**A PROJECT REPORT**

**ON**

**Development & Implementation of Industrial automation system**

*Submitted in partial fulfillment of the requirement*

for

The award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION

ENGINEERING

BY

B. Srilakshmi [22NN1A0454]

D. Gopi Prasanna [22NN1A0463]

M. Rajeswari [22NN1A0474]

B. Sravanthi [23NN5A0402]

Under The Esteemed Guidance of

**Ms . K . Vinuthna, M. Tech**

**Assistant Professor, Dept. of E.C.E**

****

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VIGNAN’S NIRULA INSTITUTE OF TECHNOLOGY & SCIENCE FOR WOMEN

(An ISO 9001:2008 Certified, Approved by AICTE, Affiliated to JNTU, KAKINADA)

**CERTIFICATE**

**VIGNAN’S NIRULA INSTITUTE OF TECHNOLOGY & SCIENCE FOR WOMEN**

(Affiliated to JNTUK & Approved by AICTE)

Vignan’s Avenue, Pedapalakaluru-522009, Guntur (District), A.P

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEER****ING**

### 



This is to certify that the project report entitled **“DEVELOPMENT & IMPLEMENTATION OF INDUSTRIAL AUTOMATION SYSTEM”** the bona fide work carried out by **B. Srilakshmi,** **D. Gopi Prasanna, M. Rajeswari, B. Sravanthi** in partial of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in ELECTRONICS AND COMMUNICTION ENGINEERING from J.N.T. University Kakinada during the academic year 2022-2026.

|  |  |  |
| --- | --- | --- |
| **project Guide**  **Ms. K. Vinuthna**  **Assistant Professor** | External Examiner | **Head of the department**  **Dr. G. Sandhya**  **Professor** |

# ACKNOWLEDGEMENT

I wish to avail this opportunity to thank **Dr. P. Radhika** Principal of Vignan’s Nirula Institute of technology and science for women for her continuous support and giving valuable suggestions during entire period of the project work.

I take this opportunity to express my gratitude to **Dr. G. Sandhya**, Head of the Department for her great help and encouragement to do this project work.

I would like to express my Unbounded gratefulness to my guide **Ms. K. Vinuthna**

Associate professor in department of Electronics &Communication Engineering for her valuable support and motivation at each and every point in successful completion of the project.

I wish to thank all teaching and non-teaching staff of the Electronics &Communication Engineering Department for giving valuable suggestions in completion of the project.

**PROJECT ASSOCIATES**

B. Srilakshmi [22NN1A0454]

D. Gopi Prasanna [22NN1A0463]

M. Rajeswari [22NN1A0474]

B. Sravanthi [23NN5A0402]

**DECLARATION**

We do hereby declare that the project entitled **“DEVELOPMENT & IMPLEMENTATION OF INDUSTRIAL AUTOMATION SYSTEM ”** authentic work carried out by us under the esteemed guidance of **Ms. K. Vinuthna** given in the Ethical Code of Conduct of the Institute and that, to the best of our knowledge and belief, it contains no material previously written by another neither person nor material(data, theoretical analysis, figures, and text) for the partially fulfillment of the award of Bachelor of Technology degree and we further admit that this has not been submitted by us anywhere else for the award of any other degree.

**PROJECT ASSOCIATE**

B. Srilakshmi [22NN1A0454]

**DECLARATION**

We do hereby declare that the project entitled **“DEVELOPMENT & IMPLEMENTATION OF INDUSTRIAL AUTOMATION SYSTEM ”** authentic work carried out by us under the esteemed guidance of **Ms. K. Vinuthna** given in the Ethical Code of Conduct of the Institute and that, to the best of our knowledge and belief, it contains no material previously written by another neither person nor material(data, theoretical analysis, figures, and text) for the partially fulfillment of the award of Bachelor of Technology degree and we further admit that this has not been submitted by us anywhere else for the award of any other degree.

**PROJECT ASSOCIATE**

D. Gopi Prasanna [22NN1A0463]

**DECLARATION**

We do hereby declare that the project entitled **“DEVELOPMENT & IMPLEMENTATION OF INDUSTRIAL AUTOMATION SYSTEM ”** authentic work carried out by us under the esteemed guidance of **Ms. K. Vinuthna** given in the Ethical Code of Conduct of the Institute and that, to the best of our knowledge and belief, it contains no material previously written by another neither person nor material(data, theoretical analysis, figures, and text) for the partially fulfillment of the award of Bachelor of Technology degree and we further admit that this has not been submitted by us anywhere else for the award of any other degree.

**PROJECT ASSOCIATE**

M. Rajeswari [22NN1A0474]

**DECLARATION**

We do hereby declare that the project entitled **“DEVELOPMENT & IMPLEMENTATION OF INDUSTRIAL AUTOMATION SYSTEM ”** authentic work carried out by us under the esteemed guidance of **Ms. K. Vinuthna** given in the Ethical Code of Conduct of the Institute and that, to the best of our knowledge and belief, it contains no material previously written by another neither person nor material(data, theoretical analysis, figures, and text) for the partially fulfillment of the award of Bachelor of Technology degree and we further admit that this has not been submitted by us anywhere else for the award of any other degree.

**PROJECT ASSOCIATE**

B. Sravanthi [23NN5A0402]

**INDEX**

**CONTENTS**

ABSTRACT

**C**[**HAPTER 1: INTRODUCTION**](#_TOC_250013)

* 1. INTRODUCTION
  2. PROBLEM STATEMENT
  3. PROBLEM SOLUTION
  4. KEY FEATURES OF THE SOLUTION
  5. LITERATURE SURVEY

**CHAPTER 2: BLOCK DIAGRAM**

**CHAPTER 3: COMPONENTS & ITS SPECIFICATIONS**

3.1 ESP32/NODEMCU MICROCONTROLLER

3.2 ULTRASONIC SENSOR (HC-SR04)

* 1. DHT SENSOR MODULE WITH LED

3.4 MQ -7 GAS SENSOR

3.5 LM7805 POWER SUPPLY MODULE

**CHAPTER 4: SOFTWARE & INSTALLATION PROCEDURE**

4.1 ABOUT ARDUINO IDE AND ITS FEATURES

4.2 WHY ARDUINO IDE?

4.3 WHY ARDUINO IDE FOR ESP32/ESP8266?

4.4 STEP BY STEP INSTALLATION PROCEDURE

**CHAPTER 5: IOT CLOUD & SERVER PLATFORM**

**CHAPTER 6: WORKING PRINCIPAL OF IDEATION**

6.1 TELL ABOUT PROJECT WORKING PROCESSES

**CHAPTER 7: PROJECT PHOTOS & RESULTS**

7.1 PROJECT PHOTOS

7.2 SCREENSHOTS OF RESULTS

**CHAPTER 8: CONCLUSION**

8.1 PROJECT CONCLUSION

**CHAPTER 9: FUTURE SCOPE**

9.1 FUTURE ADDONS OR UPDATES IN PROJECT

**CHAPTER 10: FINAL CODE & RESOURCES**

10.1 CODE

10.2 RESOURCES

**LIST OF FIGURES**

**FIG 2.1. BLOCK DIAGRAM OF AUTOMATION SYSTEM**

**FIG 3.1. ESP32 32BIT-MICROCONTROLLER**

**FIG 3.2. ULTRASONIC SENSOR (HC-SR04)**

**FIG 3.3. DHT SENSOR MODULE WITH LED**

**FIG 3.4. MQ -7 GAS SENSOR**

**FIG 3.5. LM7805 POWER SUPPLY MODULE**

**FIG 4.1. ARDUINO IDE SCREEN**

**FIG 4.2. TOOLBAR BUTTON**

**FIG 4.3. ADD FILE**

**FIG 4.4. ARDUINO SOFTWARE PAGE**

**FIG 4.5. RUNNING THE EXECUTABLE ARDUINO IDE 2 INSTALLATION FILE**

**FIG 4.6. ADDITIONAL BOARDS MANAGER URLS FIELD.**

**FIG 4.7. UPLOADING THE SKETCH**

**FIG 4.8. BLYNK LIBRARY INSTALLATION**

**FIG 5.1. EXAMPLE OF IOT SYSTEM**

**FIG 5.2. IOT INFRASTRUCTURE**

**FIG 5.3. PUBLISH-SUBSCRIBE MODEL**

**FIG 5.4. WORKING OF MQTT**

**FIG 5.5. ARDUINO IDE READY TO PROGRAM BY USING BLYNK APP**

**FIG 7.1. PROJECT PHOTOS**

**FIG 7.2. SCREENSHOT OF RESULTS**

**ABSTRACT**

Industrial automation refers to the use of control systems, such as computers, robotics, and information technologies, to handle industrial processes with minimal human intervention. It enhances efficiency, precision, and safety while reducing operational costs and errors.

This system integrates hardware and software components, including Programmable Logic Controllers (PLCs), Human-Machine Interfaces (HMIs), Supervisory Control and Data Acquisition (SCADA) systems, and Artificial Intelligence (AI) to optimize production workflows. Modern automation employs Industrial Internet of Things (IIoT), cloud computing, and machine learning to enable predictive maintenance, real-time monitoring, and data-driven decision-making.

By automating manufacturing, assembly lines, and quality control, industries achieve higher productivity, consistent product quality, and improved workplace safety. As technology evolves, smart automation is becoming more adaptive and interconnected, paving the way for Industry 4.0 and smart factories.

**CHAPTER 1**

1. **INTRODUCTION**

**1.1 Introduction:** The landscape of industrial automation is undergoing a significant transformation driven by the rapid advancement of the Internet of Things (IoT). As industries strive for increased efficiency, reduced operational costs, and enhanced productivity, IoT technologies offer innovative solutions to traditional automation challenges. By enabling connectivity between machines, sensors, and data analytics platforms, IoT facilitates real-time monitoring and control of industrial processes.

Industrial automation systems traditionally relied on isolated control mechanisms that limited data exchange and responsiveness. However, the integration of IoT introduces a paradigm shift, allowing for a more interconnected and intelligent approach. This paper explores the development and implementation of an IoT-based industrial automation system that not only streamlines operations but also enhances decision-making through data-driven insights.

The system comprises several key components, including various sensors for data acquisition, actuators for process control, and a cloud-based infrastructure for data storage and analysis. By leveraging these technologies, organizations can achieve greater visibility into their operations, implement predictive maintenance strategies, and optimize resource allocation.

This introduction sets the stage for a comprehensive exploration of the methodologies employed in developing the IoT-based automation system, the challenges encountered, and the tangible benefits realized in real-world applications. As industries continue to embrace digital transformation, the insights gained from this project contribute to the evolving narrative of IoT in industrial automation.

**1.2 Problem Statement**: In the rapidly evolving industrial sector, many companies struggle with outdated automation systems that hinder operational efficiency and responsiveness. Traditional systems often rely on manual data entry and lack the ability to provide real-time insights into machinery performance, leading to delayed responses to equipment failures and unplanned downtime. This inefficiency not only escalates maintenance costs but also affects overall productivity and resource utilization. Moreover, the absence of advanced data analytics prevents organizations from making informed decisions, resulting in suboptimal energy consumption and material waste. To address these challenges, this project aims to develop and implement an IoT-based industrial automation system. By integrating IoT sensors for continuous real-time monitoring, the system will facilitate immediate data acquisition and analytics through a cloud-based platform. Additionally, it will incorporate predictive maintenance algorithms to foresee equipment failures and optimize maintenance schedules. Ultimately, this solution will enhance operational efficiency, reduce downtime, and improve resource management, providing a scalable framework that can evolve alongside the dynamic needs of the industry.

**1.3 Problem Solution:** To address the challenges faced in traditional industrial automation, the proposed solution involves the development and implementation of an IoT-based industrial automation system with the following key components:

1. **Real-Time Monitoring and Control:**
   * Implement IoT sensors to continuously monitor various parameters (temperature, pressure, humidity, etc.) in real time.
   * Use actuators connected to a centralized control system to enable immediate responses to changing conditions, thereby minimizing delays in operations.
2. **Data Integration and Cloud Computing:**
   * Develop a cloud-based infrastructure that consolidates data from multiple sensors and devices, allowing for easy access and analysis.
   * Utilize data integration platforms that can connect with existing systems, breaking down data silos and facilitating comprehensive insights.
3. **Predictive Maintenance:**
   * Employ machine learning algorithms to analyze historical data and predict equipment failures before they occur.
   * Implement alerts and notifications to schedule maintenance proactively, reducing unplanned downtime and maintenance costs.
4. **User-Friendly Interfaces:**
   * Design intuitive dashboards and mobile applications that provide operators with easy access to real-time data and control functionalities.
   * Include customizable views that allow users to focus on specific metrics relevant to their roles, enhancing decision-making capabilities.
5. **Scalability and Flexibility:**
   * Build the system architecture to be modular, allowing for the easy addition of new devices and sensors as needed.
   * Ensure compatibility with various industrial protocols to facilitate integration with diverse equipment and technologies.
6. **Enhanced Data Analytics:**
   * Implement advanced data analytics tools to derive actionable insights from the collected data.
   * Use visualization techniques to present data trends and performance metrics, supporting informed strategic decisions.
7. **Security Measures:**
   * Incorporate robust cybersecurity protocols to protect sensitive industrial data from unauthorized access and cyber threats.
   * Implement regular security assessments and updates to ensure ongoing protection of the system.

By integrating these components, the IoT-based industrial automation system will enhance operational efficiency, facilitate proactive decision-making, and ultimately drive significant cost savings. The solution aims not only to improve current processes but also to position the organization for future growth and innovation in an increasingly digital manufacturing landscape.

**1.4 Key Features of the Solution:**

Object Detection and Distance Measurement: The ultrasonic sensor detects the object and measures the distance to it using time-of-flight principles.

Gases Detection: The MQ7 sensor detects the harmful gases

Humidity and Temperature: The DTH sensor detects the humidity and temperature in an area and measures the both humidity and temperature of the place.

IoT Integration with Blynk: The ESP32 / NodeMCU sends real-time data to the Blynk platform, allowing users to monitor object detection and distance on their mobile devices.

Alerts and Notifications: The system can be configured to send notifications to the user when an object is detected within a certain distance range, enhancing security and surveillance applications.

**1.5 Literature Survey**

The integration of the Internet of Things (IoT) into industrial automation has been a subject of extensive research and development in recent years. This literature survey highlights key studies and advancements in the field, focusing on the benefits, challenges, and technological innovations associated with IoT-based automation systems.

1. **IoT in Industrial Automation:**
   * A study by J. Lee et al. (2018) discusses the transformative impact of IoT on manufacturing processes, emphasizing the ability to collect real-time data for improved decision-making. The authors argue that IoT enhances operational efficiency through predictive analytics and machine learning applications.
2. **Challenges of Traditional Automation:**
   * Research by M. Khan et al. (2020) identifies several limitations of traditional automation systems, including data silos, lack of real-time analytics, and difficulties in scaling. This work provides a foundation for understanding the necessity of integrating IoT solutions to overcome these barriers.
3. **Frameworks for IoT Implementation:**
   * In their paper, P. Zhang and Y. Zhang (2019) propose a comprehensive framework for implementing IoT in industrial settings. The framework outlines key components such as sensor networks, data management, and cybersecurity measures, offering a structured approach to integrating IoT technologies into existing systems.
4. **Case Studies and Applications:**
   * Various case studies demonstrate successful IoT implementations in industries such as manufacturing, logistics, and energy. For instance, a case study by S. Patel et al. (2021) highlights a smart factory where IoT technologies led to a 20% increase in production efficiency and a significant reduction in downtime through predictive maintenance strategies.
5. **Security and Privacy Concerns:**
   * The increasing reliance on IoT raises significant security and privacy issues. Research by A. Kumar and R. Gupta (2022) explores the vulnerabilities associated with connected devices and proposes a multi-layered security approach to safeguard industrial IoT systems against cyber threats.
6. **Future Trends in IoT for Industrial Automation:**
   * As highlighted by C. Silva et al. (2023), emerging technologies such as edge computing and AI integration with IoT are poised to further enhance industrial automation systems. These advancements allow for faster data processing and improved decision-making at the operational level.

