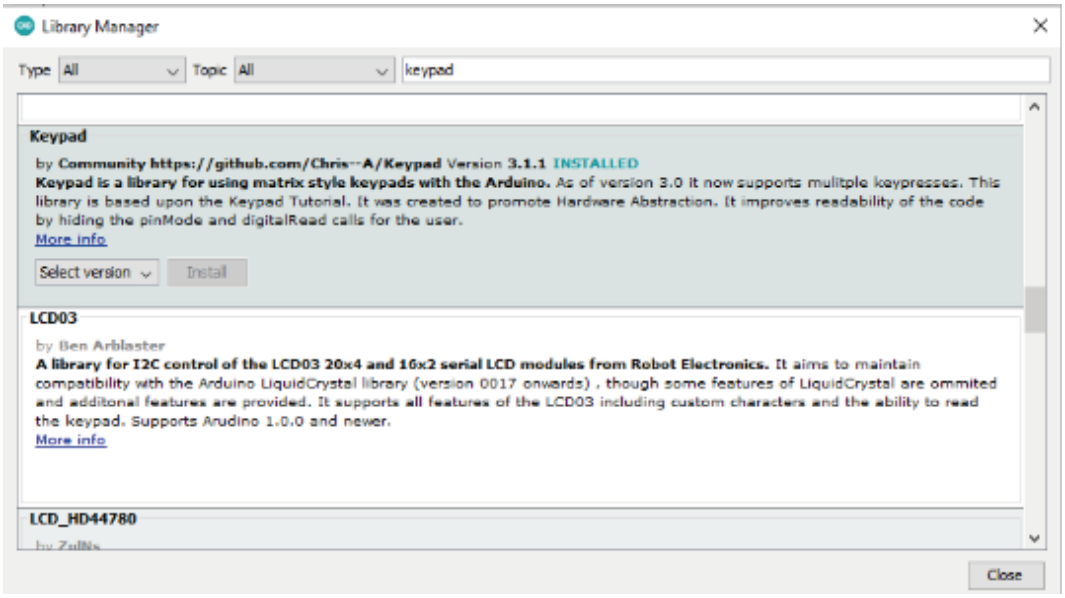
There are three ways to install Arduino Library.

- On Arduino IDE, Go to Tools >Mange Libraries



- Search "library name", then find the correct library by checking library name the author of the library. The below image shows the example of "Keypad" library

- Click Install button to install keypad library

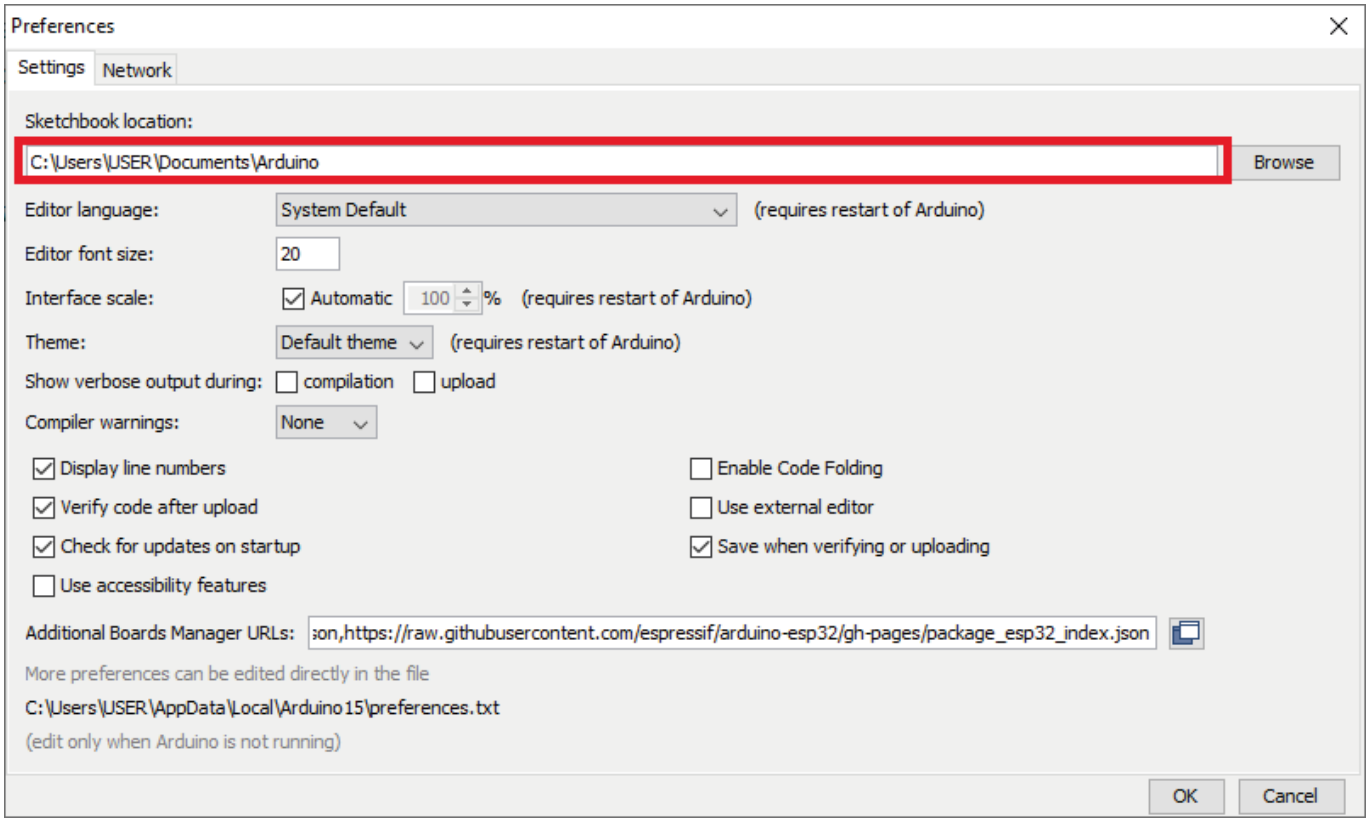


Install .zip library

- Download the .zip library to your PC
- Open Arduino IDE
- On Arduino IDE, Go to Sketch >Include Library Add .Zip library...
- Select the downloaded .zip file, and then the library will be installed

Install library manually

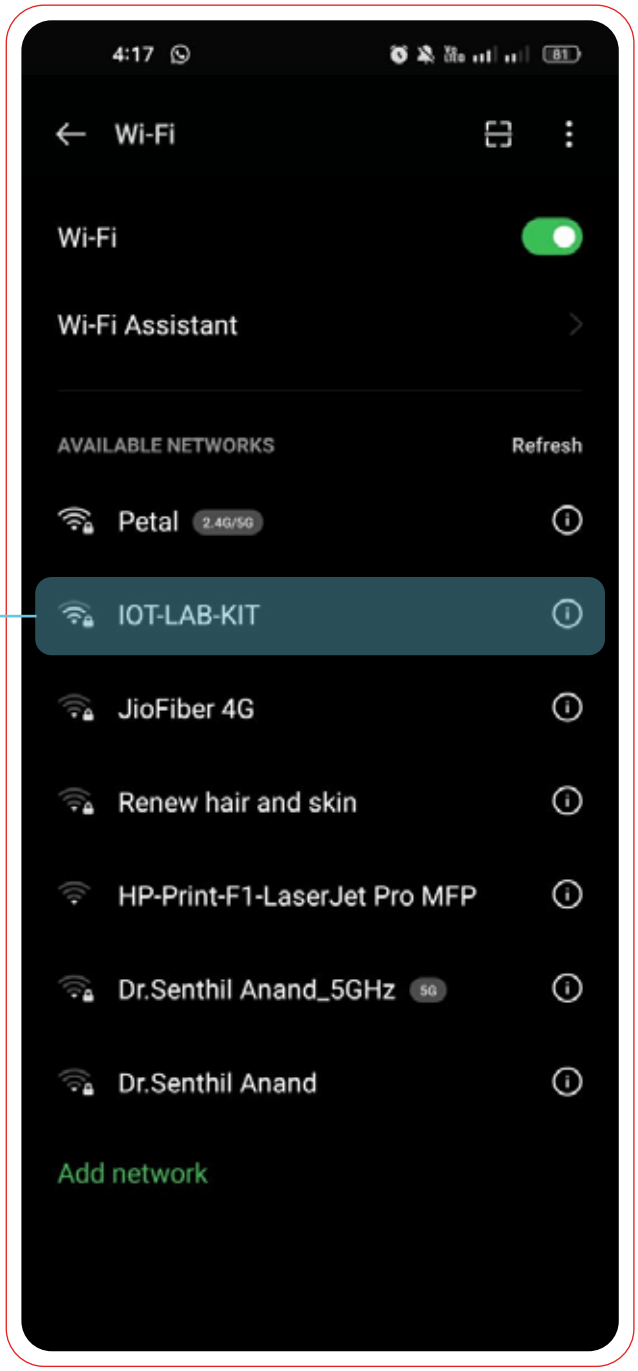
- Download the .zip library to your PC
- Unzip .zip library
- Find the directory of your Arduino libraries by navigating to File>Preferences. You will see Sketchbook location as below image



- Copy that directory and go to that directory. For example C:\Users\USER\Documents\Arduino, and then go to C:\Users\USER\Documents\Arduino\libraries subdirectory
- Copy the unzipped folder to your Arduino the above libraries location
- Close Arduino IDE and then open it

Note: Library files are given in the dashboard. You can download it and installed in the Arduino IDE.

How to add Wi-Fi SSID & Wi-Fi password



Step 1:

Enable the Wi-Fi on your Mobile & Scan the available network.

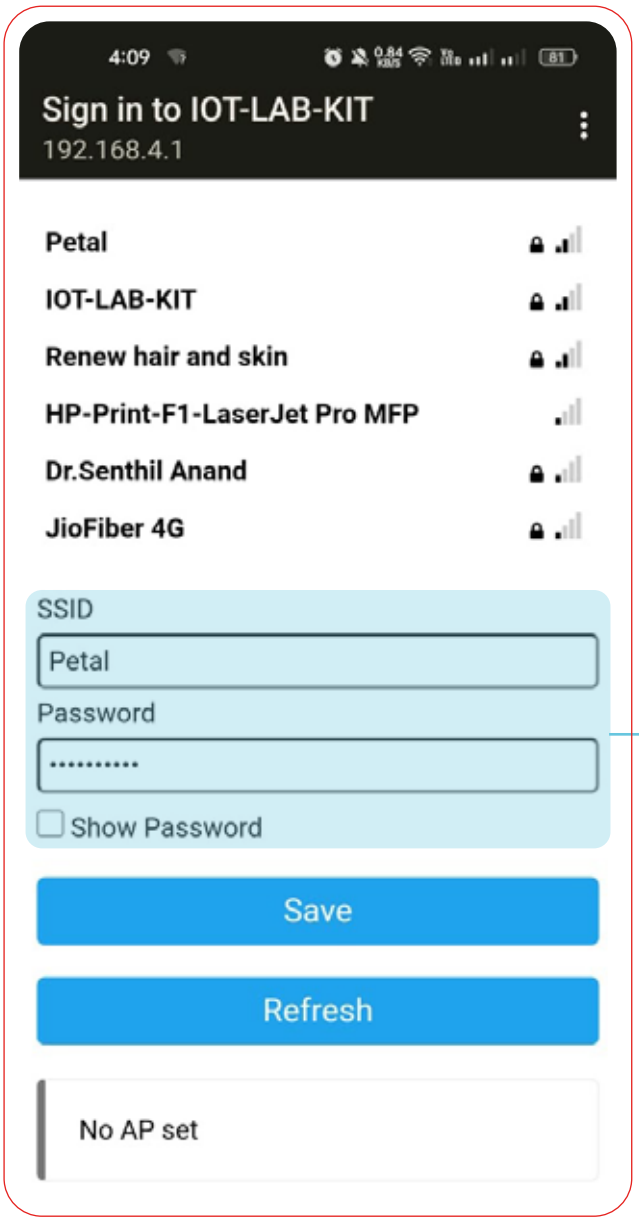
Select the **IOT-LAB-KIT** and enter the password as **12345678**

Step 2:

If the password credentials are valid, then control will navigate to the following page

Step 3:

Select Configure Wi-Fi

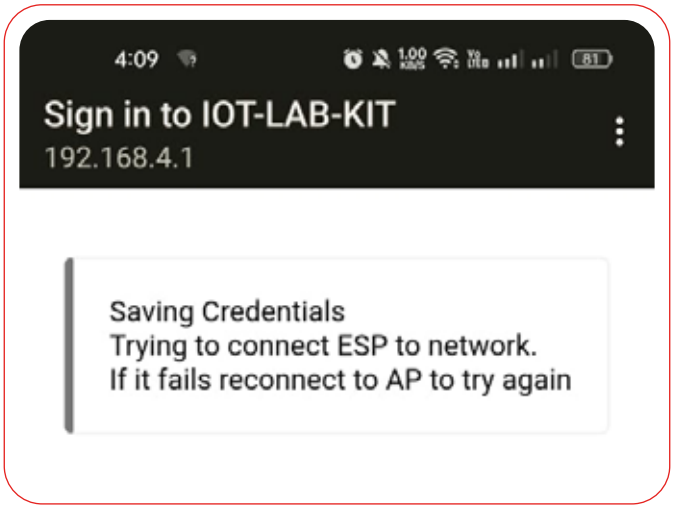


Step 4:

Select your Wi-Fi & Enter the Wi-Fi Password

Step 5:

Once the configuration process completed successfully, the message will be displayed as below.



How to Signup

Step 1:

Enter the following URL and Click on the Login / Register

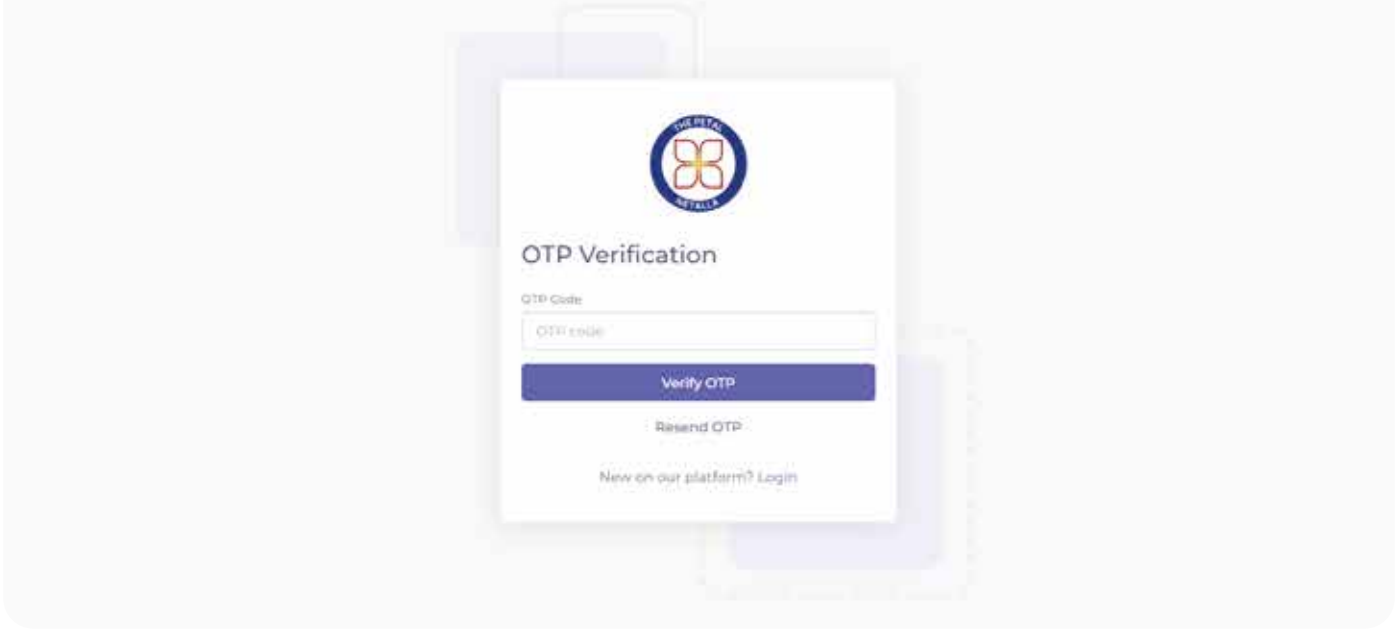
https://petalnetalla.in

Step 2:

Click on Create an account

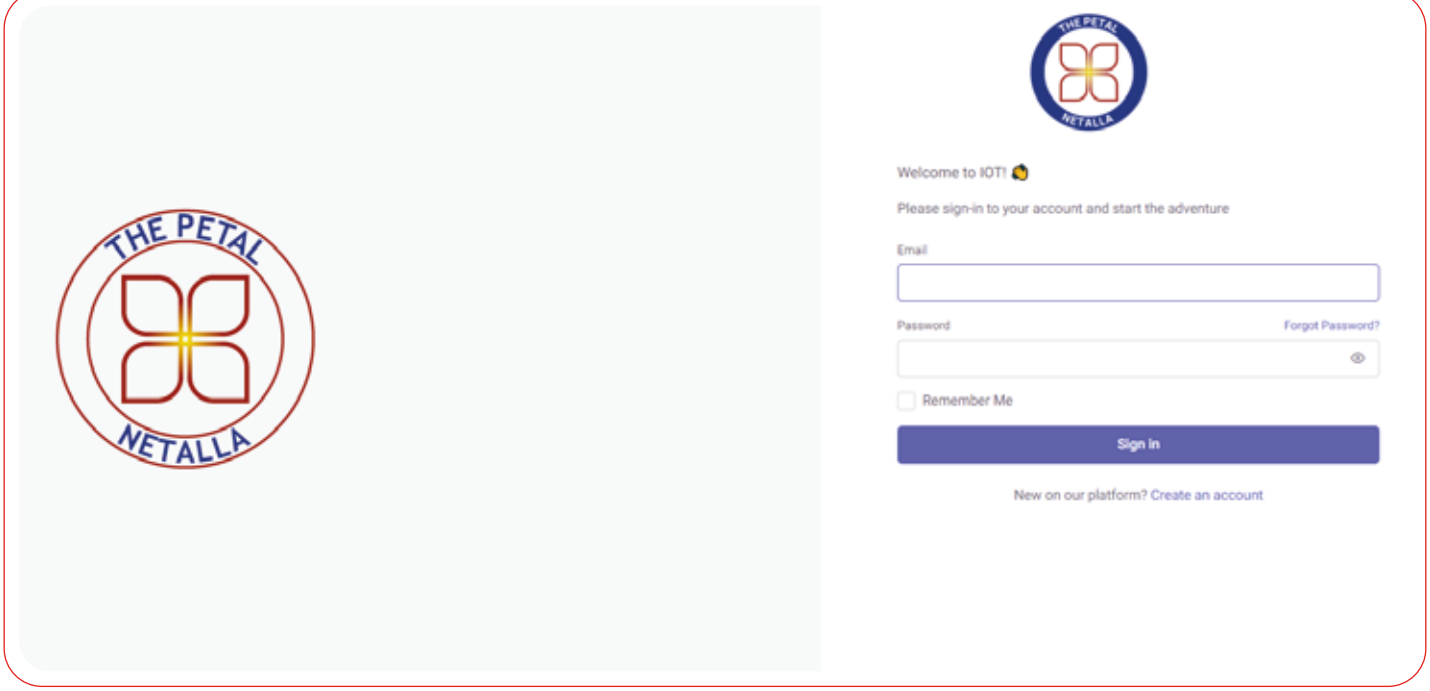
Step 3:

Once the details were entered, you may get OTP on your Email for verification.



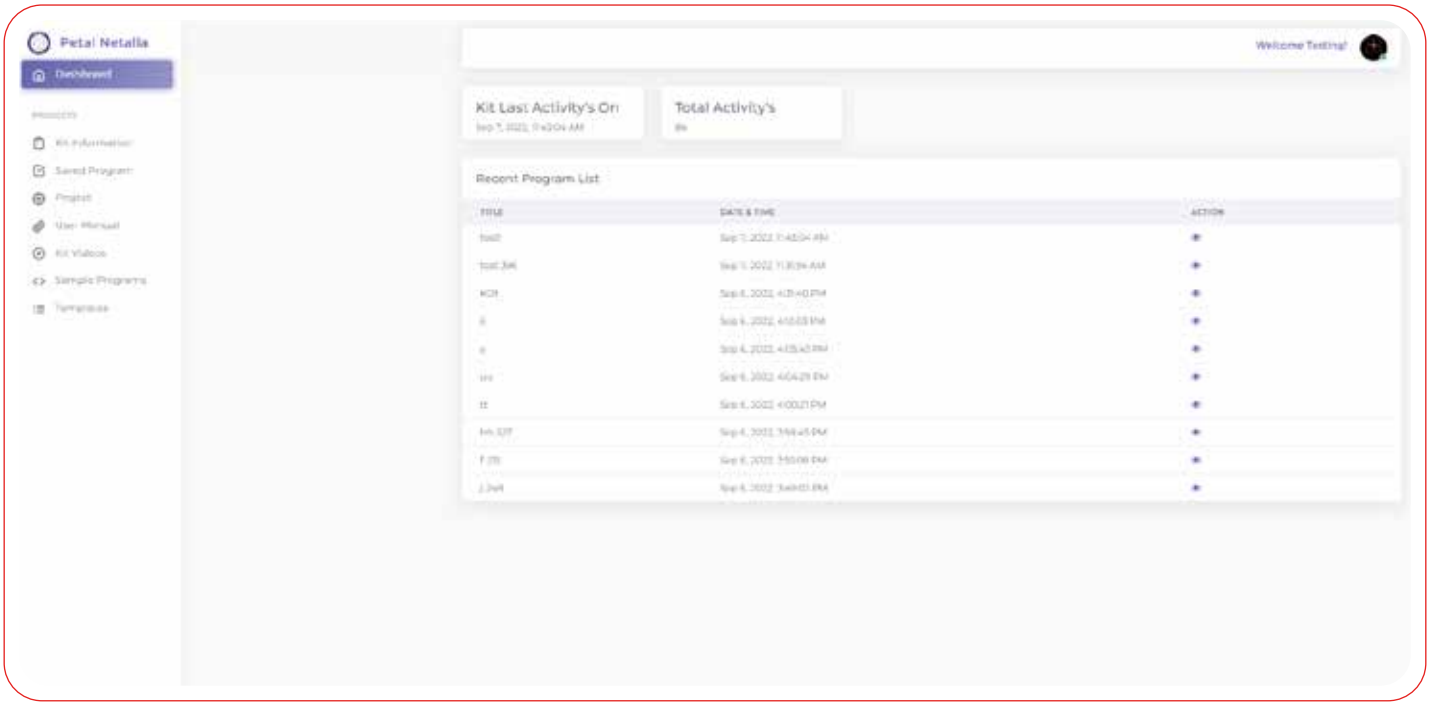
Step 4:

You will redirected to the Login page, if the OTP verification has been completed.