**CHAPTER 2**

1. **BLOCK DIAGRAM**

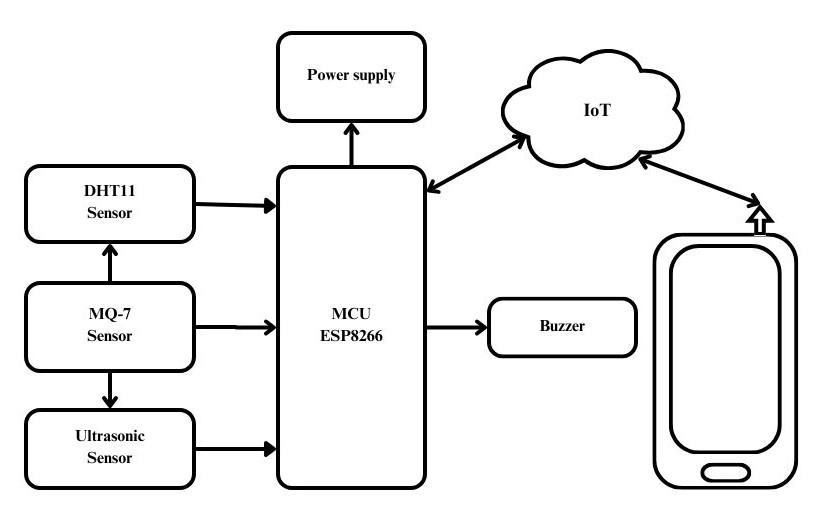
****

Fig 2.1. Block diagram of automation system

**CHAPTER 3**

1. **COMPONENTS & ITS SPECIFICATIONS**

**3.1 ESP32/NodeMCU Microcontroller**

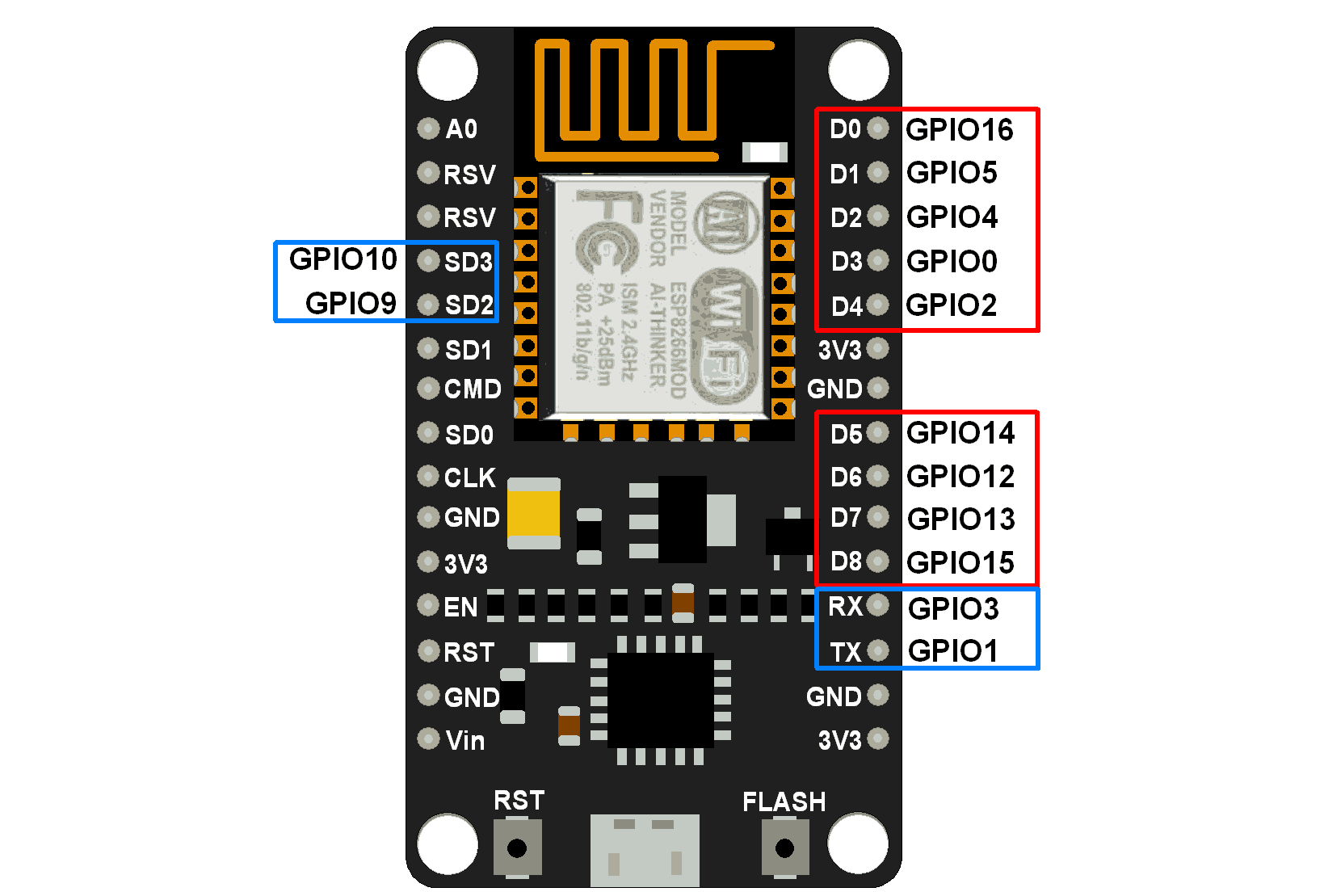


Fig 3.1. ESP32 32bit-Microcontroller

The ESP32 / ESP8266 or NodeMCU is the heart of the project, handling communication between the ultrasonic sensor and the Blynk IoT platform. It also processes the distance and angle data from the ultrasonic sensor.

Microcontroller: ESP32 / ESP8266-WROOM-32 or ESP8266 NodeMCU

Operating Voltage: 3.3V

Input Voltage: 7-12V (through Vin pin)

Wi-Fi: 802.11 b/g/n

Bluetooth: BLE and Classic (only in ESP32 / ESP8266)

GPIO Pins: 36 (ESP32 / ESP8266) or 17 (NodeMCU)

PWM Pins: ESP32 / ESP8266: up to 16 channels, NodeMCU: 8 PWM channels

Flash Memory: 4MB

CPU: Dual-core (ESP32 / ESP8266), Single-core (NodeMCU)

Clock Speed: 240 MHz (ESP32 / ESP8266), 80 MHz (NodeMCU)

Analog Input Pins: 18 ADC channels (ESP32 / ESP8266), 1 ADC channel (NodeMCU)

**3.2 Ultrasonic Sensor (HC-SR04)**

The ultrasonic sensor is used to detect the distance of an object by emitting sound waves and receiving their reflections.

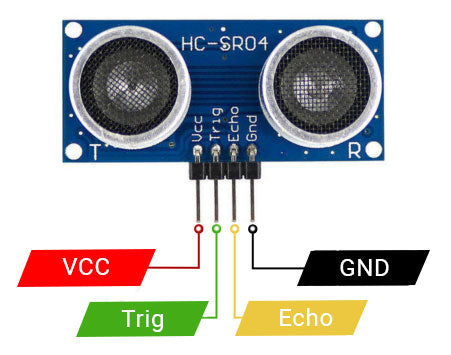


Fig 3.2.UltrasonicSensor **(**HC**-**SR04**)**

Model: HC-SR04

Operating Voltage: 5V DC

Quiescent Current: < 2mA

Working Current: 15mA

Measurement Range: 2cm - 400cm

Accuracy: ±3mm

Operating Frequency: 40 kHz

Trigger Input Signal: 10µs pulse

Echo Output Signal: Pulse width proportional to distance

**3.3 DHT Sensor Module with LED**

The DHT 11 Temperature and Humidity Sensor Module with LED is a small humidity and temperature sensor that you can connect to your Arduino and get readings for temperature and humidity in the environment.

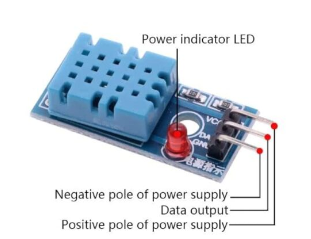
****

Fig 3.3. DHT sensor module with LED

Operating Voltage: 3.3V to 5V DC

Temperature Range: 0°C to 50°C

Temperature Accuracy: ±2°C

Humidity Range: 20% to 90% RH

Humidity Accuracy: ±5% RH

Response Time: <5 seconds

Sampling Rate: 1 Hz

Output form: Digital output

**3.4 MQ -7 Gas sensor**

MQ-7 (CO) Carbon Monoxide Coal Gas Sensor Module detects the concentrations of CO in the air and outputs its reading as an analog voltage. The sensor can measure concentrations of 10 to 10,000 ppm.

The sensor can operate at temperatures from -10 to 50°C and consumes less than 150 mA at 5 V.

This module provides both digital and analog outputs. The threshold level for digital output can be easily adjusted using the preset on the board. The MQ-7 sensor module can be easily interfaced with Micro-controllers, Arduino and etc.

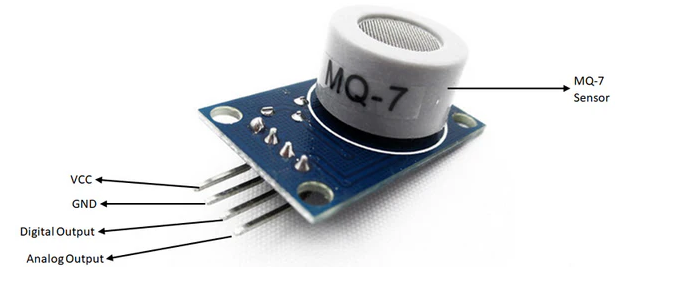


Fig 3.4.MQ -7 Gas sensor

Operating Voltage is +5V

Can be used to Measure or detect CO

Analog output voltage: 0V to 5V

Digital Output Voltage: 0V or 5V

Stable, Long life and Low Cost

Fast Response time

Heater consumption about 350mW

The Sensitivity of Digital Output pin can be varied using the potentiometer

**3.5 LM7805 Power Supply Module**

The LM7805 is a widely used linear voltage regulator designed to provide a stable 5V output from a higher input voltage, typically between 7V and 35V. It is ideal for applications where consistent 5V DC power is required, making it a perfect fit for this radar project that involves microcontrollers like the ESP32 / ESP8266, servo motors, and ultrasonic sensors.



Fig 3.5. LM7805 Power Supply Module

Specifications:

Input Voltage Range: 7V to 35V

Output Voltage: 5V ± 2% tolerance

Maximum Output Current: 1.5A

Quiescent Current (Idle Current): 5mA

Dropout Voltage: 2V (minimum input must be 2V higher than the output, i.e., 7V for a 5V output)

Thermal Protection: Built-in overheat protection

Short Circuit Protection: Yes

Operating Temperature Range: 0°C to 125°C

Package Type: TO-220

Pinout:

Input (Vin): The voltage input pin, where a higher voltage (7V to 35V) is supplied.

Ground (GND): The common ground pin shared by both input and output circuits.

Output (Vout): The regulated 5V output pin.

**CHAPTER 4**

1. **SOFTWARE & INSTALLATION PROCEDURE**

**4.1 About Arduino IDE and its features**

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as Windows, Mac OS X, and Linux. It supports the programming languages C and C++. Here, IDE stands for Integrated Development Environment.

The program or code written in the Arduino IDE is often called as sketching. We can connect Arduino board / ESP32 / ESP8266 / NodeMCU with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino '

The Arduino IDE will appear as in Fig 5:

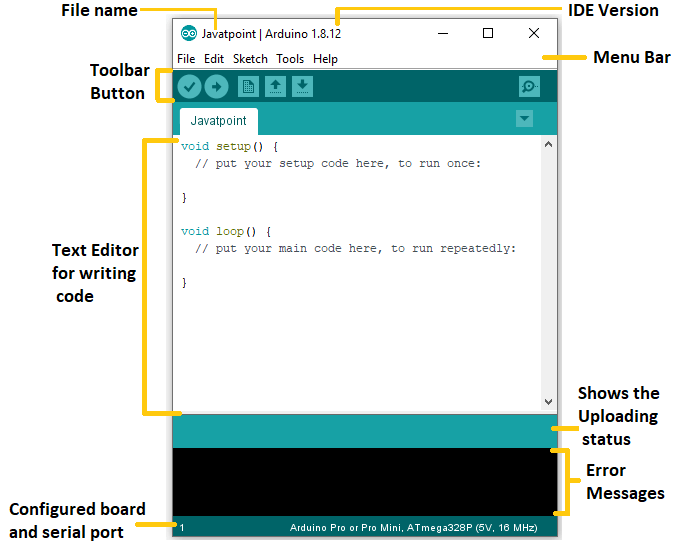


Fig 4.1. Arduino IDE Screen

Each section of the Arduino IDE display is detailed below.

**Toolbar Button**

The icons displayed on the toolbar are **New, Open, Save, Upload,** and **Verify**.

It is shown below:

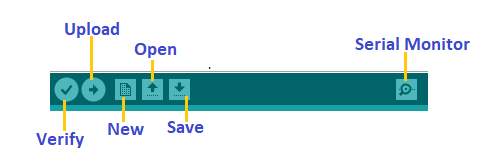


Fig 4.2. Toolbar Button

**Upload**

The Upload button compiles and runs our code written on the screen. It further uploads the code to the connected board. Before uploading the sketch, we need to make sure that the correct board and ports are selected.

We also need a USB connection to connect the board and the computer. Once all the above measures are done, click on the Upload button present on the toolbar.

The latest Arduino boards can be reset automatically before beginning with Upload. In the older boards / certain ESP32 MCU, we need to press the Reset button present on it. As soon as the uploading is done successfully, we can notice the execution of the program example: The Blink of the LED.

If the uploading is failed, it will display the message in the error window.

We do not require any additional hardware to upload our sketch using the Arduino Bootloader.

A Bootloader is defined as a small program, which is loaded in the microcontroller present on the board.

**Open**

The Open button is used to open the already created file. The selected file will be opened in the current window.

**Save**

The save button is used to save the current sketch or code.

**New**

It is used to create a new sketch or opens a new window.

**Verify**

The Verify button is used to check the compilation error of the sketch or the written code.

**Serial Monitor**

The serial monitor button is present on the right corner of the toolbar. It opens the serial monitor.

It is shown below:

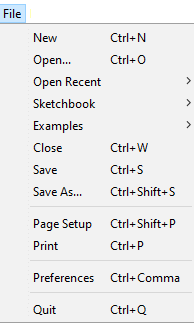
Arduino IDE

When we connect the serial monitor, the board will reset on the operating system Windows, Linux, and Mac OS X. If we want to process the control characters in our sketch, we need to use an external terminal program. The terminal program should be connected to the COM port, which will be assigned when we connect the board to the computer.

**Menu Bar in IDE**

**File Menu**

When we click on the File button on the Menu bar, a drop-down list will appear. It is shown below:



Let's discuss each option in detail.

**New**

The New button opens the new window. It does not remove the sketch which is already present.

**Open**

It allows opening the sketch, which can be browsed from the folders and computer drivers.

**Open Recent**

The Open Recent button contains the list of the recent sketches.

**Sketchbook**

It stores the current sketches created in the Arduino IDE software. It opens the selected sketch or code in a new editor at an instance.

**Examples**

It shows the different examples of small projects for a better understanding of the IDE and the board. The IDE provides examples of self-practice.

**Close**

The Close button closes the window from which the button is clicked.

**Save**

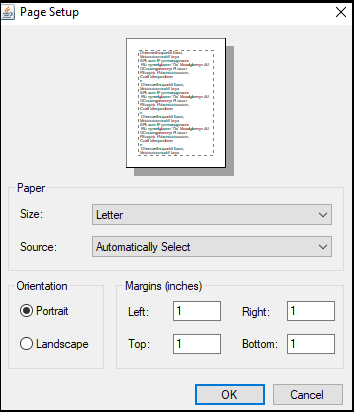
The save button is used to save the current sketch. It also saves the changes made to the current sketch. If we have not specified the name of the file, it will open the '**Save As...'** window.