Step 5:

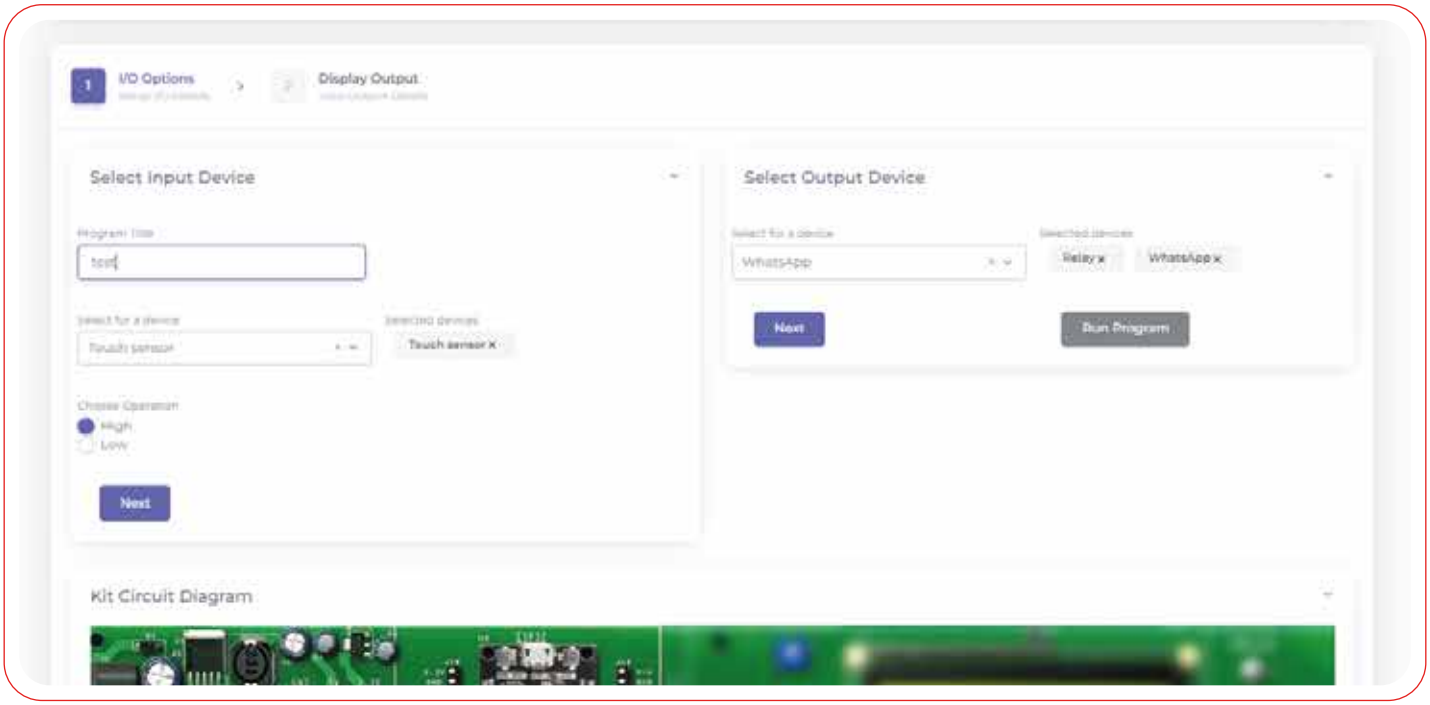
Once Login success Dashboard page will open on the screen.



Step 6:

How to start the project

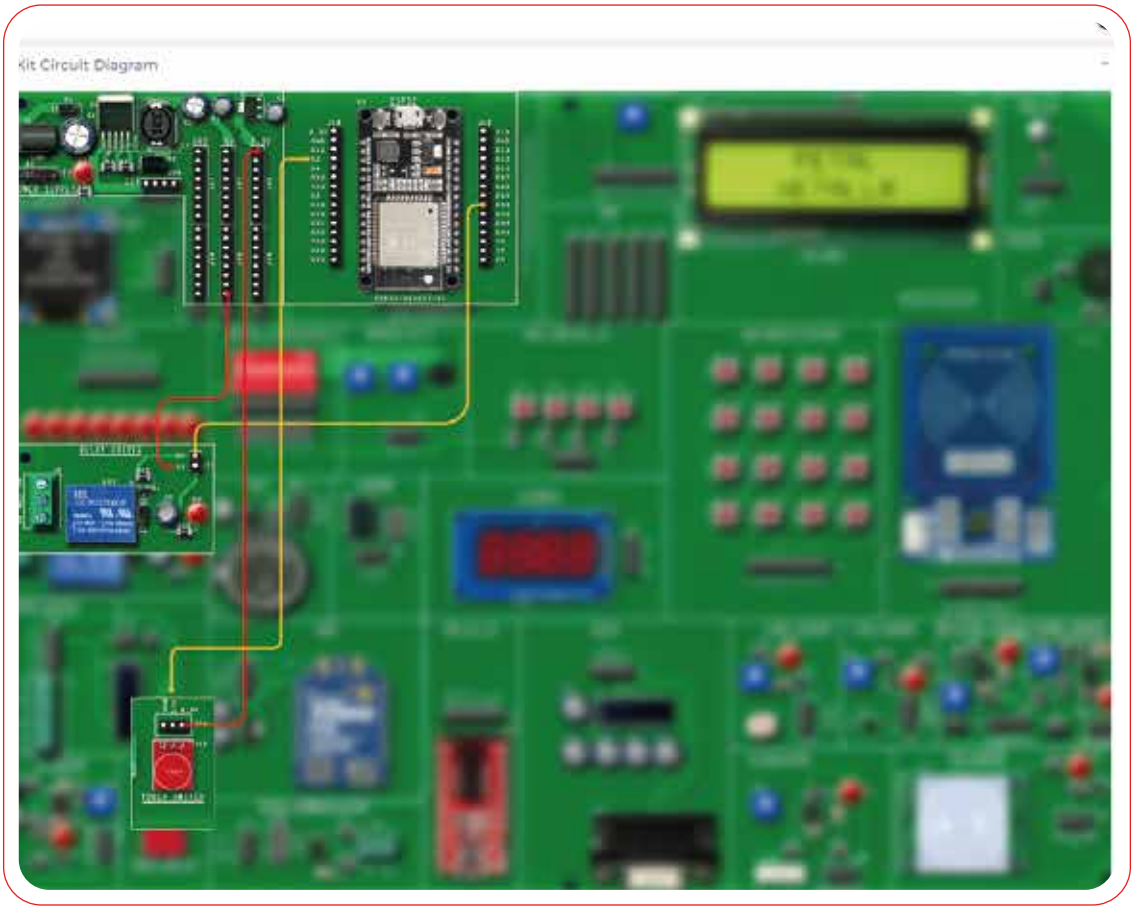
Projects → Give the program title → Select the I/P device with logic (High/Low) → Next → Select the O/P device for selected I/P device with logic (High/Low) → Next → Run Program



Step 7:

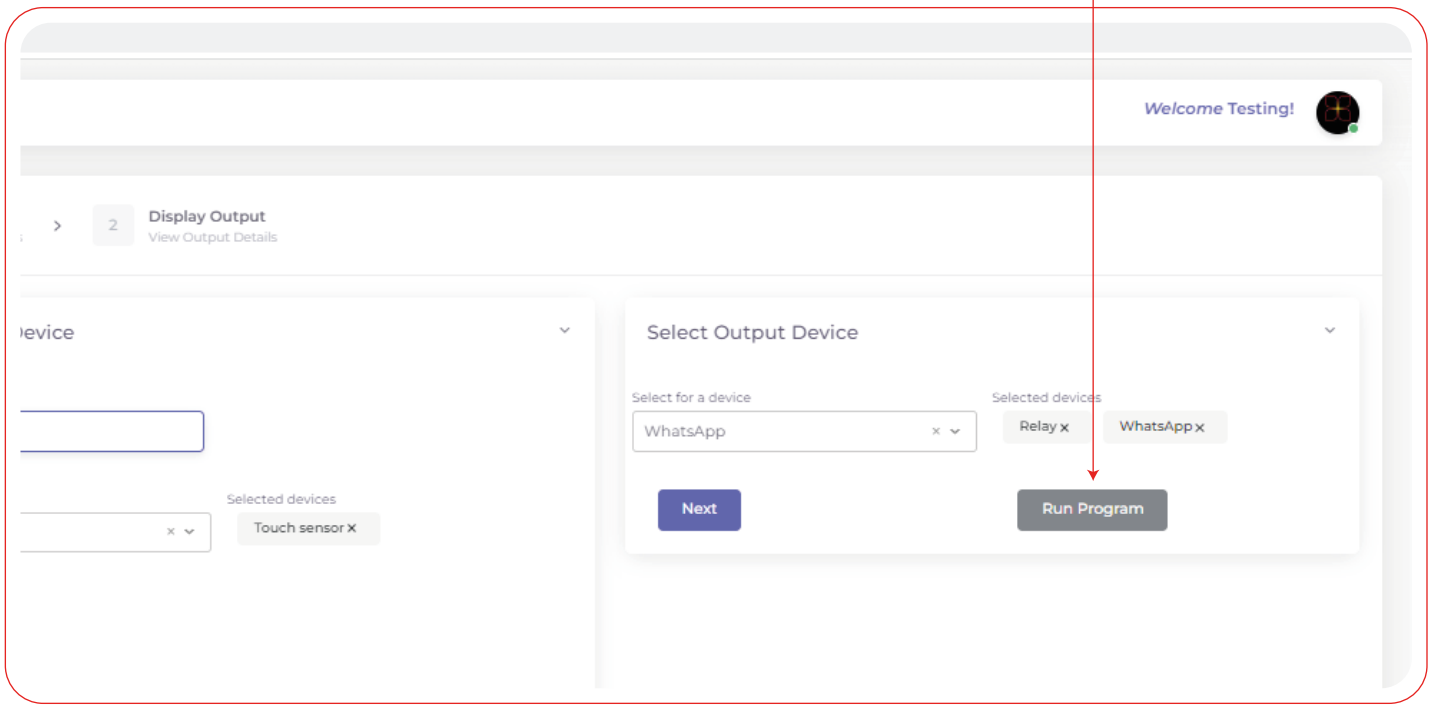
Circuit page for IoT kit wiring reference

Once done Input and Output selection Kit wiring page will view on screen and it helps to find the pin in and out for selected modules and how to wire modules. Do the wiring in IoT Kit by reference to the KIT circuit diagram page.



Step 8:

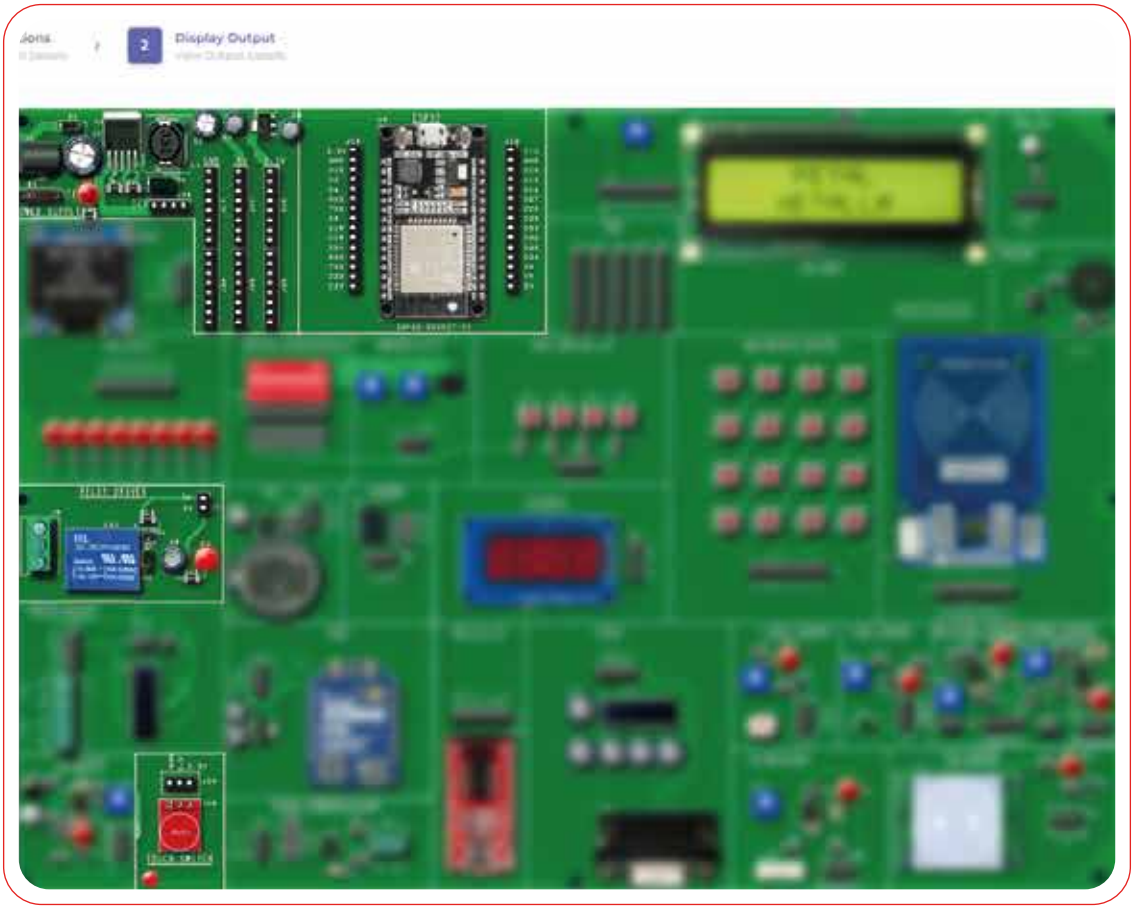
Run Program



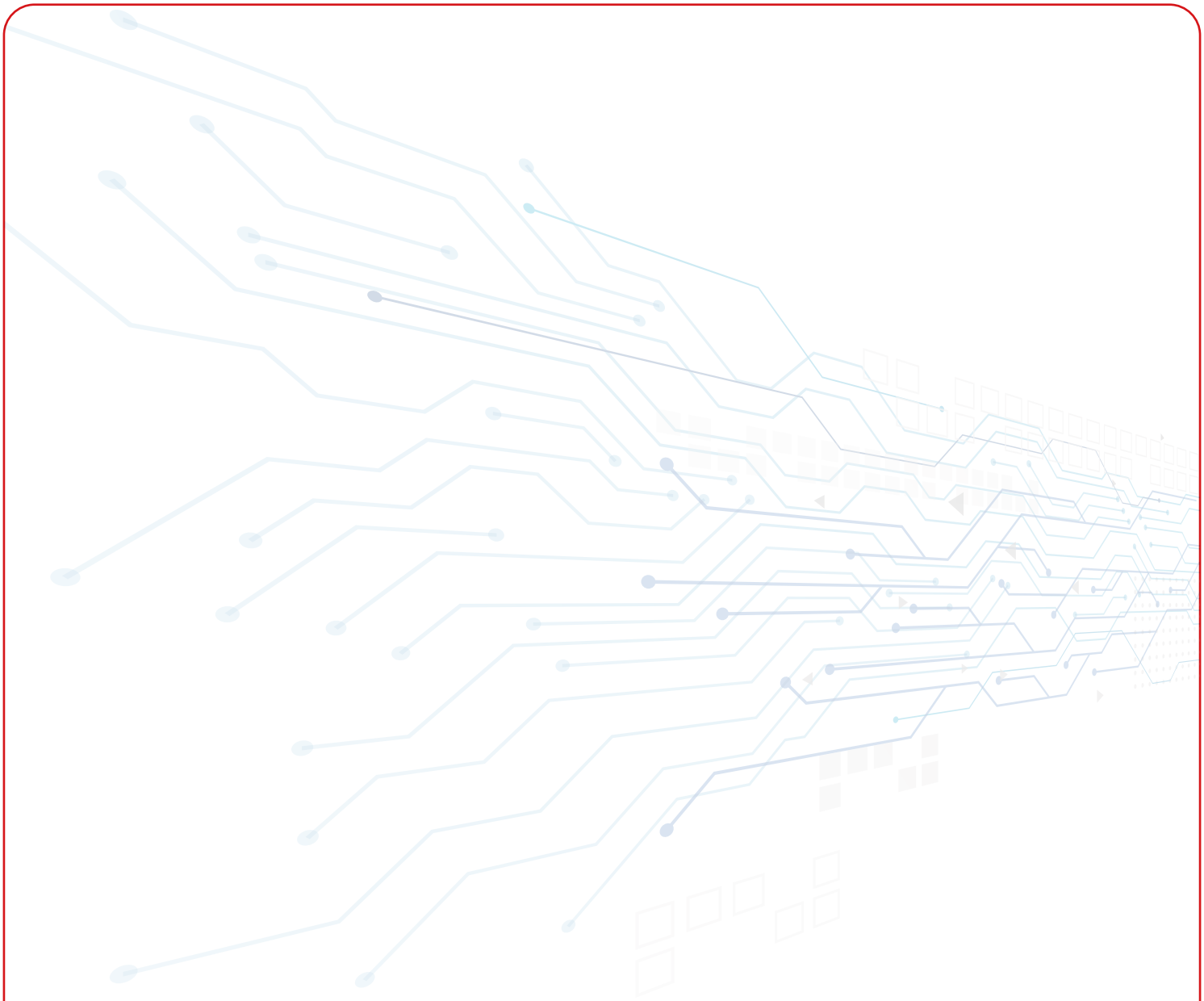
Step 9:

Output Result Page

After the Run program Output page will display on the screen with selected Input and Output devices and there you can monitor the I/O devices working status.



Follow the same procedures above to run different events by I/O devices.



The Petal Automations

Let Innovation Go Further

No:155, LIG Colony, KK Nagar,
Madurai - 625020.
Tamilnadu, India.
www.thepetalautomations.com

Experiment No: 2

Title: Study of different operating systems for Raspberry-Pi/Beagle board/Arduino. Understanding the process of OS installation on Raspberry-Pi/Beagle board/Arduino.

Aim: To study operating systems for platforms such as Raspberry-Pi/Beagle board/Arduino

Introduction:

Theory:

1) **Raspberry-Pi:** - The Pi can run the official Raspbian OS, Ubuntu Mate, Snappy Ubuntu Core, the Kodi- based media centers OSMC and LibreElec, the non-Linux based Risc OS (one for fans of 1990s Acorn computers). It can also run Windows 10 IoT Core, which is very different to the desktop version of Windows, as mentioned below.

- **OS which installs on Raspberry-Pi**: Raspbian, Ubuntu MATE, Snappy Ubuntu, Pidora, Linutop, SARPi, Arch Linux ARM, Gentoo Linux, etc.

➢ **How to install Raspbian on Raspberry-Pi:Step 1:Download Raspbian**

Step 2: Unzip the file. The Raspbian disc image is compressed, so you'll need to unzip it. The file uses the ZIP64 format, so depending on how current your built-in utilities are, you need to use certain programs to unzip it.

Step 3: Write the disc image to your microSD card. Next, pop your microSD card into your computer and write the disc image to it. The process of actually writing the image will be slightly different across these programs, but it's pretty self-explanatory no matter what you're using. Each of these program will have you select the destination (make sure you've picked your microSD card!) and the disc image (the unzipped Raspbian file). Choose, double-check, and then hit the button to write.

Step 4: Put the microSD card in your Pi and boot up. Once the disc image has been written to the microSD card, you're ready to go! Put that sucker into your Raspberry Pi, plug in the peripherals and power source, and enjoy. The current edition to Raspbian will boot directly to the desktop. Your default credentials are username **pi** and password **raspberry.**

2) **BeagleBone Black**: - The **BeagleBone Black** includes a 2GB or 4GB on-board eMMC flash memory chip. It comes with the Debian distribution factory pre-installed. You can flash new operating systems including Angstrom, Ubuntu, Android, and others.

Os which installs on BeagleBone Black: Angstrom, Android, Debian, Fedora, Buildroot, Gentoo, Nerves Erlang/OTP, Sabayon, Ubuntu, Yocto, MINIX 3

- **How to install Debian on BeagleBone Black:Step 1:**

Download Debian img.xz file.

Step 2: Unzip the file.

Step 3: Insert your MicroSD (uSD) card into the proper slot. Most uSD cards come with a full-sized SD card that is really just an adapter. If this is what you have then insert the uSD into the adapter, then into your card reader.

Step 4: Now open Win32 Disk imager, click the blue folder icon, navigate to the debian img location, and double click the file. Now click Write and let the process complete. Depending on your processor and available RAM it should be done in around 5 minutes.

Step 5: Alright, once that's done, you'll get a notification pop-up. Now we're ready to get going. Remove the SD adapter from the card slot, remove the uSD card from the adapter. With the USB cable disconnected insertthe uSD into the BBB.

Step 6: Now, this next part is pretty straight forward. Plug the USB cable in and wait some more. If everything is going right you will notice that the four (4) leds just above the USB cable are doing the KIT impression. This could take up to 45 minutes, I just did it again in around 5 minutes. Your mileage willvary. Go back and surf reddit some more.

Step 7: If you are not seeing the leds swing back and forth you will need to unplug the USB cable, press and hold down the user button above the uSD card slot (next to the 2 little 10 pin ICs) then plug in the USB cable. Release the button and wait. You should see the LEDs swinging back and forth after a few seconds. Once this happens it's waiting time. When all 4 LEDs next to the USB slot stay lit at the same time the flash process has been completed.

Step 8: Remove the uSD card and reboot your BBB. You can reboot the BBB by removing and reconnectingthe USB cable, or hitting the reset button above the USB cable near the edge of the board.

Step 9: Now using putty, or your SSH flavor of choice, connect to the BBB using the IP address 192.168.7.2.You'll be prompted for a username. Type root and press Enter. By default, there is no root password. I recommend changing this ASAP if you plan on putting your BBB on the network. To do this type password, hit enter, then enter your desired password. You will be prompted to enter it again to verify.

3) **Arduino**: - The Arduino itself has no real operating system. You develop code for the Arduino using the Arduino IDE which you can download from Arduino - Home. Versions are available for **Windows**, **Mac** and **Linux**. The Arduino is a constrained microcontroller.

Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, **used** to write and upload computer code to the physical board. You are literally writing the "firmware" when you write the

code and upload it. It's both good and its bad.

- **How to include ESP 32 in Arduino IDE:**

1. Open Arduino IDE and Go to Preferences

- Launch the Arduino IDE.
- Go to **File** > **Preferences** (on Windows) or **Arduino** > **Preferences** (on macOS).

2. Add ESP32 Board URL to the Preferences

- In the **Preferences** window, look for the field **Additional Boards Manager URLs**.
- Add the following URL to the field:

bash
Copy code
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

- If there are already URLs there, separate each URL with a comma.

3. Open Boards Manager

- Go to **Tools** > **Board** > **Boards Manager**.
- In the Boards Manager window, search for **ESP32**.

4. Install the ESP32 Board

- In the search results, find the **esp32 by Espressif Systems** entry.
- Click **Install**.

5. Select the ESP32 Board

- Once the installation is complete, go to **Tools** > **Board**.
- Scroll down to the **ESP32 Arduino** section and select your specific ESP32 board (e.g., **ESP32 Dev Module**).

6. Install Required Drivers

- Depending on your operating system, you may need to install the **CP210x USB to UART Bridge VCP drivers** (common for ESP32 boards).
- You can download them from Silicon Labs' official website.

7. Select the Port

- Connect your ESP32 to your computer via USB.
- Go to **Tools** > **Port** and select the correct port for your ESP32.

Conclusion:

Experiment No: 3

Title: Simple program digital read/write using LED.

Aim: Interfacing LED with Esp-32

Hardware Requirements: ESP32 development board , jumping wires etc.

Software Requirements: Arduino IDE

Theory:

Introduction:

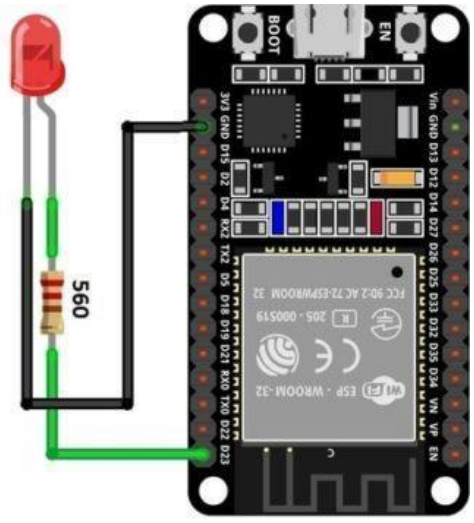


Fig 1 Interfacing LED with Esp-32

Procedure:

1. Make the connection as per diagram
2. Open Arduino IDE
3. Select the Board and the Port
4. Write a code in the editor window
5. Compile and upload the code

Code:

A) LED with Esp-32:

```
#define LED_BUILTIN=4;
void setup()
{

  PinMode(LED_BUILTIN, OUTPUT); // initialize digital pin LED_BUILTIN as an output.

}

void loop()

{

  digitalWrite(LED_BUILTIN, HIGH);          // Turn the LED on (HIGH is the voltage
                                               level)
  delay(1000);                              // Wait for a second
  digitalWrite(LED_BUILTIN, LOW);           // Turn the LED off by making
                                               the voltage LOW
  delay(1000);                              // Wait for a second
}
```

Results:

A) LED with Esp-32 :