**Save As**

We can save the sketch with a different name using the '**Save As...'** button. We can also change the name accordingly.

**Page Setup**

It allows setting the page margins, orientation, and size for printing. The '**Page Setup**' window will appear as:



**Print**

According to the settings specified in the 'Page Setup', it prepares the current sketch for printing.

**Preferences**

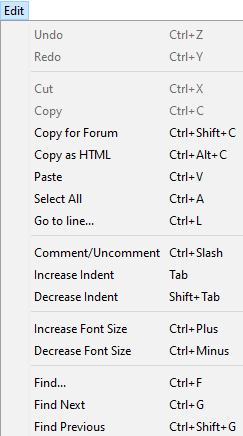
It allows the customization settings of the Arduino IDE.

**Quit**

The Quit button is used to close all the IDE windows. The same closed sketch will be reopened when we will open the Arduino IDE.

**Edit Menu**

When we click on the Edit button on the Menu bar, a drop-down list appears. It is shown below:



Let's discuss each option in detail.

**Undo**

The Undo button is used to reverse the last modification done to the sketch while editing.

**Redo**

The Redo button is used to repeat the last modification done to the sketch while editing.

**Cut**

It allows us to remove the selected text from the written code. The text is further placed to the clipboard. We can also paste that text anywhere in our sketch.

**Copy**

It creates a duplicate copy of the selected text. The text is further placed on the clipboard.

**Copy for Forum**

The 'Copy for Forum' button is used to copy the selected text to the clipboard, which is also suitable for posting to the forum.

**Copy as HTML**

The 'Copy for Forum' button is used to copy the selected text as HTML to the clipboard. It is desirable for embedding in web pages.

**Paste**

The Paste button is used to paste the selected text of the clipboard to the specified position of the cursor.

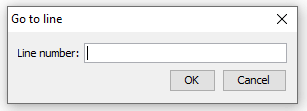
**Select All**

It selects all the text of the sketch.

**Go to line**

It moves the cursor to the specified line number.

The window will appear as:



**Comment/Decomment**

The Comment/ Decomment button is used to put or remove the comment mark (**//**) at the beginning of the specified line.

**Increase Indent**

It is used to add the space at the starting of the specified line. The spacing moves the text towards the right.

**Decrease Indent**

It is used to subtract or remove the space at the starting of the specified line. The spacing moves the text towards the left.

**Increase Font Size**

It increases the font size of the written text.

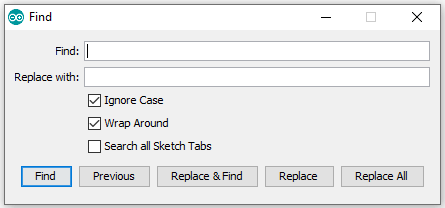
**Decrease Font Size**

It decreases the font size of the written text.

**Find**

It is used to find the specified text. We can also replace the text. It highlights the text in the sketch.

The window will appear as:



**Find Next**

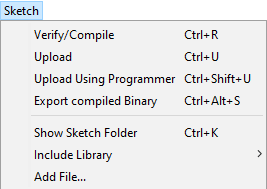
It highlights the next word, which has specified in the '**Find...'** window. If there is no such word, it will not show any highlighted text.

**Find Previous**

It highlights the previous work, which has specified in the '**Find'** window. If there is no such word, it will not show any highlighted text.

**Sketch Menu**

When we click on the Sketch button on the Menu bar, a drop-down list appears. It is shown below:



Let's discuss each option in detail.

**Verify/Compile**

It will check for the errors in the code while compiling. The memory in the console area is also reported by the IDE.

**Upload**

The Upload button is used to configure the code to the specified board through the port.

**Upload Using Programmer**

It is used to override the Bootloader that is present on the board. We can utilize the full capacity of the Flash memory using the '**Upload Using Programmer**' option. To implement this, we need to restore the Bootloader using the **Tools**-> **Burn Bootloader** option to upload it to the USB serial port.

**Export compiled Binary**

It allows saving a .**hex** file and can be kept archived. Using other tools, .hex file can also be sent to the board.

**Show Sketch Folder**

It opens the folder of the current code written or sketch.

**Include Library**

Include Library includes various Arduino libraries. The libraries are inserted into our code at the beginning of the code starting with the #. We can also import the libraries from .zip file.

**Add File...**

The Add File... button is used to add the created file in a new tab on the existing file.

For example, let's add '**Blink**' file to the '**Javatpoint**' file. The tab will now appear as:

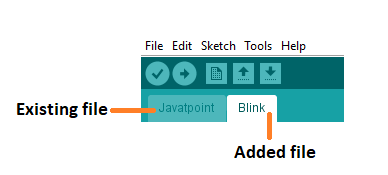
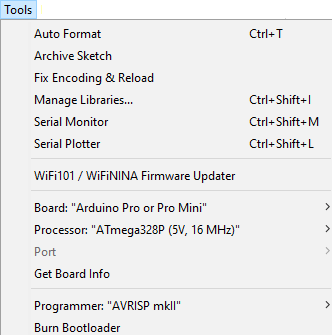


Fig 4.3. Add File

**Tools Menu**

When we click on the Tools button on the Menu bar, a drop-down list appears. It is shown below:



Let's discuss each option in detail.

**Auto Format**

The Auto Format button is used to format the written code. For example, lining the open and closed curly brackets in the code.

**Archive Sketch**

The copy of the current sketch or code is archived in the .zip format. The directory of the archived is same as the sketch.

**Fix Encoding and Reload**

This button is used to fix the inconsistency between the operating system char maps and editor char map encoding.

**Manage Libraries...**

It shows the updated list of all the installed libraries. We can also use this option to install a new library into the Arduino IDE.

**Serial Monitor**

It allows the exchange of data with the connected board on the port.

**Serial Plotter**

The Serial Plotter button is used to display the serial data in a plot. It comes preinstalled in the Arduino IDE.

**WiFi101/WiFiNINA Firmware Updater**

It is used to check and update the Wi-Fi Firmware of the connected board.

**Board**

We are required to select the board from the list of boards. The selected board must be similar to the board connected to the computer.

**Processor**

It displays the processor according to the selected board. It refreshes every time during the selection of the board.

**Port**

It consists of the virtual and real serial devices present on our machine.

**Get Board Info**

It gives the information about the selected board. We need to select the appropriate port before getting information about the board.

**Programmer**

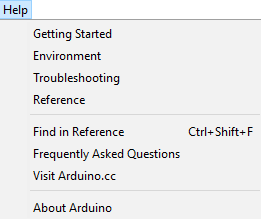
We need to select the hardware programmer while programming the board. It is required when we are not using the onboard USB serial connection. It is also required during the burning of the Bootloader.

**Burn Bootloader**

The Bootloader is present on the board onto the microcontroller. The option is useful when we have purchased the microcontroller without the bootloader. Before burning the bootloader, we need to make sure about the correct selected board and port.

**Help Menu**

When we click on the Help button on the Menu bar, a drop-down list will appear. It is shown below:



The Help section includes several documents that are easy to access, which comes along with the Arduino IDE. It consists of the number of options such as Getting Started, Environment, Troubleshooting, Reference, etc. can be accessed without the internet connection as well. It will directly link us to the official website of Arduino.

**4.2 Why Arduino IDE?**

1. Easy to use: Arduino IDE provides a user-friendly interface for writing, compiling, and uploading code.

2. Cross-platform: Available for Windows, macOS, and Linux.

3. Open source: Free to download, modify, and distribute.

4. Large community: Extensive libraries, tutorials, and forums.

5. Versatile: Supports various microcontrollers, including Arduino, ESP32, and ESP8266.

**4.3 Why Arduino IDE for ESP32/ESP8266?**

1. Simplified development: Arduino IDE streamlines the development process for ESP32/ESP8266.

2. Library support: Access to a vast library collection, including WiFi, Bluetooth, and IoT-specific libraries.

3. Community resources: Leverage ESP32/ESP8266-specific tutorials, examples, and forums.

4. Easy uploading: Upload code directly from Arduino IDE to ESP32/ESP8266 boards.

5. Cost-effective: No need for expensive development tools or software.

**ESP32-specific benefits**:

1. Dual-core processing: Arduino IDE supports ESP32's dual-core architecture.

2. Wi-Fi and Bluetooth: Easily integrate Wi-Fi and Bluetooth capabilities.

3. Low-power consumption: Optimize power usage with Arduino IDE's built-in features.

4. Advanced peripherals: Access ESP32's advanced peripherals, such as SPI, I2C, and UART.

**ESP8266-specific benefits:**

1. WiFi integration: Easily integrate WiFi capabilities.

2. Low-cost: ESP8266 is an affordable board.

3. Small footprint: Ideal for IoT projects requiring compact design.

4. Community support: Extensive ESP8266 community and resources.

**Comparison with other IDEs:**

1. Platform IO: More complex but offers advanced features and multi-platform support.

2. Visual Studio Code: More versatile but requires additional setup and configuration.

3. ESP-IDF: Official ESP32/ESP8266 IDE, but steeper learning curve.

Arduino IDE provides an ideal balance of ease, versatility, and community support, making it a popular choice for ESP32 and ESP8266 development.

**4.4 Step by Step Installation Procedure**

In this section, we will learn in easy steps, how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

The Arduino IDE 2 is an open-source project. It is a big step from its sturdy predecessor, Arduino IDE 1.x, and comes with revamped UI, improved board & library manager, debugger, autocomplete feature and much more.

**Step 1** – Download and install the Arduino IDE 2 on your Windows. You can easily download the editor by visit the Arduino Software page: <https://www.arduino.cc/en/software>

**Requirements**

* Windows - Win 10 and newer, 64 bits
* Linux - 64 bits
* macOS - Version 10.15: "Catalina" or newer, 64 bits

**For windows installation**: In Arduino IDE 2.3.3 select Win 10 and newer, 64 bits

A screenshot of a computer

Description automatically generated

Fig 4.4. Arduino Software page

To install the Arduino IDE 2 on a Windows computer, simply run the file downloaded from the software page.

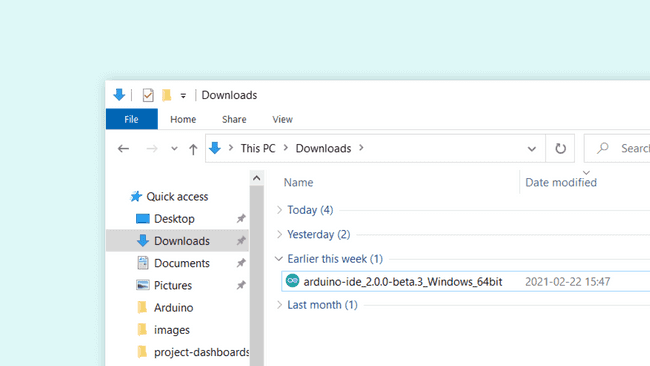
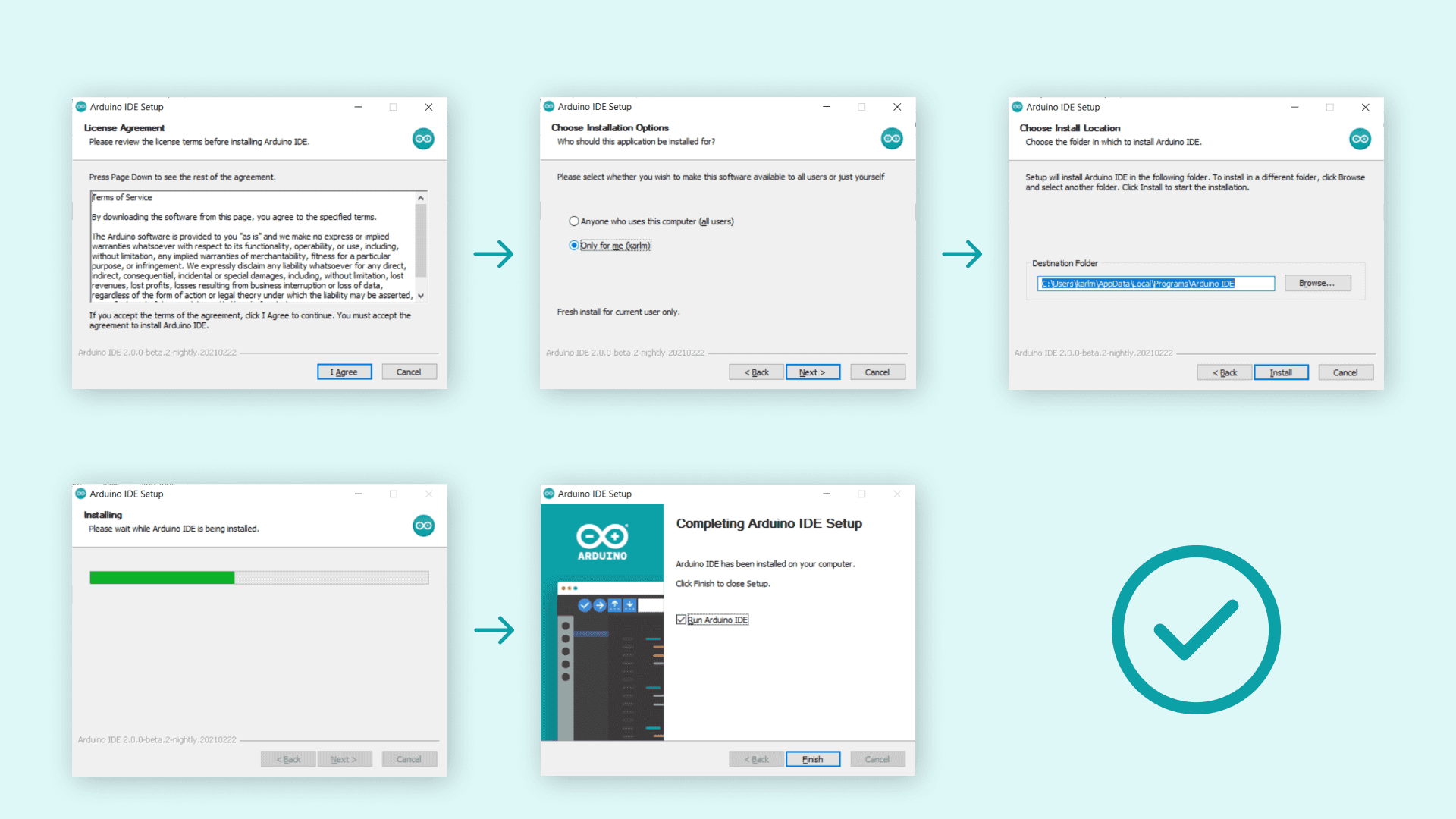


Fig 4.5. Running the Executable Arduino IDE 2 installation file

Follow the instructions in the installation guide as shown in the image. The installation may take several minutes.