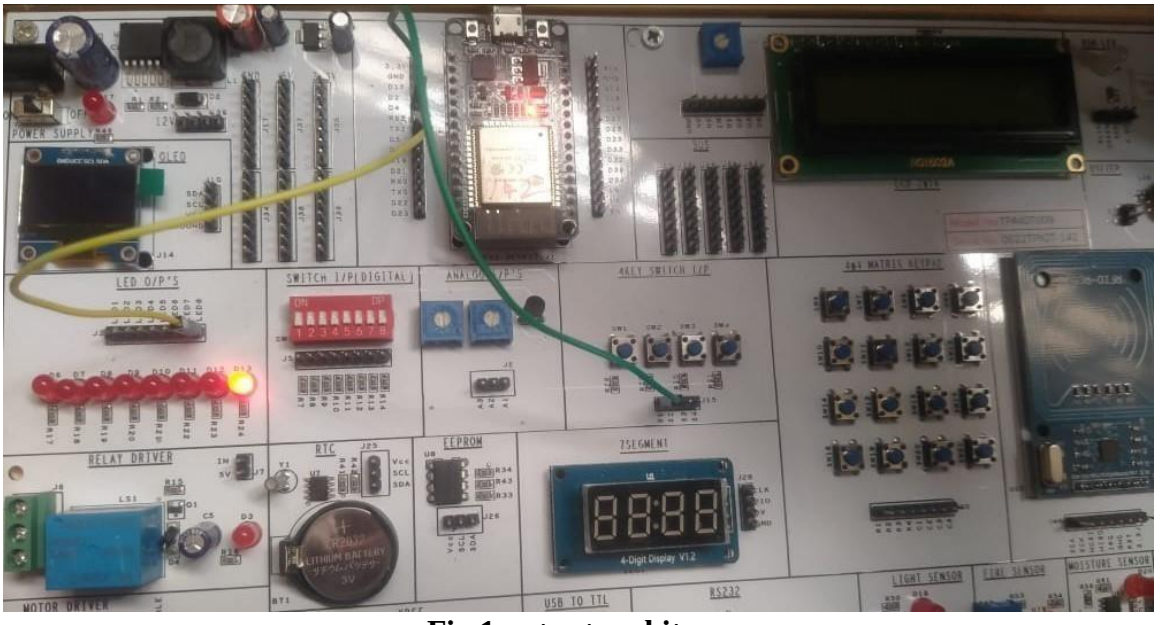


Fig.1 output on kit

Conclusion:

Experiment No:4

Title: Interfacing sensors and actuators withEsp-32.

Aim: To interface DHT11–temperaturehumidity sensor & Sound Sensor with ESP32.

Hardware Requirements: ESP32 development board, DHT11, Sound Sensor, jumping wires etc.

Software Requirements: Arduino IDE

Theory:

A) DHT11Sensor

Fig.1DHT11 Sensor

The DHT11 is a basic, ultra- low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air,and gives out a digital signal on the data pin (no analog input pins needed).

Specifications

- Low cost
- 3to5V power and I/O
- 2.5mA max current use during conversion(while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50° C temperature readings ±2°C accuracy
- Body size 15.5mmx12mmx5.5mm
- 4pins with 0.1"spacing

The DHT11 measures *relative humidity*. The relative humidity is the amount of water vapor in air vs the saturation point of water vapor in the air. At the saturation point, water vapor starts

to condense and accumulate on surfaces forming dew. The saturation point changes with air temperature. Cold air can hold less water vapor before it becomes saturated, and hot air can hold more water vapor before it becomes saturated.

The formula to calculate relative humidity is:

$$RH = \frac{\text{Actual vapor pressure}}{\text{Saturation vapor pressure}} \times 100$$

The relative humidity is expressed as a percentage. At 100% RH, condensation occurs, and at 0% RH, the air is completely dry.

No:	PinName	Description
FourpinSensor		
1	Vcc	Powersupply3.5Vto5.5V
2	Data	OutputsbothTemperatureandHumiditythroughserial Data
3	NC	NoConnectionandhencenotused
4	Ground	Connectedtotheground ofthe circuit
Threepinsensor		
1	Vcc	Powersupply3.5Vto5.5V
2	Data	OutputsbothTemperatureandHumiditythroughserial Data
3	Ground	Connectedtotheground ofthe circuit

Circuit Connection:

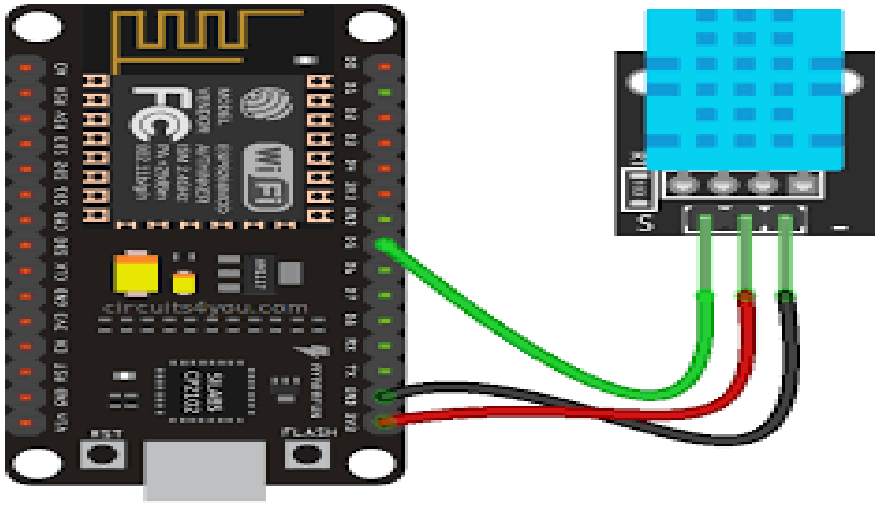


Fig.2 Interfacing of DHT11 with ESP 32

Code:

A) Interfacing DHT11 with Esp-32:

```
#include<DHT.h>
  #define DHTPIN 4              //Digital pin connected to the DHT sensor, DHTPIN+=3.4,
  DHTPIN  GND= GND
  #define DHTTYPE DHT11     //DHT11 sensor
  DHT dht(DHTPIN,DHTTYPE);  //Initialize DHT sensor
  void setup()
  {Serial.begin(9600);
   Serial.println(F("DHT11 test"));
   dht.begin();
  }

  Void loop()

  {
   delay(2000);                        //Wait a few seconds between measurements.
   floath= dht.readHumidity();         //Read humidity
   floatt=dht.readTemperature();   //Read temperature in Celsius

   //Check if any reads failed and exit early(to try again).
   if (isnan(h) || isnan(t))
   {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
   }
   Serial.print(F("Humidity:"))
   ; Serial.print(h);
   Serial.print(F("%Temperature:"));
   Serial.print(t);
   Serial.println(F("°C"));
}
```

B) Interfacing Esp-32 with Sound Sensor: #SOUND_SENSOR

```
#include<dummy.h>
#define SENSOR_PIN18                //ESP32 pin GPIO18 connected to the OUT pin of the sound sensor
int lastState = HIGH;               // The previous state from the input pin int currentState;
                                    //The current reading from the input pin
Void setup()
{
```

```
Serial.begin(9600);                    //Initialize serial communication at 9600 bits per second
 pinMode(SENSOR_PIN,INPUT);            //initialize the ESP32's pin as an input
}

Void loop()
{
currentState= digitalRead(SENSOR_PIN);          //read the state of the ESP32's input pin
if (lastState == HIGH && currentState == LOW)
Serial.println("The sound has been detected");
else if (lastState == LOW && currentState == HIGH)
Serial.println("The sound has disappeared");
lastState=currentState;                                  // save the last state

}
```

Results:

A) Interfacing DHT11 with ESP:

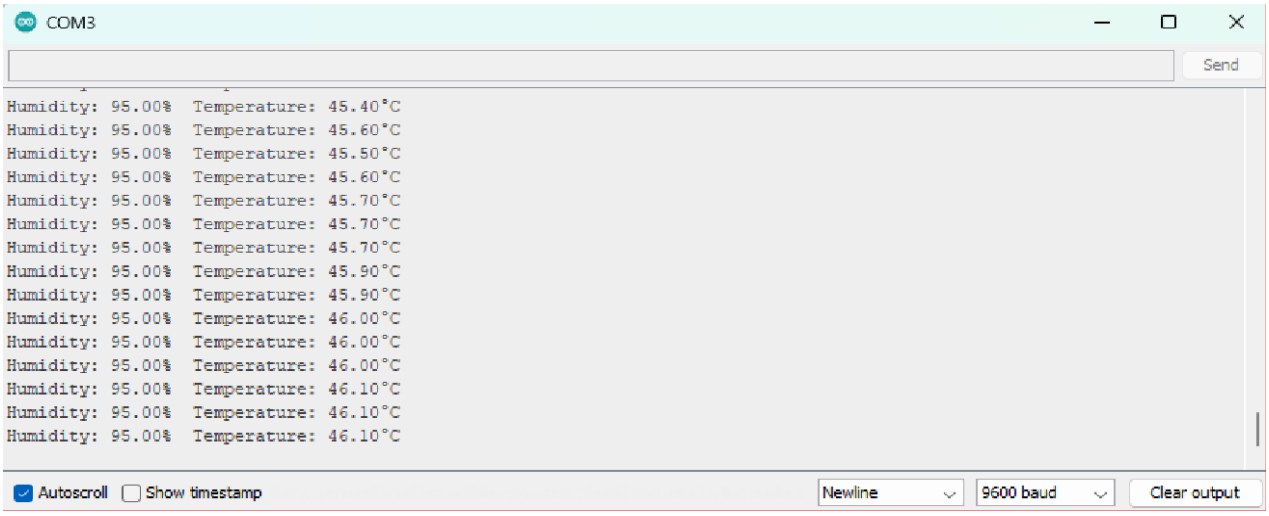


Fig.1 Output of DHT11 sensor

Conclusion:

Modernized IoT

ExperimentNo:5

Title: IoT based Stepper Motor/DC Motor Control with Arduino/Raspberry Pi.

Aim: To Interface DC Motor with Arduino ESP 32.

Hardware Requirements: ESP32 development board, L298N or L293D motor driver, DC motor, jumping wires etc.

Software Requirements: Arduino IDE

Theory:

A DC motor(Direct Current motor)is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.

Fig.1DCMotor

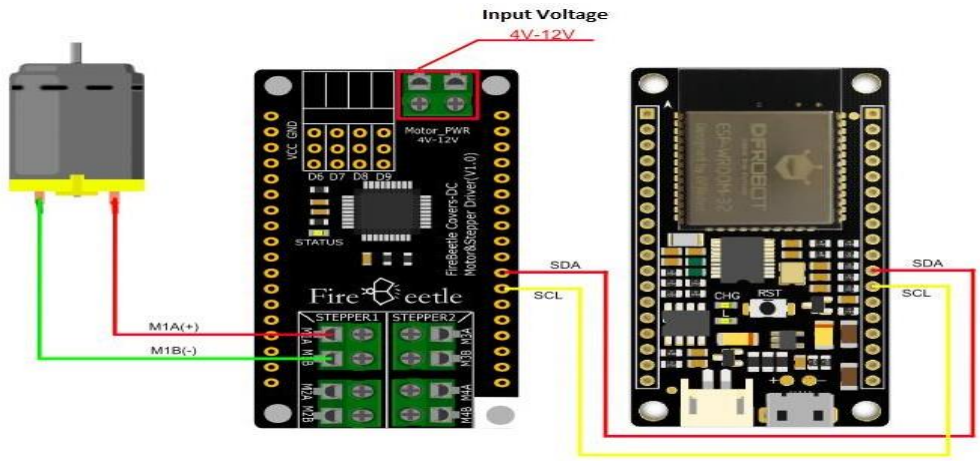


Fig.2 Interfacing of DC Motor with Esp-32

The maximum current of an Arduino I/O port is 20mA but the drive current of a motor is at least 70mA. Therefore, we cannot directly use the I/O port to drive the current; instead, we can use an L293D to drive the motor L293D. L293D is designed to provide bidirectional drive currents of up to 600mA at voltages from 4.5V to 36V. It's used to drive inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

Connections for L298N Motor Driver:

- **IN1, IN2** pins of the motor driver are connected to GPIO pins on the ESP32 for direction control.
- **ENA** pin (Enable A) is connected to a PWM-capable GPIO pin on the ESP32 for speed control.
- **OUT1, OUT2** pins are connected to the two terminals of the DC motor.
- **12V and GND**: Power input for the motor (depending on your motor's rating).