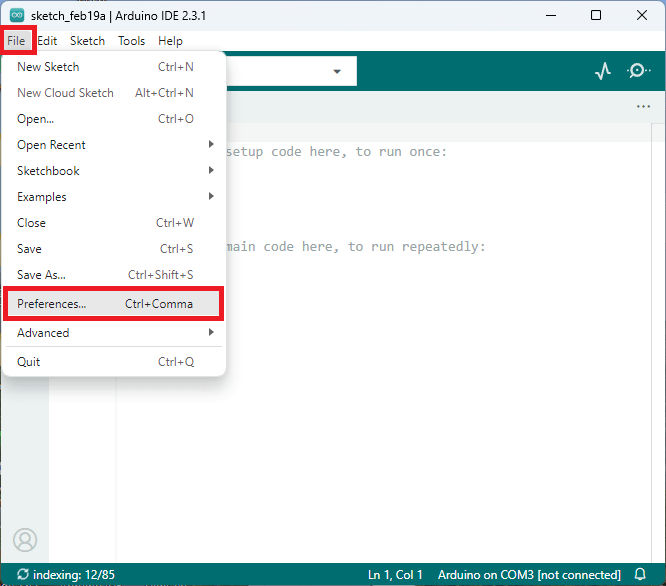
**Installing ESP32 Board in Arduino IDE 2**

Before proceeding make sure you have completed above process that is  [Arduino IDE 2](https://www.arduino.cc/en/Tutorial/getting-started-with-ide-v2) installed on your computer and follow the instructions given below

**Install ESP32 Add-on in Arduino IDE**

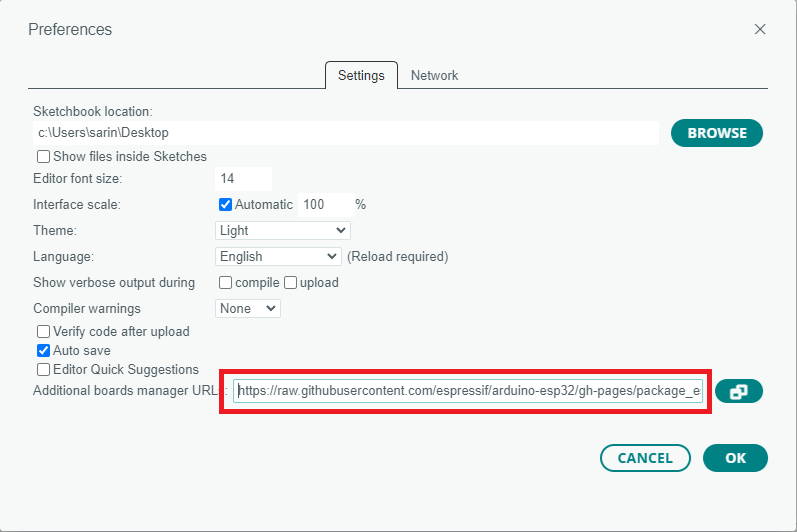
To install the ESP32 board in your Arduino IDE, follow these next instructions:

**1.**In your Arduino IDE 2, go to **File**> **Preferences**.



**2.** Copy and paste the following line to the **Additional Boards Manager** URLs field.

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

 Fig 4.6. Additional Boards Manager URLs field.

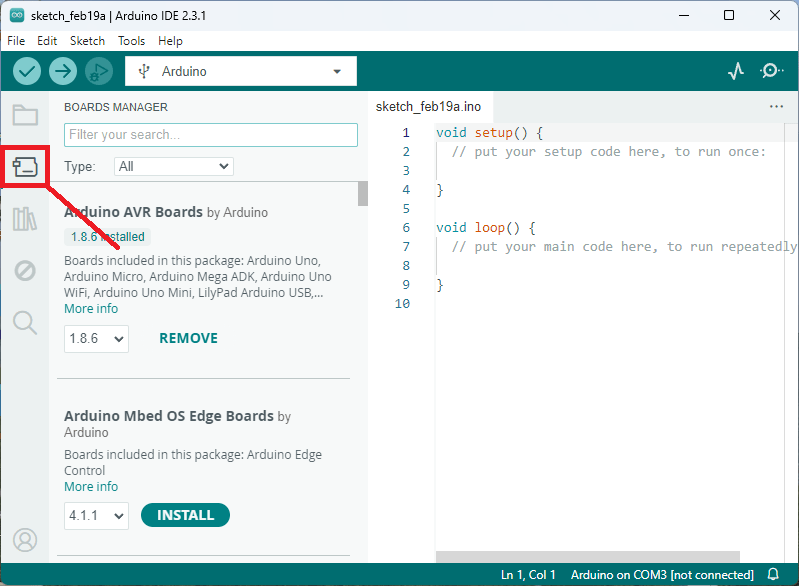
**Note:** if you already have the ESP8266 boards URL, you can separate the URLs with a comma, as follows:

<http://arduino.esp8266.com/stable/package_esp8266com_index.json>,

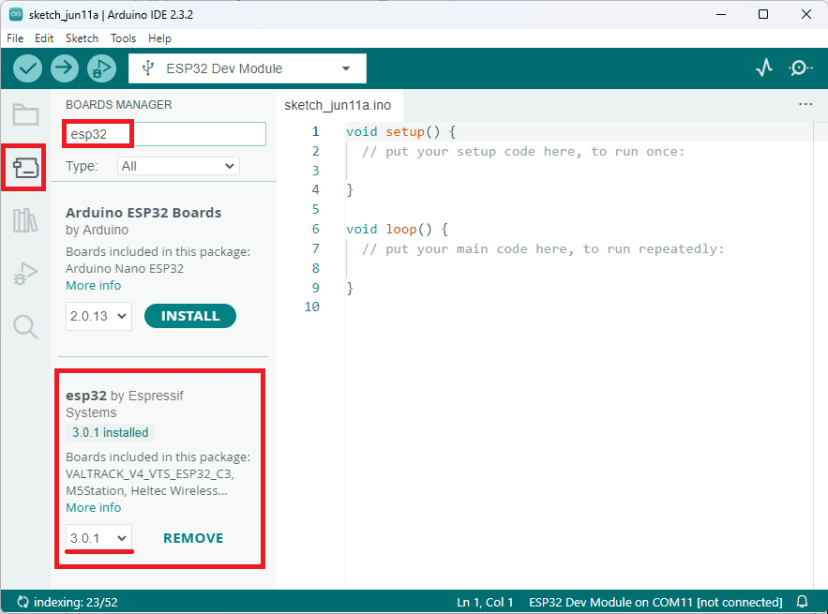
<https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json>

3. Open the Boards Manager. You can go to **Tools**> **Board**> **Boards Manager…** or you

can simply click the **Boards Manager**icon in the left-side corner.



4. Search for **ESP32**and press install button for **esp32 by Espressif Systems version 3.X**.



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

**Testing the Installation**

To test the ESP32 add-on installation, upload a simple code that blinks the on-board LED (GPIO 2).

Write the following code to your Arduino IDE:

A screenshot of a computer

Description automatically generated

**Uploading the Sketch**

Select your board before uploading the code. On the top drop-down menu, click on “**Select other board and port…**”

[A screen shot of a computer

Description automatically generated](https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2024/02/arduino-ide-2-select-board.png?quality=100&strip=all&ssl=1)

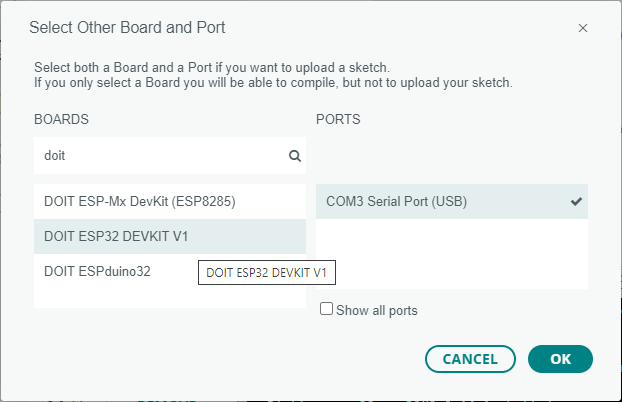
**Uploading the Sketch**

Select your board before uploading the code. On the top drop-down menu, click on “**Select other board and port…**”

[A screen shot of a computer

Description automatically generated](https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2024/02/arduino-ide-2-select-board.png?quality=100&strip=all&ssl=1)

A new window, as shown below, will open. Search for your ESP32 board model.

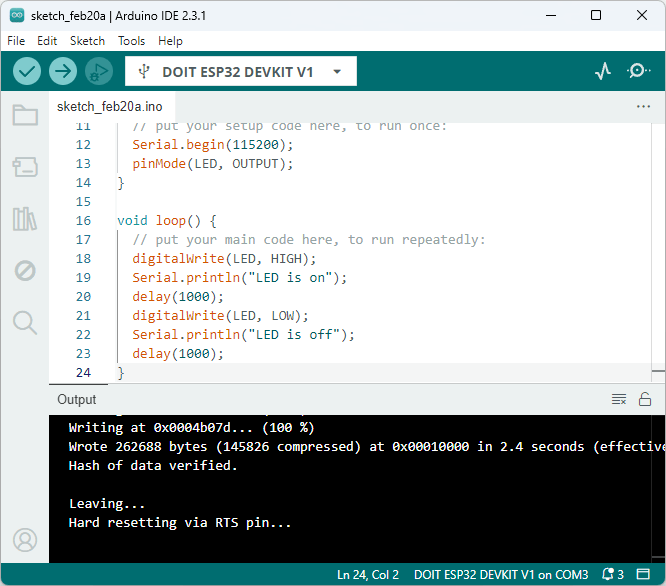
[](https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2021/05/arduino-ide-2-select-board-esp32.png?quality=100&strip=all&ssl=1)Fig 4. 7. Uploading the Sketch

Select the ESP32 board model you’re using, and the COM port. In our example, we’re using the DOIT ESP32 DEVKIT V1. Click **OK** when you’re done.

Now, you just need to click on the **Upload**button shown below.

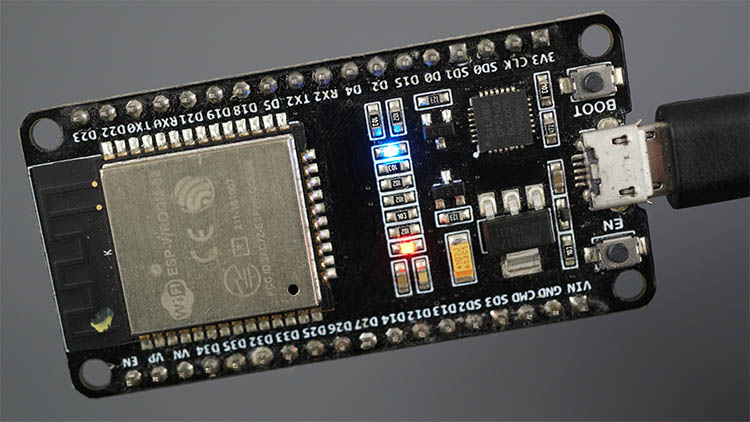
Arduino IDE 2 Upload Button

After a few seconds, the upload should be complete.



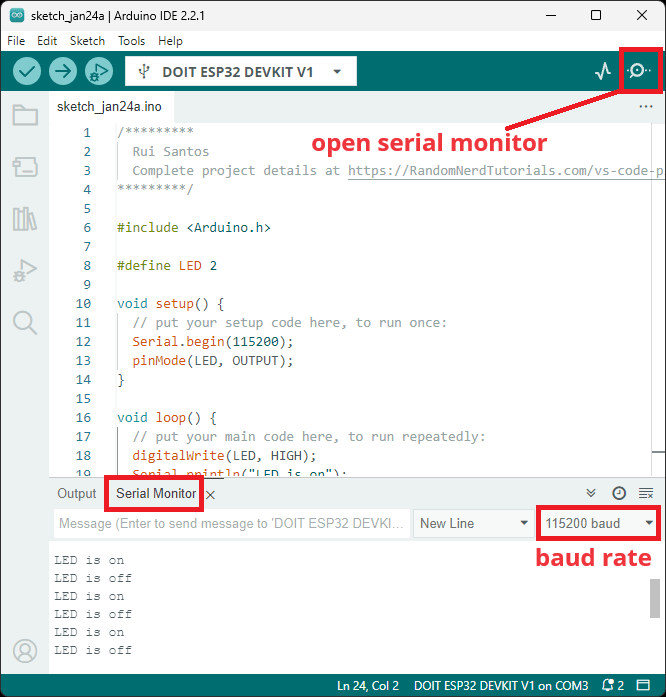
**Note**: some ESP32 development boards don’t go into flashing/uploading mode automatically when uploading a new code and you’ll see a lot of dots on the debugging window followed by an error message. If that’s the case, you need to press the ESP32 BOOT button when you start seeing the dots on the debugging window.

The ESP32 on-board LED should be blinking every second.

[](https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2020/04/ESP32-board-Built_in-LED-turned-on-HIGH.jpg?quality=100&strip=all&ssl=1)

**Serial Monitor**

You can click on the Serial Monitor icon to open the Serial Monitor tab. Make sure you select the 115200 baud rate.



That’s it! You’ve installed the ESP32 Boards successfully in Arduino IDE 2.

**Troubleshooting**

If you try to upload a new sketch to your ESP32 and you get this error message “*A fatal error occurred: Failed to connect to ESP32: Timed out… Connecting…*”. It means that your ESP32 is not in flashing/uploading mode.

Having the right board name and COM port selected, follow these steps:

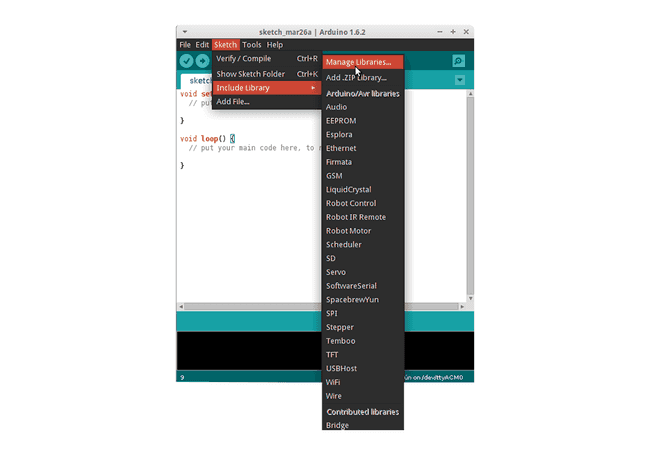
* Hold-down the **BOOT** button in your ESP32 board
* Press the **Upload**button in the Arduino IDE to upload your sketch
* After you see the “*Connecting….*” message in your Arduino IDE, release the finger from the **BOOT**button
* After that, you should see the “*Done uploading*” message

**Installing Libraries**

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the [LiquidCrystal library](https://www.arduino.cc/reference/en/libraries/liquidcrystal/) makes it easy to talk to character LCD displays. There are thousands of libraries available for download directly through the Arduino IDE, and you can find all of them listed at the [Arduino Library Reference](https://www.arduino.cc/reference/en/libraries/).

**Using the Library Manager**

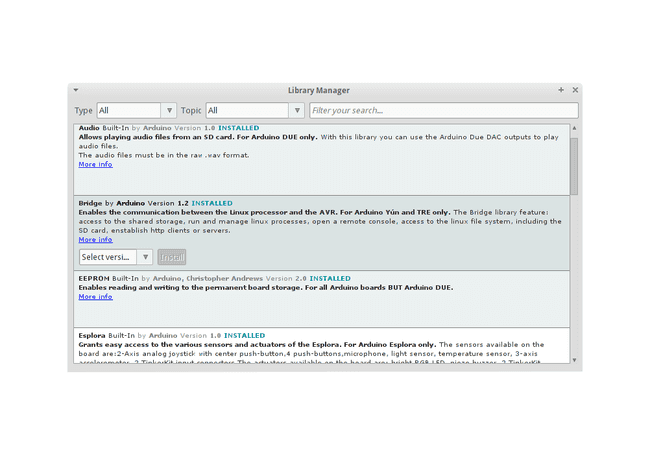
To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.6.2). Open the IDE and click to the "Sketch" menu and then *Include Library > Manage Libraries*.



Then the Library Manager will open, and you will find a list of libraries that are already installed or ready for installation. In this example we will install the Bridge library. Scroll the list to find it, click on it, then select the version of the library you want to install. Sometimes only one version of the library is available. If the version selection menu does not appear, don't worry it is normal.



Finally click on install and wait for the IDE to install the new library. Downloading may take time depending on your connection speed. Once it has finished, an *Installed* tag should appear next to the Bridge library. You can close the library manager.

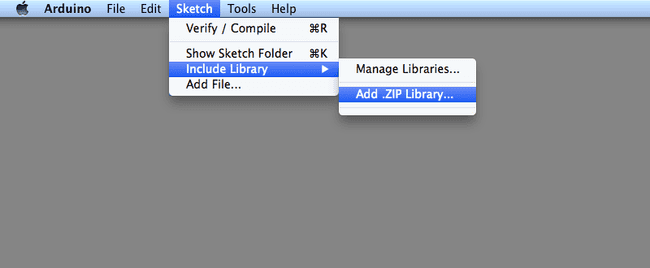


You can now find the new library available in the *Sketch > Include Library* menu. If you want to add your own library to Library Manager, follow [these instructions](https://github.com/arduino/library-registry#adding-a-library-to-library-manager).