Explanation:

1. **Pin Definitions**:
    - **IN1** and **IN2**: These pins control the direction of the motor. Setting IN1 to HIGH and IN2 to LOW moves the motor forward, and reversing the values moves the motor backward.
    - **ENA**: This is the enable pin for the motor driver, and it's controlled via PWM to adjust motor speed.
2. **PWM Control**:
    - The analogWrite() function is used to provide PWM to the motor driver. This function adjusts the motor speed by setting the PWM duty cycle (0-255).
    - For example, analogWrite(ENA, 128) will run the motor at 50% speed, and analogWrite(ENA, 0) stops the motor.
3. **Motor Operation**:

- The motor is run forward for 2 seconds, stopped for 1 second, then reversed for 2 seconds, and stopped again.
- The speed can be controlled by changing the PWM value.

Code:

```
//Pin definitions for ESP32
Const int motor Pin1=27;              //Connect IN1 of L298N to this pin of esp32 i.e.gpio27
Const int motor Pin2=26;             //Connect IN2 of L298N to this pin of esp32 i.e.gpio26//ground and
3.3v

void setup()
 {
 pinMode(motorPin1, OUTPUT);  //Set motor control pins as outputs
 pinMode(motorPin2, OUTPUT);
}
voidloop()
{
 //Rotate motor in one direction for 2 seconds
 digitalWrite(motorPin1, HIGH);
 digitalWrite(motorPin2, LOW);
 delay(2000);

// Stop motor for 1 second
 digitalWrite(motorPin1,LOW);
 digitalWrite(motorPin2,LOW);
 delay(1000);

 //Rotate motor in the opposite direction for 2 seconds
 digitalWrite(motorPin1, LOW);
 digitalWrite(motorPin2, HIGH);
 delay(2000);

 // Stop motor for 1 second
 digitalWrite(motorPin1,LOW
 );
 digitalWrite(motorPin2,LOW
 ); delay(1000);
}
```

OR

```
// Define pins for motor control

const int IN1 = 15;              // GPIO pin for motor direction control (IN1)
```

```
 const int IN2 = 2;                // GPIO pin for motor direction control (IN2)
 const int ENA = 4;                // PWM-capable GPIO pin for motor speed control (ENA)

void setup()
{
// Set the motor control pins as outputs
pinMode(IN1, OUTPUT);
pinMode(IN2, OUTPUT);
pinMode(ENA, OUTPUT);

// Start the Serial Monitor for debugging
Serial.begin(115200);
 }

void loop()
{
// Run motor in forward direction at 50% speed

 Serial.println("Motor Forward");
 digitalWrite(IN1, HIGH);
digitalWrite(IN2, LOW);
analogWrite(ENA, 128);              // Set PWM to 50% (128 out of 255)
delay(2000);                         // Run for 2 seconds
// Stop the motor
Serial.println("Motor Stop");
analogWrite(ENA, 0);
// Set PWM to 0 (stop the motor)
delay(1000);                        // Stop for 1 second

 // Run motor in reverse direction at 75% speed
Serial.println("Motor Reverse");
digitalWrite(IN1, LOW);
digitalWrite(IN2, HIGH);
 analogWrite(ENA, 192);                // Set PWM to 75% (192 out of 255)
 delay(2000);                           // Run for 2 seconds
// Stop the motor again
Serial.println("Motor Stop");
 analogWrite(ENA, 0);                 // Set PWM to 0 (stop the motor)
delay(1000);                           // Stop for 1 second }
```

Results:

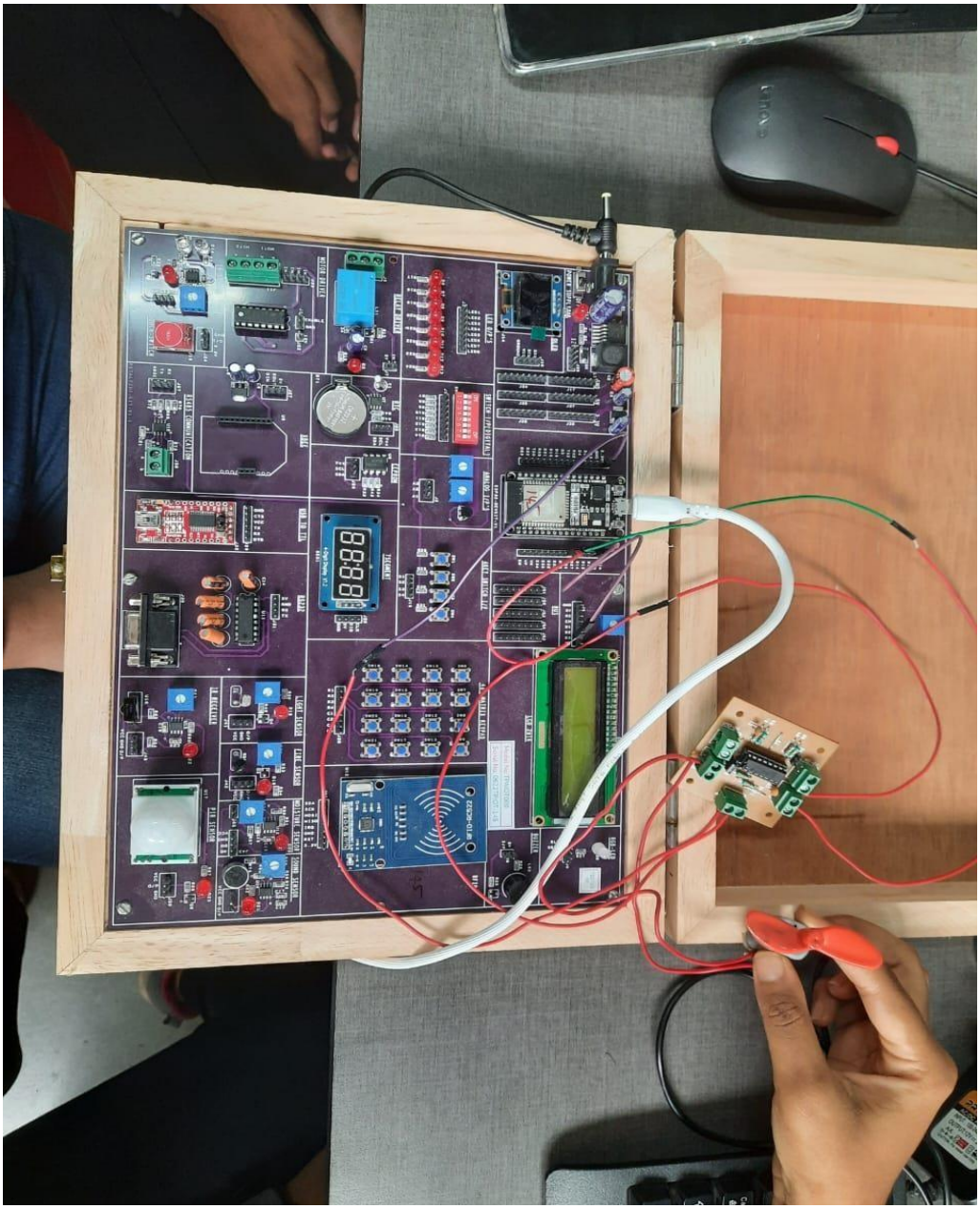


Fig.3 Actual output

Conclusion:

Experiment No: 6

Title: Introduction to MQTT and sending messages to cloud using Esp-32.

Aim: : Introduction to MQTT and store the information on ThingSpeak cloud.

Hardware Requirements: ESP 32 development board

Software Requirements: Arduino IDE, ThingSpeak Cloud

Theory:

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency, or unreliable networks. It is widely used in IoT (Internet of Things) systems to enable communication between devices, sensors, and servers. MQTT is ideal for applications where bandwidth, battery power, and computing capacity are limited.

Key Characteristics:

1) **Lightweight**: MQTT is simple and has minimal overhead, making it suitable for low-power devices.
2) **Publish/Subscribe Model**: MQTT uses a publish/subscribe architecture, which decouples message senders (publishers) from receivers (subscribers).
3) **Low Bandwidth**: Optimized for bandwidth-constrained environments like wireless sensor networks or mobile communication.
4) **Reliable**: MQTT provides three levels of Quality of Service (QoS), ensuring reliable message delivery based on the needs of the application.
5) **Open Standard**: MQTT is an open protocol standardized by OASIS.

How MQTT Works:

1) **Broker**: The central server that handles message distribution. Devices do not communicate directly with each other but send messages to the broker, which then forwards these messages to all subscribers of the relevant topic.
2) **Publisher**: A device that sends data (messages) to the broker under a specific **topic** (like a "channel" or "subject").
3) **Subscriber**: A device that listens for data under a specific **topic** and receives messages sent to that topic.

Workflow:

- **Publish/Subscribe Mechanism**:

  - A device (publisher) sends a message on a **topic** (e.g., "home/livingroom/temperature").
  - The **broker** receives the message and forwards it to any **subscribers** that have subscribed to that topic.
  - Devices can subscribe to one or multiple topics and receive only the relevant data.

MQTT Structure:

- **Topics**: These are hierarchical strings that act as message filters. For example, a topic might be "sensors/temperature/livingroom".
- **Messages**: These contain the payload (the data) and are sent by publishers under a specific topic.

  - **QoS (Quality of Service)**: Defines the level of guarantee for message delivery.
  - **QoS 0**: "At most once" – The message is delivered once with no confirmation.
  - **QoS 1**: "At least once" – The message is guaranteed to arrive, but could be delivered multiple times.
  - **QoS 2**: "Exactly once" – The message is guaranteed to arrive exactly once.

MQTT in IoT:

MQTT is widely used in IoT systems to connect sensors, devices, and applications. It is particularly useful in scenarios where network reliability or power efficiency is a concern, such as smart homes, industrial automation, and remote monitoring systems.

Advantages of MQTT:

1) **Efficient Bandwidth Usage**: Minimal protocol overhead means it can transmit data efficiently over limited networks.
2) **Asynchronous Communication**: Decoupling publishers from subscribers allows for more scalable and flexible communication.
3) **Reliability**: With QoS levels, you can ensure message delivery based on the application's needs.
4) **Open Protocol**: Being an open standard, MQTT has widespread support and numerous implementations in libraries for various platforms and programming languages.

Use Cases:

- **IoT Networks**: Connect sensors, devices, and gateways in home automation, smart cities, and industrial IoT.
- **Mobile Applications**: Used in mobile apps for messaging and live notifications.
- **Remote Monitoring**: Ideal for gathering data from remote or low-powered sensors over cellular networks.

Example MQTT Architecture:

1) A temperature sensor (publisher) in a smart home publishes data to the topic "home/temperature".
2) The MQTT broker receives this message and forwards it to devices (subscribers) like a mobile app, dashboard, or actuator that have subscribed to "home/temperature".
3) Subscribers react to the received messages (e.g., display temperature, trigger an action).