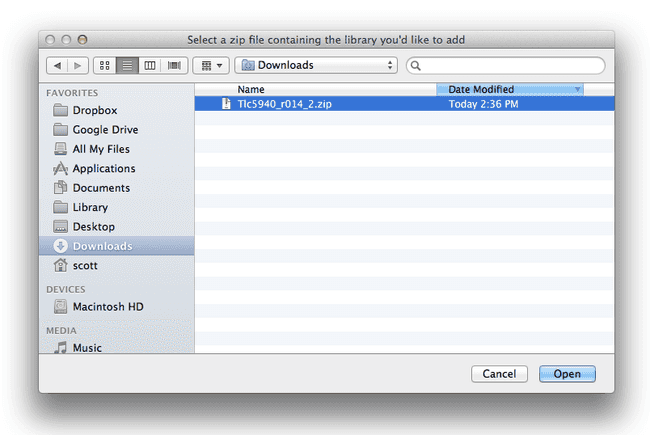
**Importing a .zip Library**

Libraries are often distributed as a ZIP file or folder. The name of the folder is the name of the library. Inside the folder will be a .cpp file, a .h file and often a keywords.txt file, examples folder, and other files required by the library. Starting with version 1.0.5, you can install 3rd party libraries in the IDE. Do not unzip the downloaded library, leave it as is.

In the Arduino IDE, navigate to *Sketch > Include Library > Add .ZIP Library*. At the top of the drop-down list, select the option to "Add .ZIP Library''.



You will be prompted to select the library you would like to add. Navigate to the .zip file's location and open it.



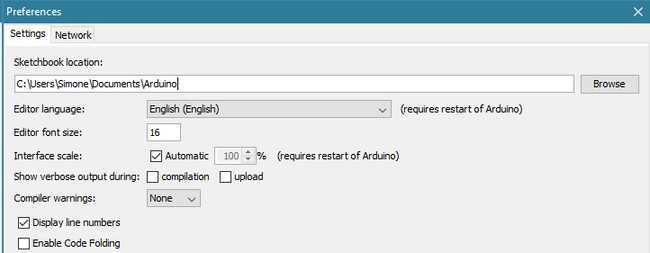
Return to the *Sketch > Include Library menu.* menu. You should now see the library at the bottom of the drop-down menu. It is ready to be used in your sketch. The zip file will have been expanded in the *libraries* folder in your Arduino sketches directory.

Note: the library will be available to use in sketches, but with older IDE versions examples for the library will not be exposed in the *File > Examples* until after the IDE has restarted.

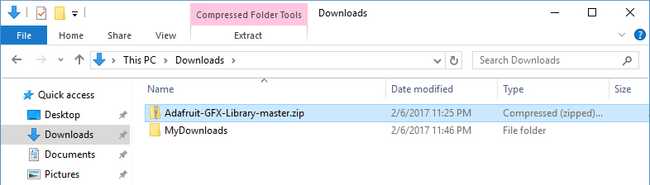
**Manual Installation**

When you want to add a library manually, you need to download it as a ZIP file, expand it and put in the proper directory. The ZIP file contains all you need, including usage examples if the author has provided them. The library manager is designed to install this ZIP file automatically as explained in the former chapter, but there are cases where you may want to perform the installation process manually and put the library in the *libraries* folder of your sketchbook by yourself.

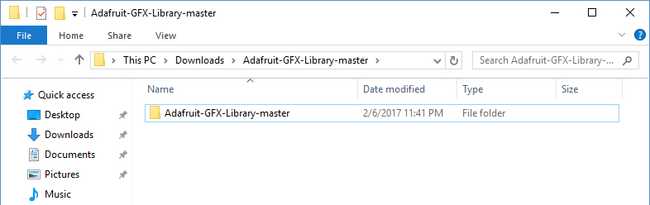
You can find or change the location of your sketchbook folder at *File > Preferences > Sketchbook* location.



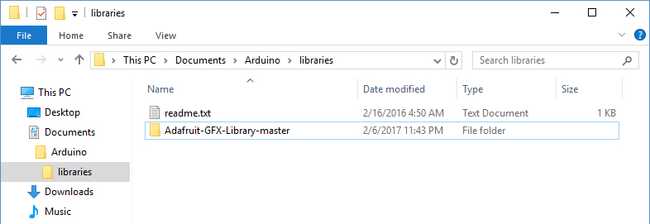
Go to the directory where you have downloaded the ZIP file of the library



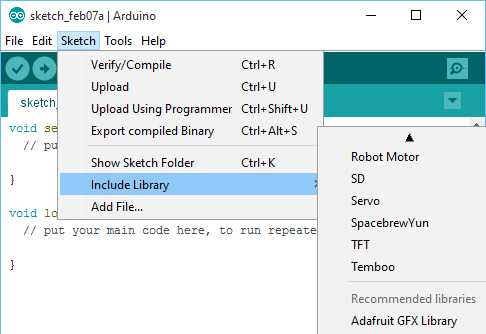
Extract the ZIP file with all its folder structure in a temporary folder, then select the main folder, that should have the library name



Copy it in the "libraries" folder inside your sketchbook.



Start the Arduino Software (IDE), go to *Sketch > Include Library*. Verify that the library you just added is available in the list.



Please note: Arduino libraries are managed in three different places: inside the IDE installation folder, inside the core folder and in the libraries folder inside your sketchbook. The way libraries are chosen during compilation is designed to allow the update of libraries present in the distribution. This means that placing a library in the "libraries" folder in your sketchbook overrides the other libraries versions.

**Blynk Library Installation**

To install a new library Blynk into your Arduino IDE, you can use the Library Manager.

First, connect your computer o the Internet. Open the IDE and click to the “Sketch” menu and then Include Library > Manage Libraries.

**A screenshot of a computer

Description automatically generated**

Fig 4.8. Blynk Library Installation

Then the Library Manager will open and you will find a list of libraries that are already installed or ready for installation. Search for Blynk library and in the version selection choose the latest version to date.

**A screenshot of a computer

Description automatically generated**

Finally click on Install and wait for the IDE to install the new library. Downloading may take time depending on your connection speed. Once it has finished, an Installed tag should appear next to the Bridge library. You can close the library manager.

**CHAPTER 5**

1. **IOT CLOUD & SERVER PLATFORM**

**What is the internet of things (IoT)?**

The internet of things, or IoT, is a network of interrelated devices that connect and exchange data with other IoT devices and the cloud. IoT devices are typically embedded with technology such as sensors and software and can include mechanical and digital machines and consumer objects.

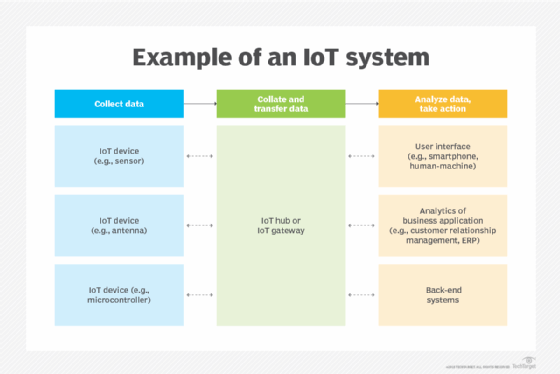
These devices encompass everything from everyday household items to complex industrial tools. Increasingly, organizations in a variety of industries are using IoT to operate more efficiently, deliver enhanced customer service, improve decision-making and increase the value of the business.

With IoT, data is transferable over a network without requiring human-to-human or human-to-computer interactions.

A thing in the internet of things can be a person with a heart monitor implant, a farm animal with a biochip transponder, an automobile that has built-in sensors to alert the driver when tire pressure is low, or any other natural or man-made object that can be assigned an Internet Protocol address and can transfer data over a network.

**How does IoT work?**

IoT systems function by gathering data from sensors embedded in IoT devices, which is then transmitted through an IoT gateway for analysis by an application or back-end system



*Fig 5.1. Example of IoT system*

The following four elements are incorporated into an IoT ecosystem for it to function:

**Sensors or devices**

An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware to collect, send and act on data acquired from their environments.

**Connectivity**

IoT devices can communicate with one another through a network over the internet. These devices share sensor data by connecting to an IoT gateway, which acts as a central hub where IoT devices can send data. Before the data is shared, it can also be sent to an edge device where it is analyzed locally.

**Data analysis**

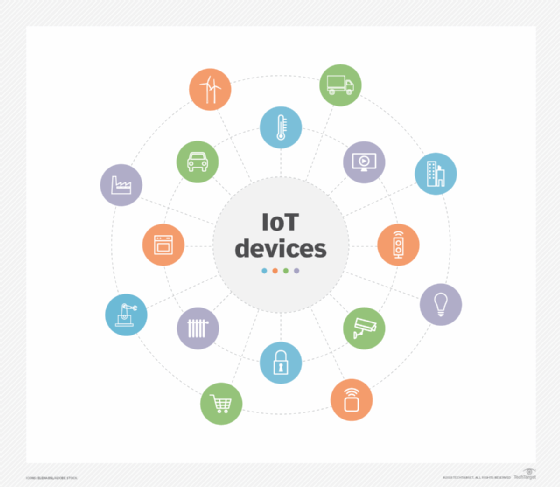
Only the relevant data is used to identify patterns, offer recommendations and identify potential issues before they escalate. Analyzing data locally reduces the volume of data sent to the cloud, which minimizes bandwidth consumption.

Sometimes, these devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices. For example, they can set them up, give them instructions or access the data. The connectivity, networking and communication protocols used with these web-enabled devices largely depend on the specific IoT applications deployed.

IoT can also use artificial intelligence and machine learning to make data collection processes easier and more dynamic.

**Graphical user interface**

A graphical user interface ([UI](https://www.techtarget.com/searchapparchitecture/definition/user-interface-UI)) is typically used to manage IoT devices. For example, a website or a mobile app can be used as an UI to manage, control and register smart devices.



Consumer, enterprise and industrial IoT devices include smart TVs and smart sensors outfitted for conference rooms and assembly line machines.

**IoT applications include the following**:

* **Agriculture.**IoT can benefit farmers by making their jobs easier. For example, sensors can collect data on rainfall, humidity, temperature and soil content and IoT can help automate farming techniques. Additionally, IoT devices can be used to oversee the health of livestock, monitor equipment and streamline supply chain management.
* **Construction.** IoT can help monitor operations surrounding infrastructure. Sensors, for example, can monitor events or changes within structural buildings, bridges and other infrastructure that could potentially compromise safety. This provides benefits such as improved incident management and response, reduced operations costs and improved service quality.
* **Home automation.** A home automation business can use IoT to monitor and manipulate mechanical and electrical systems in a building. Homeowners can also remotely control and automate their home environment by using IoT devices, including smart thermostats, lighting systems, security cameras and voice assistants such as Alexa and Siri for increased comfort and energy efficiency.
* **Smart buildings and cities.**Smart cities can help citizens reduce waste and energy consumption. They can reduce energy costs using sensors that detect how many occupants are in a room and turning the air conditioner on if sensors detect a conference room is full or lowering the heat if everyone in the office has gone home.
* **Urban consumption systems.** IoT technologies can also be used to monitor and manage urban consumption such as traffic lights, parking meters, waste management systems and public transportation networks.
* **Healthcare monitoring.**IoT devices such as remote patient monitoring systems, smart medical devices and medication trackers let healthcare providers monitor patients' health status, manage chronic conditions and provide timely interventions. IoT gives providers the ability to monitor patients more closely by analyzing the generated data. Hospitals also often use IoT systems to complete tasks such as inventory management for both pharmaceuticals and medical instruments.
* **Retail.** IoT sensors and beacons in retail stores can track customer movement, analyze shopping patterns, manage inventory levels and personalize marketing messages. This enhances the shopping experience for customers and optimizes store operations.
* **Transportation.** IoT devices help the transportation industry by monitoring vehicle performance, optimizing routes and tracking shipments. For example, the fuel efficiency of connected cars can be monitored to reduce fuel costs and improve sustainability. IoT devices can also monitor the condition of cargo, so it reaches its destination in optimal condition.
* **Wearable devices.** Wearable devices with sensors and software can collect and analyze user data, sending messages to other technologies about the users to make their lives easier and more comfortable. Wearable devices are also used for public safety -- for example, by improving first responders' response times during emergencies by providing optimized routes to a location or by tracking construction workers' or firefighters' vital signs at life-threatening sites.
* **Energy management.** IoT-enabled smart grids, smart meters and energy management systems let utility companies and consumers monitor and optimize energy usage, manage demand-response programs and integrate renewable energy sources more efficiently. For example, the data collected by the IoT devices and sensors helps identify patterns, peak usage times and areas of inefficiency.

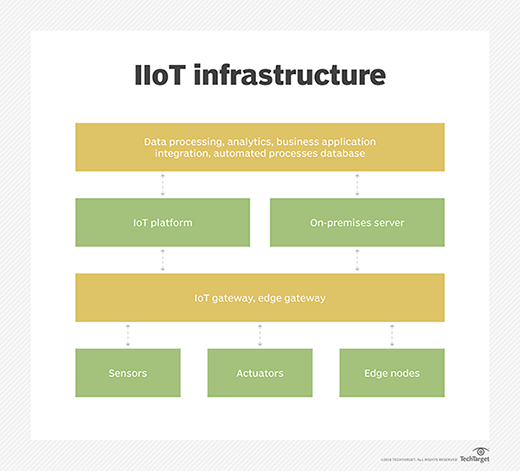


Fig 5.2. IoT Infrastructure

**What Is IoT Cloud?**

An IoT cloud is a massive network that supports IoT devices and applications. This includes the underlying infrastructure, servers and storage, needed for real-time operations and processing. An IoT cloud also includes the services and standards necessary for connecting, managing, and securing different IoT devices and applications.

**Why IoT Cloud?**

IoT clouds offer an efficient, flexible, and scalable model for delivering the infrastructure and services needed to power IoT devices and applications for businesses with limited resources. IoT clouds offer on-demand, cost-efficient hyperscale so organizations can leverage the significant potential of IoT without having to build the underlying infrastructure and services from scratch.

**Servers**

Servers in IoT connect physical devices to the internet, allowing them to communicate and share data. They can be used to monitor and control devices remotely, and to allow devices to interact with each other autonomously.

Here are some types of servers used in IoT:

* **IoT Data Server**

A data integration controller that can be used in factories, production lines, and cloud systems. It has standard data management functions for collection, processing, saving, noticing, and publishing.

* **IoT web server**

A platform that connects devices to the internet, allowing them to communicate and share data. It can be used to monitor and control devices remotely, and to allow devices to interact with each other autonomously.

* **Windows Server IoT**

A full version of Windows Server that provides enterprise manageability and security to IoT solutions.

* **Embedded web server**

A small, lightweight web server that can be integrated into other software or hardware systems. It's used to provide a web-based interface for monitoring and controlling devices.

* **Network server**

A server that plays a crucial role in IoT ecosystems by managing network resources and device communications. It allows IoT networks to scale up to support thousands or even millions of devices.

**Message Queue Telemetry Transport Protocol (MQTT)**

Message Queuing Telemetry Transport, or MQTT, is a communications protocol designed for Internet of Things devices with extremely high latency and restricted low bandwidth. Message Queuing Telemetry Transport is a perfect protocol for machine-to-machine (M2M) communication since it is designed specifically for low-bandwidth, high-latency settings.

**What is Message Queue Telemetry Transport Protocol (MQTT)?**

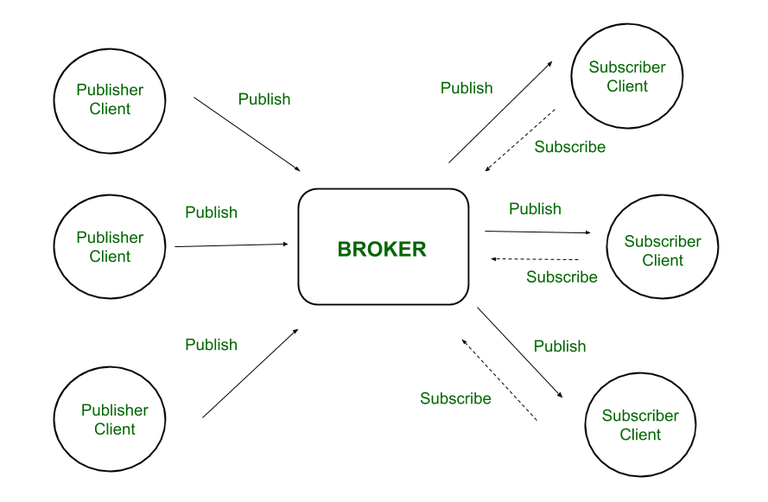
**MQTT** is a simple, lightweight messaging protocol used to establish communication between multiple devices. It is a TCP-based protocol relying on the publish-subscribe model. This communication protocol is suitable for transmitting data between resource-constrained devices having low bandwidth and low power requirements. Hence this messaging protocol is widely used for communication in the IoT Framework.

**Publish-Subscribe Model**

This model involves multiple clients interacting with each other, without having any direct connection established between them. All clients communicate with other clients only via a third party known as a Broker.

**MQTT Client and Broker**

Clients publish messages on different topics to brokers. The broker is the central server that receives these messages and filters them based on their topics. It then sends these messages to respective clients that have subscribed to those different topics. The heart of any publish/subscribe protocol is the MQTT broker. A broker can handle up to thousands of concurrently connected MQTT customers, depending on how it is implemented. All communications must be received by the broker, who will then sort them, ascertain who subscribed to each one, and deliver the messages to the clients who have subscribed. All persistent customers’ sessions, including missed messages and subscriptions, are likewise kept by the Broker.



*Fig 5.3. Publish-Subscribe Model*

Hence client that has subscribed to a specific topic receives all messages published on that topic.

Here the broker is central hub that receives messages, filters them, and distributes them to appropriate clients, such that both message publishers, as well as subscribers, are clients.

**Working of MQTT**

MQTT’s publish/subscribe (pub/sub) communication style, which aims to maximise available bandwidth, is an alternative to conventional client-server architecture that communicates directly with an endpoint. In contrast, the client who transmits the message (the publisher) and the client or clients who receive it (the subscribers) are not connected in the pub/sub paradigm. Third parties—the brokers—manage the relationships between the publishers and subscribers because they don’t communicate with one another directly.

Publishers and subscribers, which denote whether a client is publishing messages or has subscribed to receive messages, are examples of MQTT clients. The same MQTT client can be used to accomplish these two features. A publish occurs when a client or device want to submit data to a server or broker.

The term “subscribe” refers to the reversal of the procedure. Several clients can connect to a broker under the pub/sub paradigm and subscribe to subjects that interest them.

A screenshot of a computer

Description automatically generated

*Fig 5.4. Working of MQTT*

When a broker and a subscribing client lose contact, the broker will store messages in a buffer and send them to the subscriber whenever the broker is back up and running. The broker has the right to cut off communication with subscribers and send them a cached message containing publisher instructions if the publishing client abruptly disconnects from the broker.

“Publishers send the messages, subscribers receive the messages they are interested in, and brokers pass the messages from the publishers to the subscribers,” reads an IBM write-up describing the pub/sub paradigm. MQTT clients, such as publishers and subscribers, can only speak with MQTT brokers. Any device or programme that runs a MQTT library can be a MQTT client, ranging from microcontrollers like the Arduino to entire application servers housed in the cloud.

**Characteristics of MQTT**

* **Lightweight:**MQTT is designed to be lightweight, making it suitable for use in aid-restrained environments inclusive of embedded systems and low-strength devices. The protocol minimizes bandwidth and processing overhead, enabling green communication even on restricted networks.
* **Publish-Subscribe Model:**In the publish-subscribe version, clients (publishers) send messages to subjects, and different clients (subscribers) acquire messages from subjects of interest. This decoupling of producers and purchasers permits for flexible and dynamic conversation styles.
* **Quality of Service (QoS) Levels:**MQTT supports exclusive stages of message delivery warranty, referred to as Quality of Service (QoS). QoS levels range from 0 to 2, providing various stages of reliability and message transport guarantees, relying at the utility necessities.
* **Retained Messages:**MQTT lets in agents to store retained messages on topics, making sure that new subscribers acquire the maximum latest message posted on a subject right now after subscribing. This characteristic is beneficial for fame updates and configuration settings.
* **Last Will and Testament (LWT):** MQTT clients can specify a Last Will and Testament message to be posted by way of the broker in the occasion of an sudden consumer disconnect. This function affords a mechanism for detecting patron failures and dealing with them gracefully.
* **Security:**MQTT helps various protection mechanisms, consisting of Transport Layer Security (TLS) encryption and authentication mechanisms which include username/password and consumer certificates. These capabilities make certain the confidentiality, integrity, and authenticity of messages exchanged over MQTT connections.

**Advantages of MQTT**

This model is not restricted to one-to-one communication between clients. Although the publisher client sends a single message on specific topic, broker sends multiple messages to all different clients subscribed to that topic. Similarly, messages sent by multiple such publisher clients on multiple different topics will be sent to all multiple clients subscribed to those topics. Hence one-to-many, many-to-one, as well as many-to-many communication is possible using this model. Also, clients can publish data and at the same time receive data due to this two-way communication protocol. Hence MQTT is considered to be bi-directional protocol. The default unencrypted MQTT port used for data transmission is 1883. The encrypted port for secure transmission is 8883.

* Lightweight protocol that is quick to create and allows for efficient data transport
* Minimal data packet usage, resulting in low network usage
* Effective data dispersion
* The effective use of remote sensing and control
* Prompt and effective message delivery
* Minimizes power consumption, which is beneficial for the linked devices, and maximizes network capacity.
* Data transmission is quick, efficient, and lightweight because MQTT messages have small code footprint. These control messages have a fixed header of size 2 bytes and payload message up to size 256 megabytes.

**Introduction about Blynk IoT**

Blynk is a software platform that allows users to connect their hardware to the cloud and manage connected devices remotely. Blynk is designed for the Internet of Things (IoT) and can be used for personal projects or commercial products.

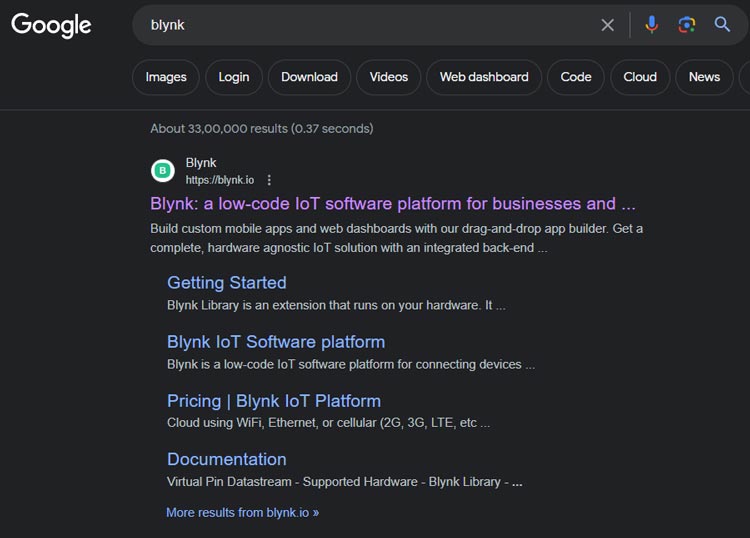
Here are some features of Blynk:

* No-code approach: Blynk allows users to build and scale IoT without coding. For example, users can control an LED or motor from their phone with just a few clicks.
* Remote management: Users can remotely control devices from anywhere, receive notifications, and analyze data.
* Multi-tenancy: Users can configure roles and permissions to control who has access to data.
* Scalability: Blynk can handle thousands of devices and is used by businesses of all sizes.
* Open-source: Blynk's server is open-source and can be run locally or on Blynk Cloud.
* Blynk App: Users can create interfaces for their projects using widgets.
* Blynk Libraries: Users can communicate with the server and process commands for popular hardware platforms.

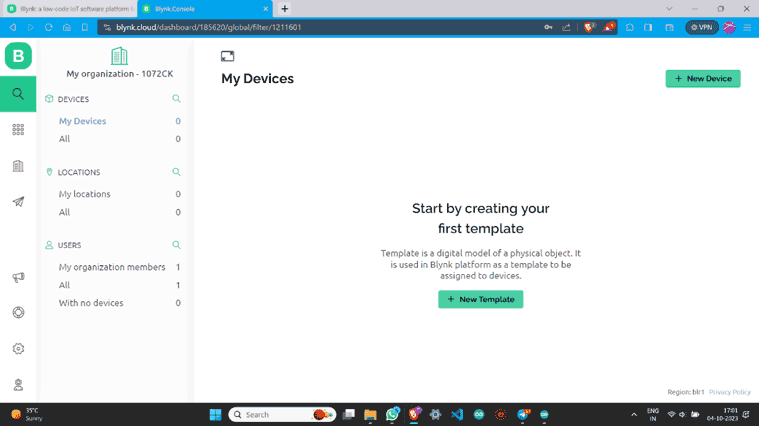
Blynk is free for personal use and prototyping. Businesses can purchase subscriptions to publish Blynk-powered apps for their products.

**Setup Blynk 2.0 App**

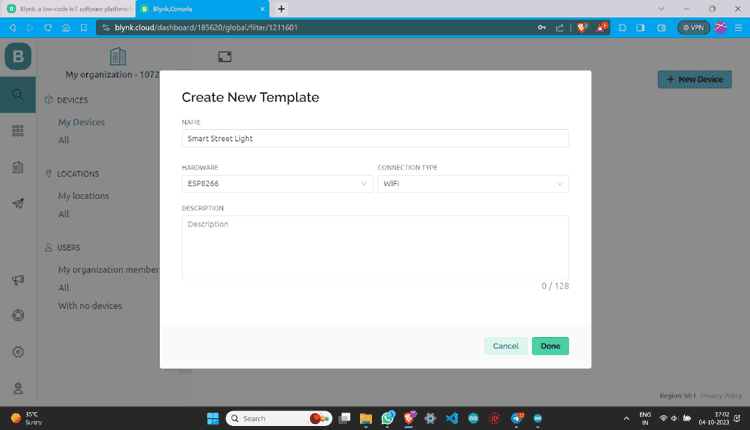
Step 1: Go to the Blynk website and click the "Sign Up" button. Enter your email address and create a password.

****

Step 2: Once you have signed up on the Blynk and logged in, tap the "+ New Template" button to create a new template.

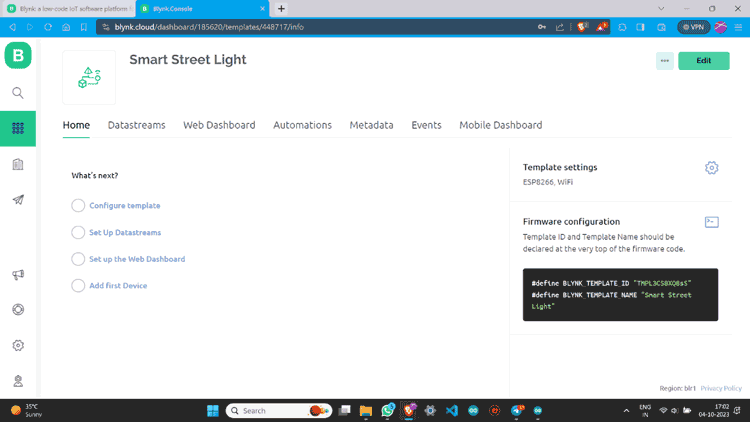
****

Step 3: Name the template, select the Development Board, set WiFi as the connection type and then save this template.

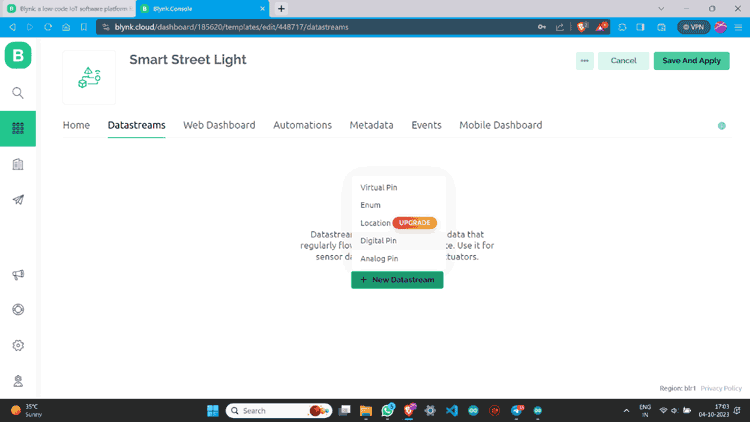
****

Step 4:  Once a new template is created you will see the below screen on your blynk account

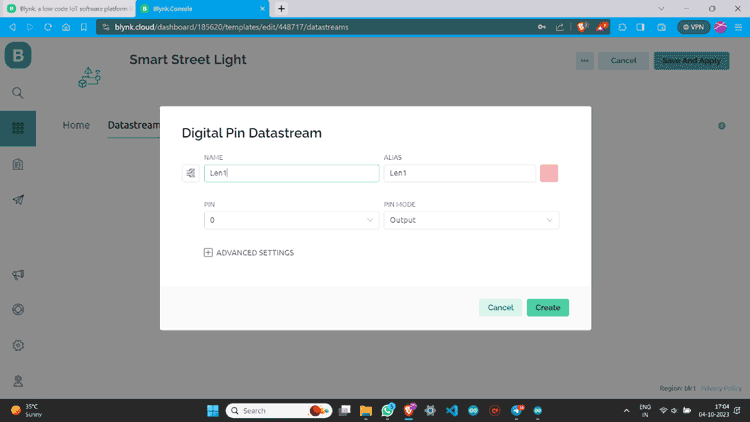
.

****

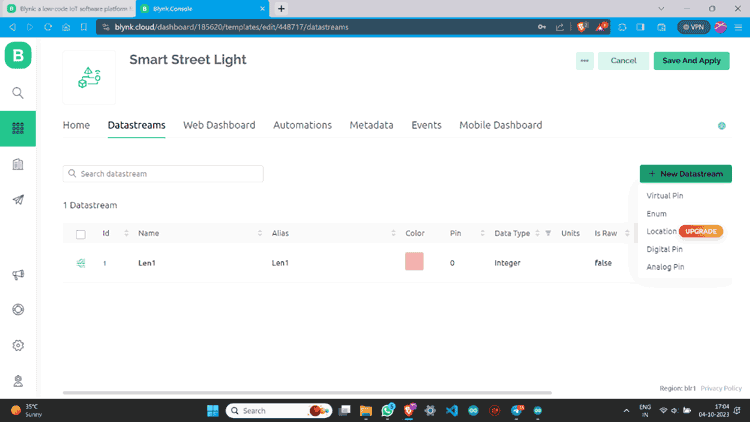
Step 5:  Go to Datastreams and click on “+ New Datastream”. Then Choose “Digital Pin”.

****

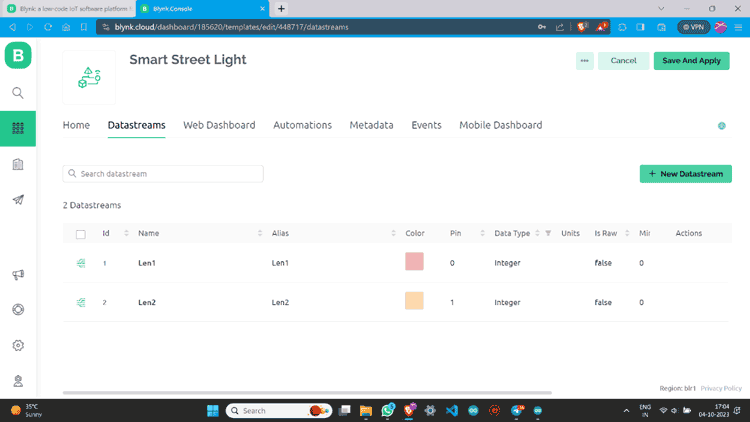
Step 6: Name Datastream and select pin. Assign it’s PIN MODE as Output and Create.