Code:

```
#include <WiFi.h> #include
<PubSubClient.h>

// Replace with your network credentialsconst
char* ssid = "GUEST AISSMSCOE";
const char* password = "";                          // Enter your password

// Replace with your MQTT broker details
const char* mqtt_server = "broker.hivemq.com";      // Example broker
const int mqtt_port = 1883;
const char* mqtt_user = "";                         // If required
const char* mqtt_pass = "";                         // If required

WiFiClient espClient; PubSubClient
client(espClient);

void setup() {
   Serial.begin(115200);
   setup_wifi();
   client.setServer(mqtt_server, mqtt_port);
   client.setCallback(callback);
}
  void loop()
  {
       if (!client.connected())
       {
       reconnect();
       }
     client.loop();
     // Publish a message every 5 secondsstatic
       unsigned long lastMsg = 0; unsigned long
       now = millis();
     if (now - lastMsg > 5000)
      {
     lastMsg = now;
      char msg[50];
     snprintf(msg, 50, "Hello %ld", now);
     client.publish("esp32/test", msg); }}
```

```
}
    void loop() {
      if (!client.connected()) {
        reconnect();
      }
      client.loop();
      // Publish a message every 5 secondsstatic
      unsigned long lastMsg = 0; unsigned long
      now = millis();
      if (now - lastMsg > 5000) {
        lastMsg = now;
        char msg[50];
        snprintf(msg, 50, "Hello %ld", now);
        client.publish("esp32/test", msg);
      }
    }
    void setup_wifi() {
      delay(10);
      Serial.print("Connecting to ");
      Serial.println(ssid); WiFi.begin(ssid,
      password);
      while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected!"); Serial.print("IP
    address: "); Serial.println(WiFi.localIP());
  }

  void reconnect() {
    while (!client.connected()) { Serial.print("Attempting
      MQTT connection...");String clientId = "esp32-client-
      ";
      clientId += String(random(0xffff), HEX);
      if (client.connect(clientId.c_str(), mqtt_user, mqtt_pass)) {
        Serial.println("connected"); client.subscribe("esp32/test");
      } else {
        Serial.print("failed, rc=");
        Serial.print(client.state()); Serial.println(" try
        again in 5 seconds");delay(5000);
      }
    }
  }

  void callback(char* topic, byte* payload, unsigned int length) {
```

```
    Serial.print("Message arrived on topic: "); Serial.print(topic);
    Serial.print(". Message: ");
    for (int i = 0; i < length; i++) {
      Serial.print((char)payload[i]);
    }
    Serial.println();
}
```

Results:

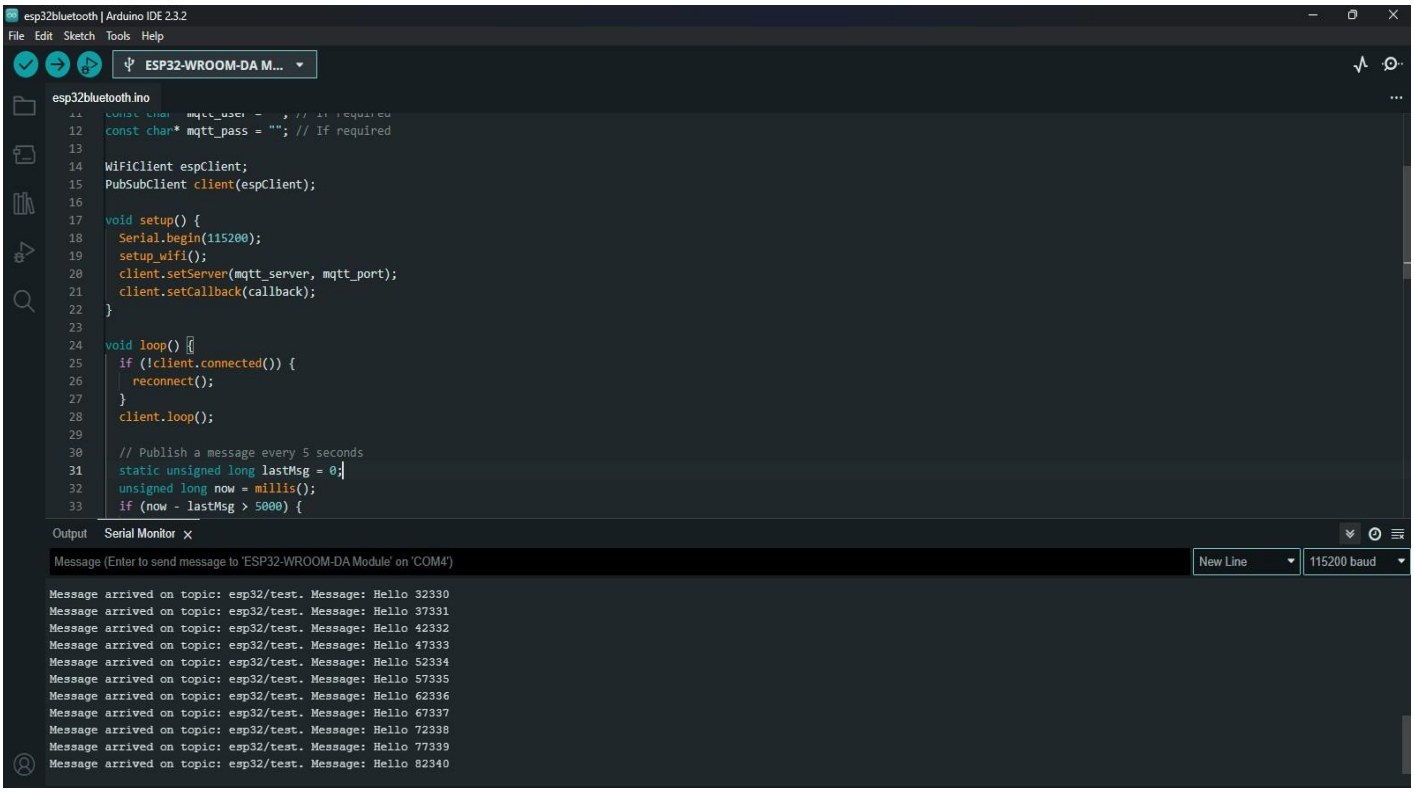


Fig.1 Output

Conclusion:

Modernized IoT

Experiment No: 7

Title: Get the status of a bulb at a remote place(using Wifi).

Aim: Get the status of a bulb at a remote place(using Wifi).

Hardware Requirements: Esp-32 development board,jumping wire etc.

Software Requirements: ArduinoIDE

Theory:

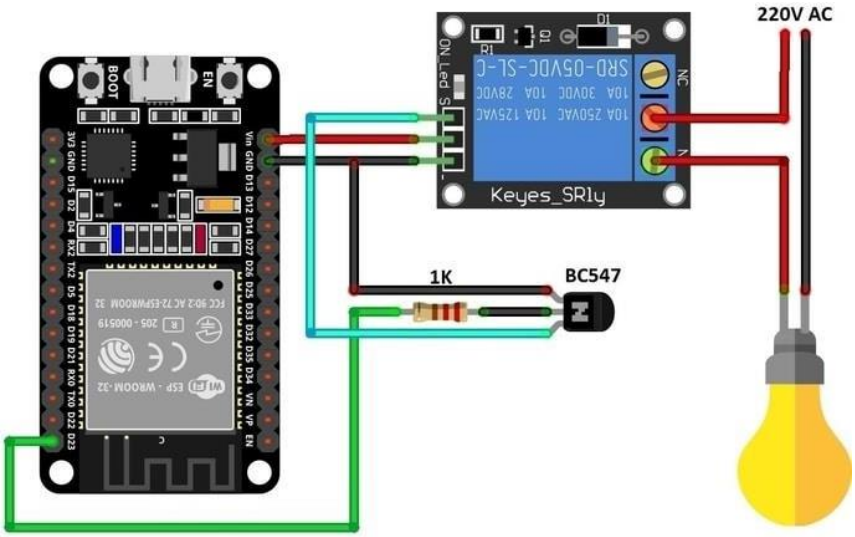


Fig.1Interfacing of Bulb with Esp-32

Procedure

1. GatherESP32,relay module,bulb,wiring,andpower supply.
2. Connect relay in put to ESP32 GPIO13 and relay output to the bulb circuit.
3. Open Arduino IDE, select ESP32 board, and setup Wi-Fi credentials in the code.
4. Upload code to ESP32 and open Serial Monitor to obtain the IPaddress.

5. Access the ESP32 IP address in a web browser on the same network.
6. Use the web interface buttons to turn the bulb on and off via the relay.
7. Observe the bulb's response and check for confirmation messages on Serial Monitor.
8. Record observations, then safely disconnect power and disassemble the setup.

Code:

```
   #include<WiFi.h>

Const char*ssid="PIXEL7";
const char*password="TFXSAITAMA";
intrelayPin=13;              //GPIO13(or an other pin of your choice)
WiFiServer server(80);

void setup()
 { Serial.begin(115200);
delay(10);
pinMode(relayPin,OUTPUT);
 digitalWrite(relayPin,LOW);          //Ensure relay is off initially

  // Connect to WiFi network
  Serial.println();
  Serial.print("Connectingto");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while(WiFi.status()!=WL_CONNECTED)
  { delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  // Start the server
  server.begin();
  Serial.println("Server
  started");

  // Print theIP address
  Serial.print("Use this URL to connect:");
  Serial.print("http://");
  Serial.print(WiFi.localIP());
  Serial.println("/");
}
```

```
Void loop(){
  // Check if a client has connected
  WiFiClientclient=server.available();
   if (!client)
  {
    return;
  }
  //Wait until the client sends some data
  Serial.println("New client");
  while(!client.available())
   {
    delay(1);
  }

  //Read the first line of the request
  Stringrequest=client.read
  StringUntil('\r');
  Serial.println(request);
  client.flush();

  //Match the request
  int value = LOW;
  if (request.indexOf("/LED=ON") != -1)
   { digitalWrite(relayPin,HIGH);              //Turn relay on value = HIGH;
  }
  if (request.indexOf("/LED=OFF") != -1)
  { digitalWrite(relayPin,LOW);                //Turnrelayoff value = LOW;
  }

  // Return the response
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println(""); // do not forget this one
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");client.print("Rela
  y is now: ");
  client.print((value==HIGH)?"On":"Off");
  client.println("<br><br>");
  client.println("<a href=\"/LED=ON\"><button>Turn On</button></a>");
  client.println("<ahref=\"/LED=OFF\"><button>TurnOff</button></a><br/>");
  client.println("</html>");
  delay(1);
  Serial.println("Client
  disconnected"); Serial.println("");
}
```

Results:

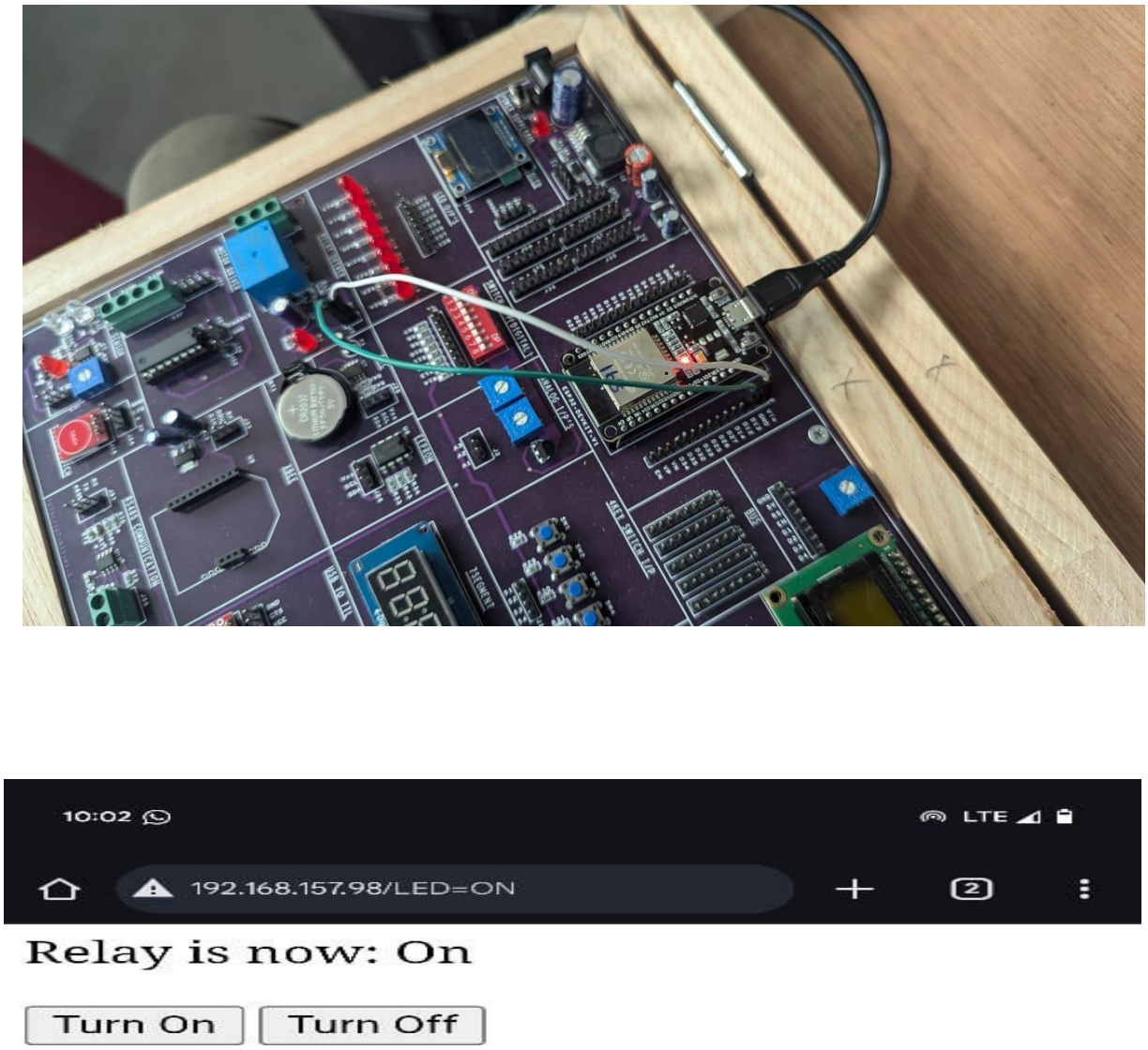


Fig.2 Output

Conclusion:

Modernized IoT

ExperimentNo:8

Title: Interfacing Esp-32 to Bluetooth Module

Aim: To Interface Esp-32 to BluetoothModule

Hardware Requirements: Esp-32development board, Smartphone (with a Bluetooth terminal app such as "Serial Bluetooth Terminal")etc.