****

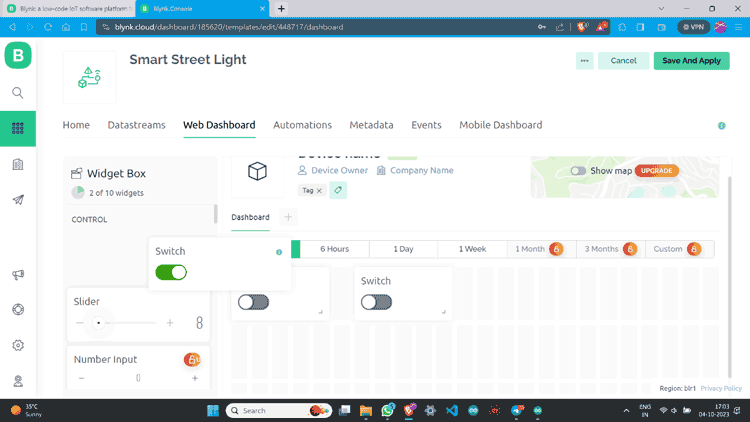
Step 7: Once a datastream is created on the blynk website, it will look something like this below

****

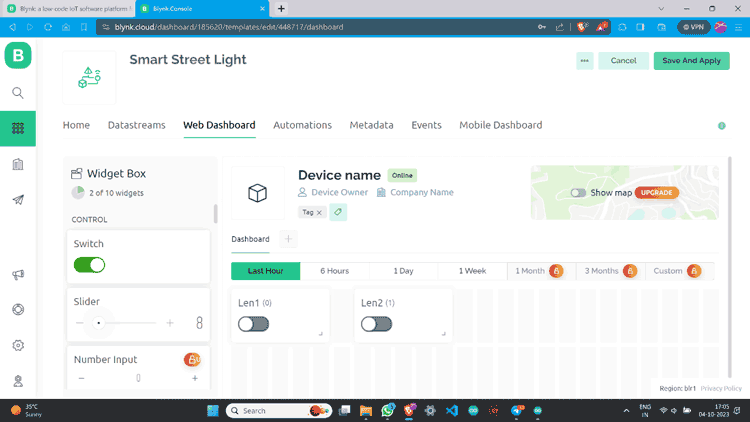
Step 8: I have made one more data stream with the same values.

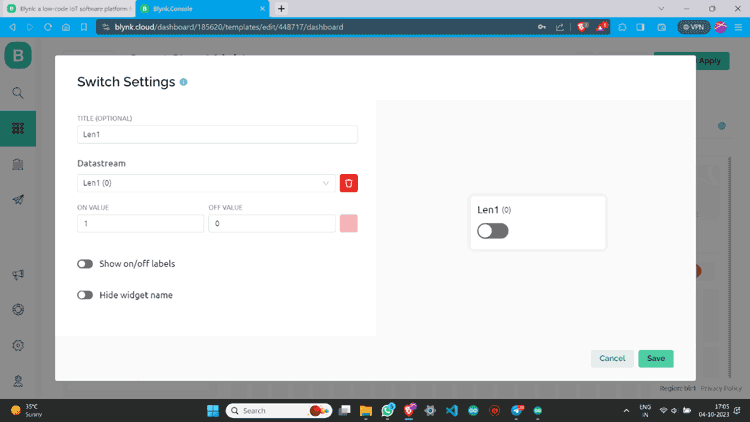
****

Step 9: Go to “Web Dashboard” and add Switch widgets.

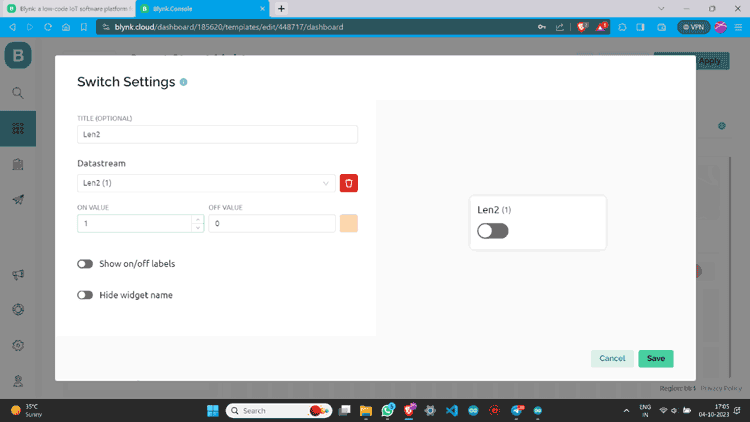
****

Step 10: Click on the Gear button, give values according to your choice, and save.

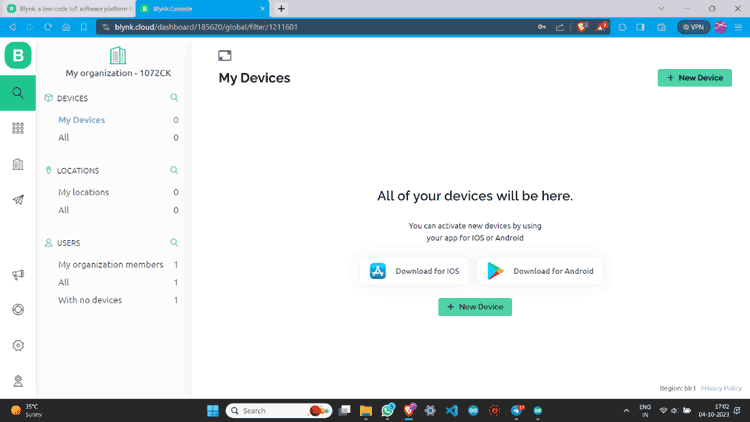
****

****

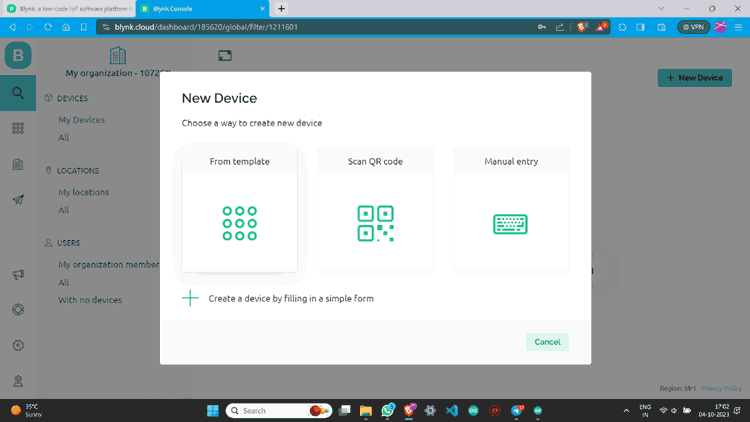
Step 11: I have made one more switch and assigned all values.

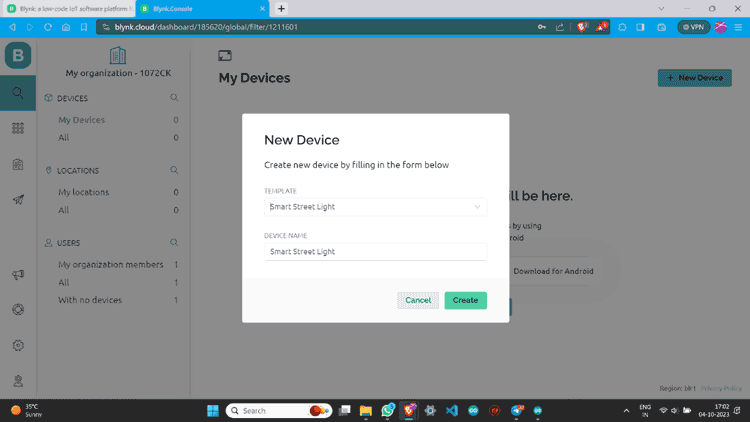
****

Step 12: Go to the lens icon and click on “+ New Device”.

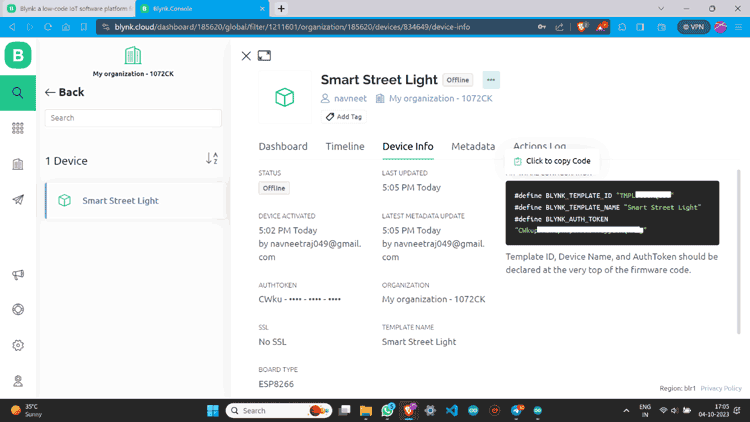
****

Step 13: Click “From template”, select the template, and create.

****

****

Step 14: Now you can see the device. Click on “Device Info” and copy the Auth Token.

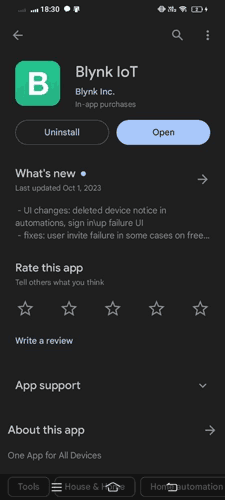
****

**Blynk 2.0 Application Setup**

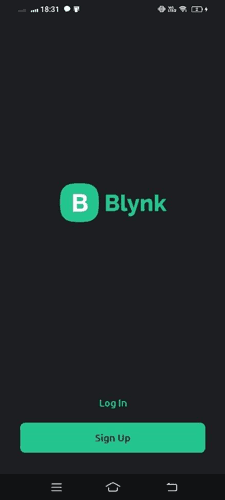
Now after Web setup, we have to make a Mobile App Interface so that we can able to control it from any Android or IOS

Step 1:  Download the Blynk app.

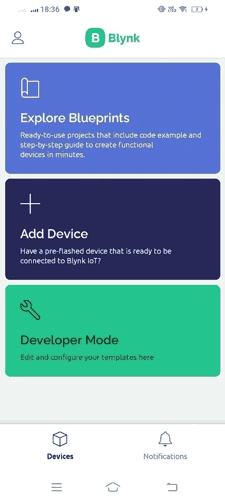
The Blynk app is available for both IOS and Android devices. You can download the app from the App Store or Google Play.

****

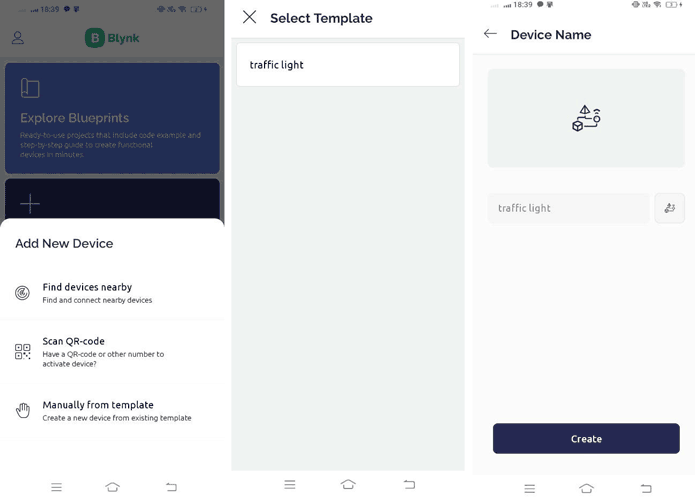
Step 2: Now open the Blynk IOT app and Log in with the credentials.

****

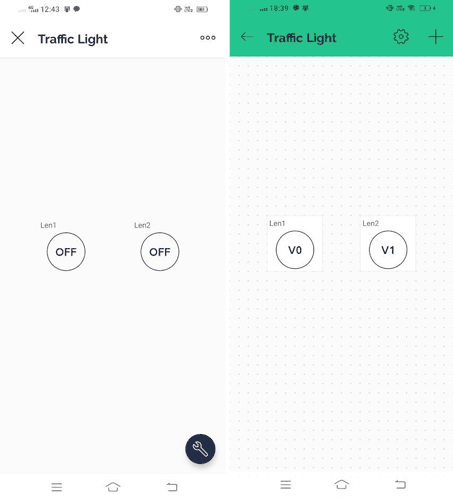
Step 3: After logging in click on “Add Device”.

****

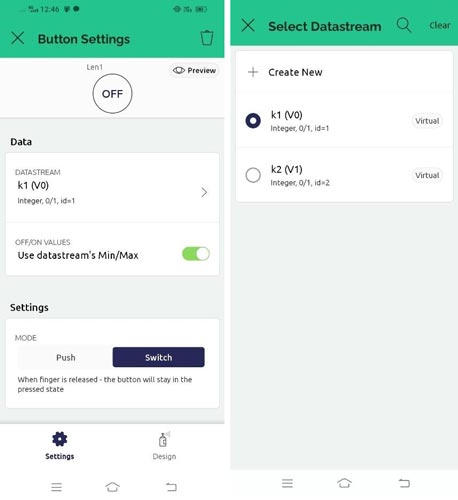
Step 4: Click “Manually from template”, select the device we already created, and then create.

****

Step 5: Now you are on the dashboard of the device then click on the tool button to enter in Developer Mode.

****

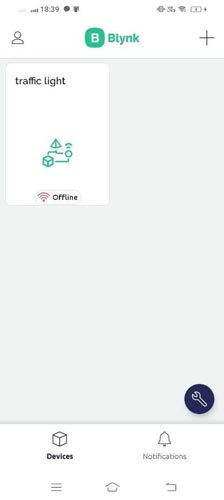
Step 6: Click on the button icons to assign values like Datastream and Mode. You can also change its design and looks.

****

Step 7: You Have successfully created a device dashboard.

****

Step 8: You can check the device status also.

****

**Setup Arduino IDE to NodeMCU ESP8266 board to use Blynk**

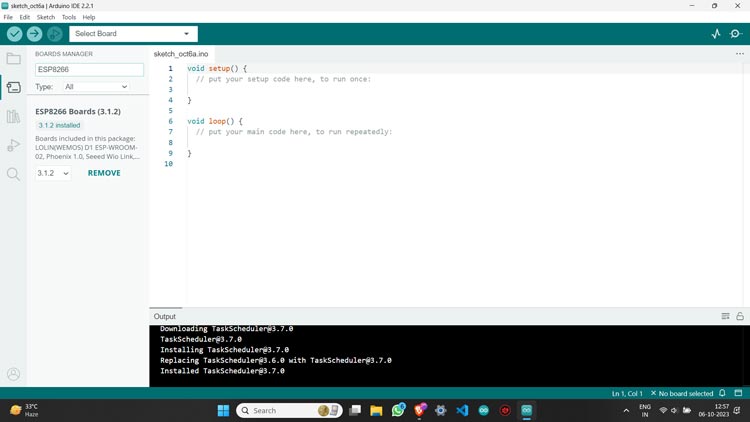
Now we have to set our Arduino IDE to program the NodeMCU ESP8266 board.

Step 1: In Arduino IDE go to the “BOARDS MANAGER” and install ESP8266.

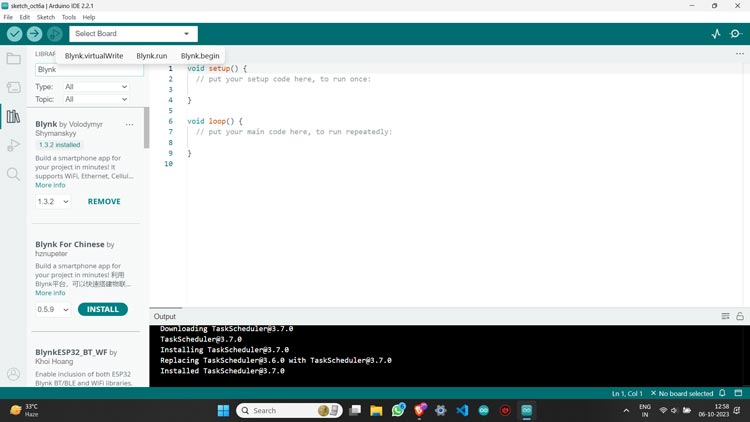
**Install Blynk Library in Arduino IDE**

to set our **Arduino IDE** to program the **NodeMCU ESP8266 board**.

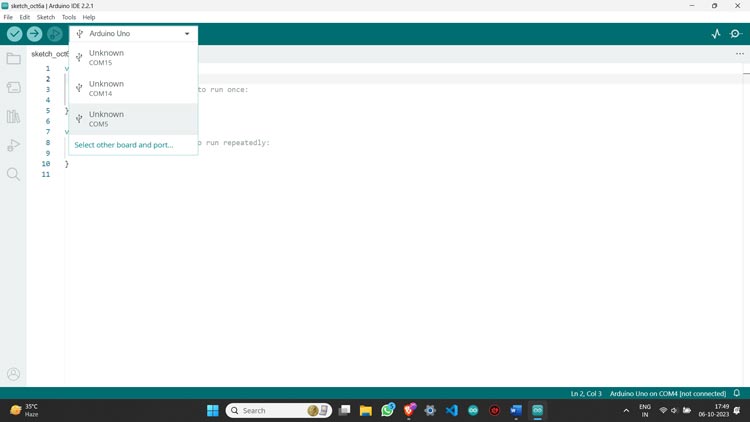
Step 1: In Arduino IDE go to the “**BOARDS MANAGER**” and install **ESP8266**.

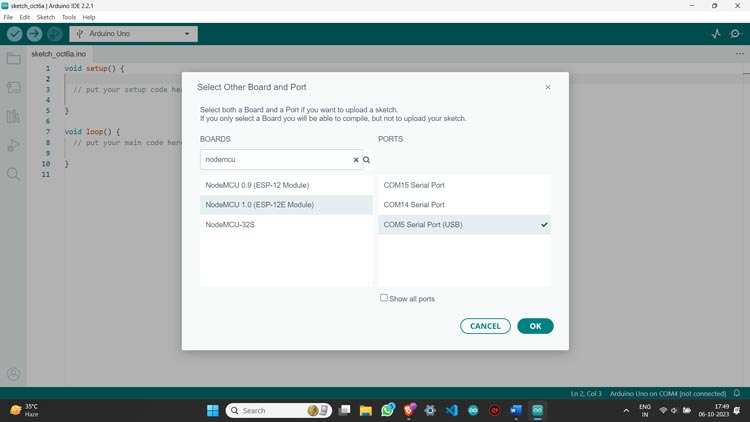


Step 2: Then go to the “**LIBRARY MANAGER**” and install the **Blynk library**.



Step 3: Plug your NodeMCU ESP8266 board into the PC using a **USB cable** and then **select your board and port**.





*Fig 5.5. Arduino IDE ready to program by using Blynk app*

Step 4: Your Arduino IDE is now **ready**to program your board for using Blynk app

**CHAPTER 6**

1. **WORKING PRINCIPAL OF IDEATION**

**6.1 Tell about Project Working Processes:**

The development and implementation of the IoT-based industrial automation system follow a systematic approach, comprising several key phases to ensure effective execution. The project work flow includes the following processes:

1. **Requirement Analysis:**
   * Conduct a thorough assessment of the current industrial processes to identify specific automation needs and areas for improvement.
   * Engage stakeholders to gather requirements, including operational goals, performance metrics, and constraints related to existing systems.
2. **System Design:**
   * Develop a comprehensive architecture for the IoT-based automation system, outlining the integration of sensors and communication protocols.
   * Create detailed specifications for hardware and software components, including the selection of IoT platforms, data storage solutions, and user interface designs.
3. **Component Selection:**
   * Choose appropriate IoT devices (sensors, controllers, ) based on the requirements identified in the analysis phase.
   * Evaluate and select cloud services for data storage and processing, ensuring compatibility with the system architecture.
4. **Development Phase:**
   * **Hardware Setup:** Install and configure sensors on the production floor, ensuring proper connectivity to the control system.
   * **Software Development:** Develop the backend infrastructure for data collection and processing, including APIs for communication between devices and the cloud.
   * **User Interface:** Create intuitive dashboards and applications for operators to monitor and control processes in real time.
5. **Integration and Testing:**
   * Integrate all components of the system, ensuring seamless communication between sensors, and the cloud platform.
   * Conduct rigorous testing to validate system functionality, performance, and reliability. This includes unit testing, integration testing, and system testing under various operational conditions.
6. **Deployment:**
   * Implement the system in a live production environment, ensuring minimal disruption to ongoing operations.
   * Provide training sessions for staff to familiarize them with the new system, emphasizing its features and benefits.
7. **Monitoring and Optimization:**
   * After deployment, continuously monitor system performance and gather feedback from users.
   * Utilize collected data for ongoing optimization, adjusting parameters and refining algorithms to enhance efficiency and productivity.
8. **Maintenance and Support:**
   * Establish a maintenance plan to address any technical issues that arise post-deployment.
   * Provide ongoing support and updates to the system to incorporate new features and improve security measures.

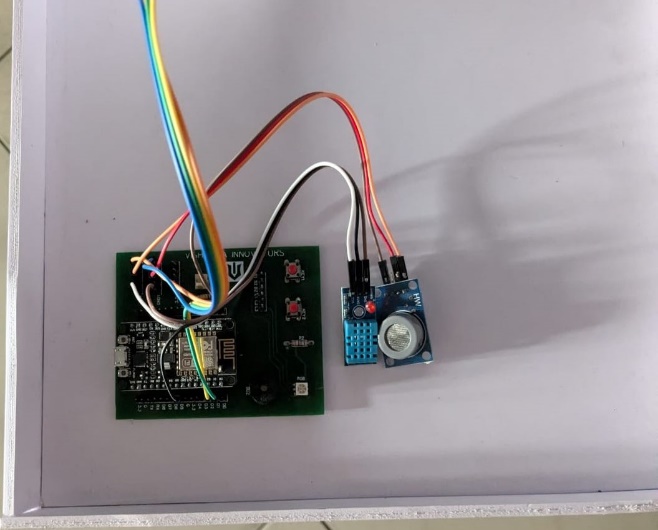
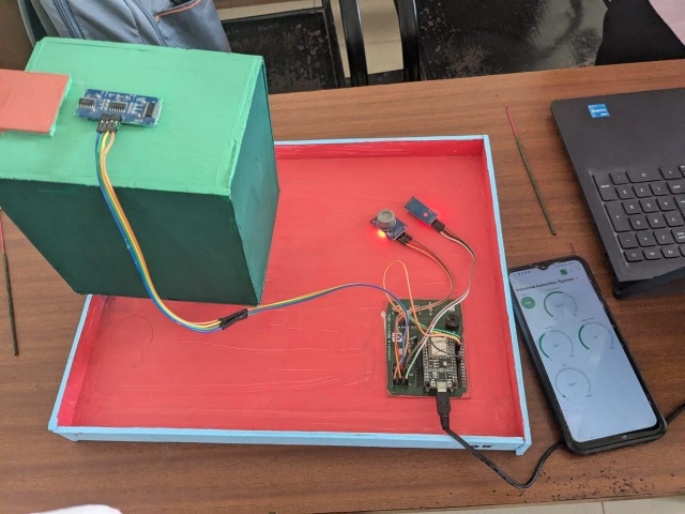
**Conclusion of Working Processes**

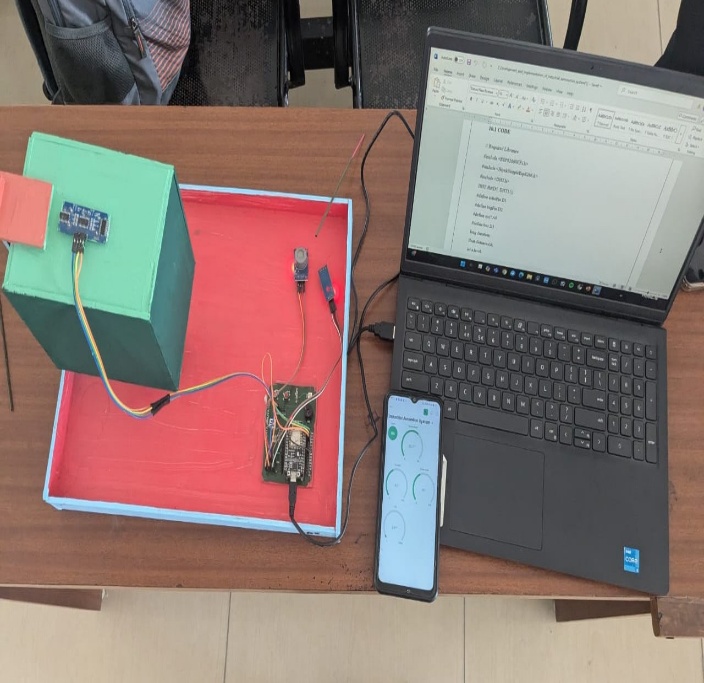
By following these structured working processes, the project aims to deliver a robust IoT-based industrial automation system that meets the identified needs and enhances operational efficiency. Each phase is critical to ensuring that the final system is effective, reliable, and scalable for future growth.

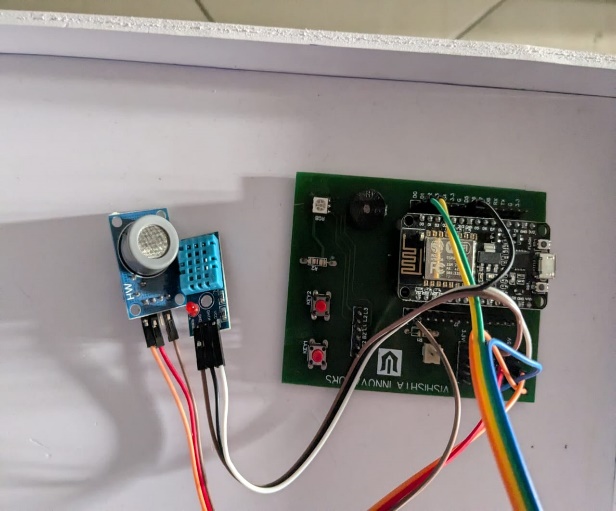
**CHAPTER 7**

1. **PROJECT PHOTOS & RESULTS**

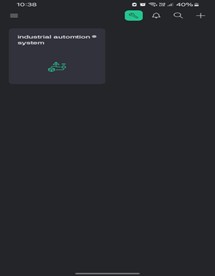
**7.1 Project Photos**

****

****

****

**7.2 Screenshots of Results**

**A screenshot of a device

AI-generated content may be incorrect.**

**A screenshot of a device

AI-generated content may be incorrect.**

**CHAPTER 8**

1. **CONCLUSION**

**8.1** **Project Conclusion:**

The development and implementation of an IoT-based industrial automation system represent a significant advancement in enhancing operational efficiency and responsiveness within manufacturing environments. This project successfully integrated cutting-edge IoT technologies to address key challenges faced by traditional automation systems, such as limited real-time data access, inefficient resource management, and the inability to adapt quickly to changing market demands.

We use Arduino Ide for writing and uploading code to the NodeMCU. Blynk App to create a user-friendly dashboard for displaying the data. By using of sensors on this project we just detecting the ranges of temperature, humidity, boiler range and harmful gases.

In conclusion, the development and implementation of an IoT-based industrial automation system represent a significant advancement in addressing the challenges faced by modern manufacturing environments. By harnessing real-time data monitoring, predictive analytics, and enhanced resource management, this system not only improves operational efficiency but also reduces costs associated with maintenance and downtime. The integration of IoT technologies allows for smarter decision-making and greater adaptability in an ever-evolving industrial landscape. As industries continue to embrace digital transformation, this innovative solution positions organizations to thrive in a competitive market, ensuring they are better equipped to respond to future challenges and opportunities.

**CHAPTER 9**

1. **FUTURE SCOPE**
   1. **Future addons or Updates in project:**

Future add-ons and updates for an industrial automation system utilizing IoT can significantly enhance its functionality and efficiency. Key areas of focus include advanced data analytics, such as predictive maintenance and real-time monitoring, which provide deeper operational insights. Integrating artificial intelligence can enable automated decision-making and anomaly detection, allowing for quicker responses to issues. Enhanced security measures, including advanced encryption and regular security audits, will help safeguard data transmission. Ensuring interoperability with open standards and developing APIs will facilitate seamless integration with third-party applications, while a modular architecture will support scalability.

Additionally, user experience enhancements like mobile applications and intuitive interfaces will make the system more accessible to operators. Incorporating energy management tools and exploring renewable energy integration can optimize resource usage and sustainability. Edge computing solutions will process data locally to reduce latency, and implementing digital twins can simulate operations for improved predictive analytics. Finally, automated reporting tools will ensure compliance with regulatory standards. Regular assessments and user feedback will be crucial in prioritizing these future developments effectively.

4o mini

**CHAPTER 10**

1. **FINAL CODE & RESOURCES**

The final code for the IoT-based Industrial automation system project that integrates the ultrasonic sensor with the ESP8266, detecting the ranges of temperature and humidity by DHT11 and detecting the gases by MQ7 and sending data to the Blynk platform. Additionally, resources such as libraries and references are provided for further development.

**10.1 CODE**

// Required Libraries

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

#include <DHT.h>

DHT dht(D5, DHT11);

#define echoPin D1

#define trigPin D2

#define mq7 A0

#define buz D3

long duration;

float distance,t,h;

int a,level;

// WiFi credentials.

// Set password to "" for open networks.

char ssid[] = "industrial";

char pass[] = "automation";

// This function will be called every time Slider Widget

// in Blynk app writes values to the Virtual Pin 1

BLYNK\_WRITE(V4)

{

a = param.asInt(); // assigning incoming value from pin V1 to a variable

digitalWrite(buz,a);

}

void setup()

{

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(buz, OUTPUT);

Serial.begin(9600);

dht.begin();

Blynk.begin(BLYNK\_AUTH\_TOKEN, ssid, pass);

}

void loop()

{

Blynk.run();

h = dht.readHumidity();

t = dht.readTemperature();

Serial.print("T=");

Serial.println(t);

Serial.print("H=");

Serial.println(h);

int co=map(analogRead(mq7),0,4095,0,1000);

Serial.print("CO=");

Serial.println(co);

Blynk.virtualWrite(V3, co);

Blynk.virtualWrite(V0, t);

Blynk.virtualWrite(V1, h);

delay(50);

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = duration \* 0.034 / 2;

Serial.println(distance);

level=map(distance,2,20,100,0);

Serial.println(level);

Blynk.virtualWrite(V2, level);

if(co>=450)

{

digitalWrite(buz, HIGH);

//Blynk.logEvent("notify","harmful gases released");

delay(4000);

digitalWrite(buz, LOW);

}

}

**10.2 Resources:**

1. **Libraries**:
   * **DHT.h**: Library for temperature and humidity.
   * **ESP8266WiFi.h**: Handles Wi-Fi communication for NodeMCU.
   * **BlynkSimpleEsp8266.h**: Blynk library to interface with the Blynk IoT platform.
2. **Hardware Components**:
   * **NodeMCU (ESP8266)**: Microcontroller for Wi-Fi communication.
   * **Ultrasonic Sensor (HC-SR04)**: Used for distance measurement.
   * **DHT Sensor**: Used for detecting the temperature and humidity.
   * **MQ**-**7 Sensor:** Used for gas detection.
   * **Power Supply**: Ensure the NodeMCU has a stable 5V power supply.
3. **Blynk Platform**:
   * Virtual pins **V0** in Blynk app for temperature.
   * Virtual pins **V1** in Blynk app for humidity.
   * Virtual pins **V2** in Blynk app for boiler level.
   * Virtual pins **V3** in Blynk app for CO.
   * Virtual pins **V4** in Blynk app for alarm.
4. **Tools**:
   * **Arduino IDE**: For writing and uploading code to the NodeMCU.
   * **Blynk App**: To create a user-friendly dashboard for displaying the boiler data.