Software Requirements: Arduino IDE

Theory:

ESP32 is a popular IoT development board that comes with both **Wi-Fi** and **Bluetooth** capabilities. The Bluetooth functionality includes both classic Bluetooth and BLE (Bluetooth Low Energy). In this experiment, we will use the **classic Bluetooth** mode of ESP32 to communicate with a smartphone, allowing bidirectional communication between the ESP32 and the phone.

Code:

```
//This example code is in the PublicDomain(orCC0licensed,atyouroption.)
//ByEvandroCopercini-2018
//
//This example creates a bridge between Serial and Classical Bluetooth(SPP)
//and also demonstrate that SerialBT have the same functionalities of a normal Serial
//Note:Pairingis authenticated automatically by this device
#include "BluetoothSerial.h"
Stringdevice_name="ESP32-BT-Slave";

//Check if Bluetooth is available
#if!defined(CONFIG_BT_ENABLED)||!defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled!Please run`makemenuconfig`toandenableit
#endif

//Check Serial Port Profile
#if!defined(CONFIG_BT_SPP_ENABLED)
```

```
#error SerialPort Profilefor Bluetooth is not available or not enabled. It is only available for the ESP32 chip.
#endif
BluetoothSerial SerialBT; void
setup() {
   Serial.begin(115200);
   SerialBT.begin(device_name);        //Bluetooth device name
   //SerialBT.deleteAllBondedDevices(); //Uncomment this to delete paired devices; Mustbe
                                called after begin
   Serial.printf("The device with name\"%s\"is started.\nNow you can pair it with
   Bluetooth!\n", device_name.c_str());
}

voidloop()
{
   if(Serial.available())
   {

      SerialBT.write(Serial.read());
   }
   if (SerialBT.available()) { Serial.write(SerialBT.read());
   }
   delay(20);
}
```

Uploading Code:

1) Connect the ESP32 to your computer via the USB cable.
2) Select the ESP32 board in Tools > Board > ESP32 Dev Module.
3) Select the appropriate COM port under Tools > Port.
4) Upload the code by clicking the upload button in the Arduino IDE.

Pairing ESP32 with Smartphone:

1) On your smartphone, open Bluetooth settings and search for available devices.
2) Find and pair with "ESP32_Bluetooth".
3) Open the Serial Bluetooth Terminal app (or any other Bluetooth terminal app).
4) Connect to the ESP32 from the app.

Testing the Communication:

1) In the Serial Bluetooth Terminal app, type a message and send it to the ESP32.

   - You should see the message displayed in the Arduino IDE's Serial Monitor.
   - The ESP32 will echo the received message back to your phone.

2) Similarly, you can send messages from the Serial Monitor, and they will appear in the Bluetooth terminal app.

Expected Output:

- Messages sent from the phone are displayed on the Serial Monitor and echoed back to the phone.
- Messages sent from the Serial Monitor are displayed on the phone's Bluetooth terminal app .

Results:

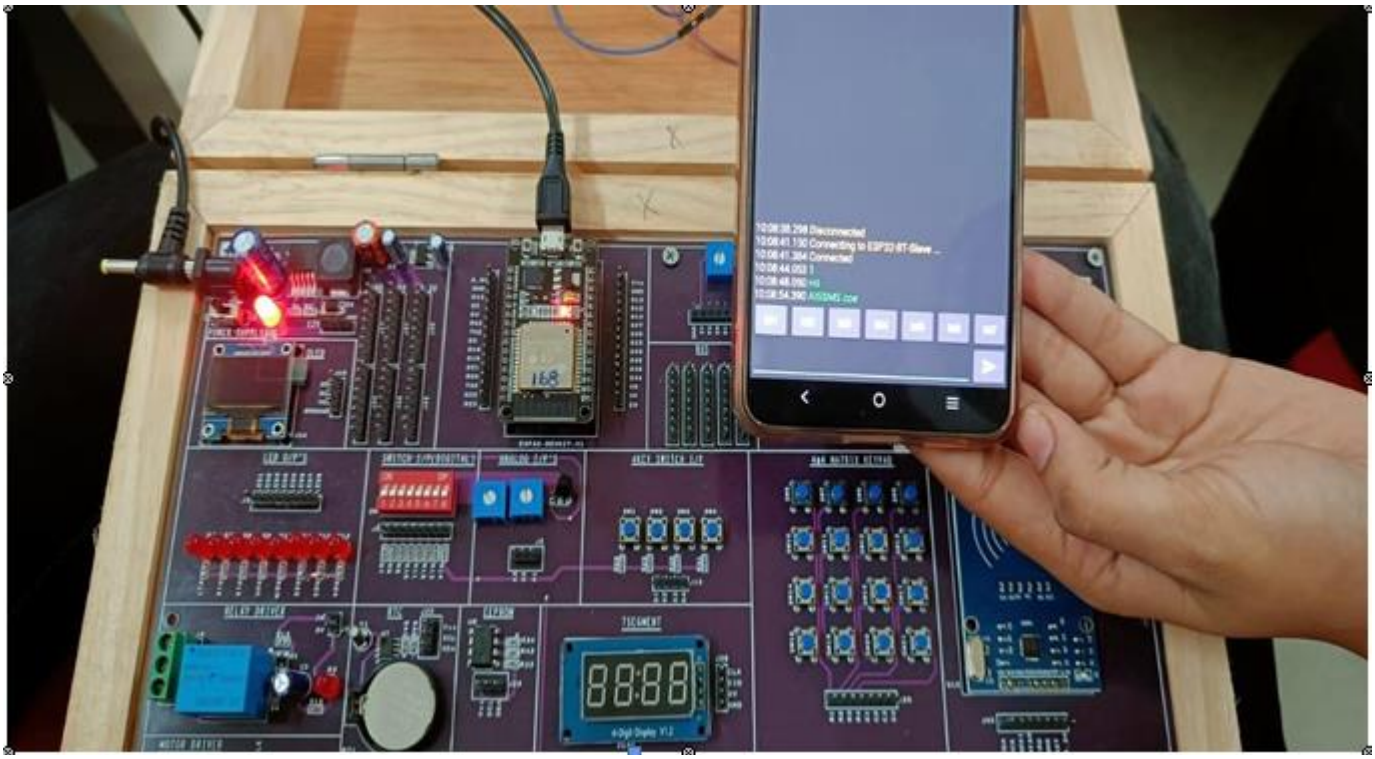


Fig.1Interfacing Bluetooth Module with Esp-32

Conclusion:

Modernized IoT

ExperimentNo:9

Title: Study of Communication between Arduino and Raspberry PI using any wireless medium like ZigBee.

Aim: To Study Communication between Arduino and Raspberry PI using any wireless medium like ZigBee Theory:

Hardware Requirements:

1) Arduino Uno (or any other compatible Arduino board)
2) Raspberry Pi (any model with GPIO access)
3) 2 x **XBee Series 2 modules** (for ZigBee communication)
4) 2 x XBee Shields or USB adapters (for Arduino and Raspberry Pi)
5) Jumper wires
6) USB cables (for programming Arduino and Raspberry Pi)
7) XCTU software (for configuring XBee modules)
8) External power source (for Arduino or Raspberry Pi if required)

Theory:

ZigBee is a low-power wireless communication protocol based on the IEEE 802.15.4 standard. It is ideal for low-data-rate, low-power applications such as IoT devices, sensor networks, and home automation. ZigBee modules like the **XBee** series enable point-to-point, star, and mesh communication, making them perfect for reliable wireless communication between devices like **Arduino** and **Raspberry Pi**.

In this experiment, we will use ZigBee to establish wireless communication between an Arduino (as the transmitting device) and a Raspberry Pi (as the receiving device). The XBee modules will be configured to communicate using the ZigBee protocol, allowing bidirectional data exchange.

Procedure:

1. Configuring the XBee Modules:

Before starting the hardware setup, we need to configure the XBee modules for communication.

Step 1: Install XCTU Software

- Download and install **XCTU** from the Digi website.
- Connect the XBee modules to your computer using XBee USB adapters or shields.
- Launch XCTU and detect the XBee modules.

Step 2: Configure XBee Modules

- Set one XBee module as the **Coordinator** (to be connected to the Raspberry Pi) and the other as an **End Device** or **Router** (to be connected to the Arduino).
- Assign the same **PAN ID** (Personal Area Network ID) to both modules to ensure they are on the same network.
- Set the **Destination Address** of each module to match the serial address of the other, enabling point-to-point communication.

Step 3: Test Communication in XCTU

- After configuration, test the connection between the XBee modules using the XCTU terminal window by sending messages from one module and checking if the other receives them.

2. Hardware Setup:

Setup 1: Arduino with XBee Module

1. Connect the **XBee Shield** to the Arduino board.
2. Place the XBee module on the XBee Shield.
3. Connect the Arduino to the computer via USB for programming.
4. The XBee Shield uses pins D0 (RX) and D1 (TX) for serial communication. These pins are used for sending data wirelessly to the Raspberry Pi.

Setup 2: Raspberry Pi with XBee Module

1. Connect the second XBee module to the Raspberry Pi using an **XBee USB adapter** or by connecting directly to the GPIO pins (using TX, RX, 3.3V, and GND).
2. If using the GPIO pins:
   - Connect **TX of XBee** to **RX of Raspberry Pi**.
   - Connect **RX of XBee** to **TX of Raspberry Pi**.
   - Connect **3.3V** and **GND** properly.
3. Use SSH or connect a monitor, keyboard, and mouse to the Raspberry Pi for programming.

3. Arduino Code:

Upload the following code to the Arduino to send data over ZigBee to the Raspberry Pi.

```
#include <SoftwareSerial.h>

SoftwareSerial XBee(2, 3); // RX, TX pins for XBee communication

void setup() {
  Serial.begin(9600);      // Initialize serial communication with the computer
  XBee.begin(9600);        // Initialize XBee communication
}

void loop() {
  // Send a simple message to the XBee
  XBee.println("Hello from Arduino");
  delay(1000);  // Send data every second
```

}

4. Raspberry Pi Setup:

Step 1: Install Python Libraries for Serial Communication

Install the Python pyserial library to handle communication with the XBee module.

bash
Copy code
sudo apt update
sudo apt install python3-pip
pip3 install pyserial

Step 2: Python Code for Receiving Data on Raspberry Pi

Write a Python script to receive data wirelessly from the Arduino via the XBee module.

python
Copy code

```
import serial

# Set up the serial connection to the XBee module
xbee = serial.Serial('/dev/ttyUSB0', 9600)  # Replace with correct serial port

while True:
    if xbee.in_waiting > 0:
        # Read incoming data from the XBee
        incoming_data = xbee.readline().decode('utf-8').strip()
        print(f"Received from Arduino: {incoming_data}")
```

- Replace '/dev/ttyUSB0' with the correct port for the XBee module on the Raspberry Pi.
- Run the script:

bash
Copy code
python3 xbee_receive.py

5. Testing the Communication:

1) Power up both the Arduino and Raspberry Pi with their respective XBee modules.
2) On the Raspberry Pi, run the Python script to start receiving data.
3) The Arduino will continuously send the message "Hello from Arduino" via ZigBee to the Raspberry Pi.
4) The Raspberry Pi should display the received message in the terminal.

Expected Output:

- The message "Hello from Arduino" sent from the Arduino should appear on the Raspberry Pi terminal every second.
- The communication should be wireless and reliable over a reasonable range (typically 30-100 meters indoors, depending on interference and module specifications).

Conclusion: