CHAPTER 1

INTRODUCTION TO IOT

1.1 INTERNET OF THINGS

Fig1.1:Internet of Things

The major concept using in the Google assistant-controlled home automation is the Internet of Things. The Internet of Things (IoT) can be connecting various types of objects like smart phones, personal computer and tablets to the internet, which brings new-fangled type of communication between things and things, and things and people.

Home automation system any man-made objects that can be assigned an IP address and it has the ability to transfer data successfully over a network, the interaction through a network is called IoT. The internet helps us to bring immediate solutions for many problems and able to connect from any of the remote places. The Internets of Things technology is used to come in with innovative idea and large development space for smart homes to improve the living standards of life. The growth of the Internet of Things will reform a number of sectors, like healthcare, automation energy, transportation, etc. The cloud computing can be used in such case to implement the IoT infrastructure that augmented with sensors and actuators to monitor and control “things” from anywhere.

1.2 CHARACTERISTIC OF INTERNET OF THINGS

Connectivity: Connectivity is an important requirement of the IoT infrastructure. Things of IoT should be connected to the IoT infrastructure. Anyone, anywhere, anytime can connect, this should be guaranteed at all times. For example, connection between people through internet devices like mobile phones, and other gadgets, also connection between Internet devices such as routers, gateways, sensors, etc.

Intelligence and Identity: The extraction of knowledge from the generated data is very important. For example, a sensor generates data, but that data will only be useful if it is interpreted properly. Each IoT device has a unique identity. This identification is helpful in tracking the equipment and at times for querying its status.

Scalability: The number of elements connected to the IoT zone is increasing day by day. Hence, an IoT setup should be capable of handling the massive expansion. The data generated as an outcome is enormous, and it should be handled appropriately.

Dynamic and Self-Adapting(Complexity): IoT devices should dynamically adapt themselves to the changing context sand scenarios. Assume a camera meant for the surveillance. It should be adaptable to work in different conditions and different light situations (morning, afternoon, night).

Architecture: IoT architecture cannot be homogeneous in nature. It should be hybrid, supporting different manufacturers 'products to function in the IoT network. IoT is not owned by anyone engineering branch. IoT is a reality when multiple domains come together.

Safety: There is a danger of the sensitive personal details of the users getting compromised when all his/her devices are connected to the internet. This can cause a loss to the user. Hence, data security is the major challenge. Besides, the equipment involved is huge. IoT networks may also beat the risk. Therefore, equipment safety is also critical

1.3 KEY FEATURES OF IOT:

1. **Connectivity:** Devices communicate via the internet or local networks (Wi-Fi, Bluetooth, Zigbee, etc.).
2. **Automation:** Reduces the need for human intervention through real-time data processing.
3. **Data Collection & Analysis:** Gathers information from sensors to make informed decisions.
4. **Remote Control & Monitoring:** Users can manage devices from anywhere using mobile apps or web interfaces.
5. **Scalability:** IoT systems can be expanded by adding more devices.

How IoT Works:

- **Sensors & Devices:** Collect data (e.g., temperature, motion, humidity).
- **Connectivity:** Data is transmitted via the internet or networks.
- **Data Processing:** Cloud servers analyze the data.
- **User Interface:** The processed data is displayed through apps or dashboards for action.

1.4 APPLICATIONS OF IOT:

- **Smart Homes:** Automated lighting, security cameras, and voice assistants (e.g., Alexa, Google Home).
- **Healthcare:** Wearable devices monitor heart rate, oxygen levels, and physical activity.
- **Industrial IoT (IIoT):** Smart factories optimize manufacturing processes and predictive maintenance.
- **Smart Cities:** Traffic management, waste management, and smart grids.
- **Agriculture:** IoT-based irrigation and soil monitoring systems enhance crop yield.

Benefits of IoT:

- Improved efficiency and productivity
- Cost savings through automation
- Better decision-making with real-time insights
- Enhanced security and safety

Challenges of IoT:

- Security risks (cyber threats, data privacy concerns)
- Compatibility and interoperability between different IoT devices
- High initial costs for implementation

1.5 ADVANTAGES OF IOT:

1. **Automation & Control:**

- Reduces human intervention by enabling smart devices to operate automatically.
- Example: Smart thermostats adjust room temperature without manual input.

2. **Efficiency & Productivity:**

- IoT optimizes processes in industries, reducing errors and saving time.
- Example: Automated inventory tracking in warehouses.

3. **Cost Savings:**

- Reduces operational costs by improving energy efficiency and resource utilization.
- Example: Smart lighting systems lower electricity bills.

4. **Real-time Monitoring & Data Collection:**

- Provides instant updates on device performance, health, and environmental conditions.
- Example: Wearable health monitors track heart rate and blood pressure.

5. **Improved Decision-Making:**

- AI and analytics process IoT data to provide insights for better decision-making.
- Example: Predictive maintenance in factories prevents unexpected breakdowns.

6. **Enhanced Security & Safety:**

- IoT-based surveillance systems improve home and business security.
- Example: Smart doorbells with cameras detect visitors remotely.

7. **Convenience & Remote Access:**

- Users can control devices from anywhere using smartphones or computers.
- Example: Turning off home appliances via a mobile app.

1.6 DISADVANTAGES OF IOT:

1. **Security Risks & Cyber Threats:**

- IoT devices are vulnerable to hacking, data breaches, and malware attacks.
- Example: Hackers gaining access to smart home systems.

2. **Privacy Concerns:**

- IoT collects large amounts of personal data, leading to privacy issues.
- Example: Smart assistants (Alexa, Google Home) may record conversations.

3. **High Initial Cost & Maintenance:**

- Setting up IoT systems requires investment in devices, infrastructure, and security.
- Example: Industrial IoT solutions require expensive sensors and cloud storage.

4. **Interoperability Issues:**

- Different manufacturers use different protocols, making device integration difficult.
- Example: A smart lock may not work with a particular home automation system.

5. **Dependence on Internet & Power Supply:**

- IoT devices rely on internet connectivity; disruptions affect functionality.
- Example: A smart home system may fail during a power outage or weak Wi-Fi signal.

6. **Data Overload & Management Issues:**

- IoT generates massive data, requiring efficient storage and processing solutions.
- Example: A smart city traffic system needs powerful servers to analyze real-time data.

7. **Ethical & Legal Concerns:**

- Governments must regulate IoT data usage and prevent misuse.
- Example: Unauthorized surveillance through IoT cameras.

CHAPTER-2

LITERATURE SURVEY

2.1 EXISTINGSYSTEM

The traditional underground water management system operates through manual intervention, where an operator controls a pump to transfer water from an underground source to a storage tank. When water is needed, the operator switches on the pump, monitors the tank's filling level visually, and turns off the pump when sufficient water is collected.The stored water is then distributed to users through basic distribution systems. This manual process lacks automation, sensors, or monitoring capabilities, making it prone to inefficiencies like overflow, pump damage, and water wastage, while requiring constant human supervision for operation and maintenance.

2.2 LIMITATIONS

- **Manual Dependency**–Requires constant human supervision for operation and monitoring.
- **Water Wastage**–No automation leads to overflow and inefficient usage.
- **Pump Damage**–Lack of monitoring increases the risk of dry running and wear.
- **Energy Inefficiency**–Pumps may run longer than necessary, wasting electricity
- **No Data Tracking**–Absence of sensors prevents tracking consumption and detecting leaks.

CHAPTER-3

INTRODUCTION TO PROJECT

Water scarcity is one of the most pressing global challenges, affecting millions of people and ecosystems worldwide. With increasing population growth, urbanization, and climate change, the demand for freshwater is rapidly rising. However, inefficient water distribution systems, unchecked consumption, and lack of proper monitoring have led to excessive water wastage. Many traditional water supply systems do not have automated control mechanisms to regulate consumption, making it difficult to ensure sustainable usage. If these issues are not addressed, future generations may face severe water shortages, impacting agriculture, industries, and daily human needs.

To tackle this problem, we propose a **Smart Underground Water Management System** that leverages modern technology to monitor, regulate, and optimize water usage effectively. This system is designed to provide an efficient solution for managing underground water resources by integrating **real-time monitoring, automatic control, and remote access** features. The primary goal is to prevent unnecessary water wastage while ensuring a reliable and sustainable supply.

3.1 EXISTING SYSTEM

The traditional underground water management system operates through manual intervention, where an operator controls a pump to transfer water from an underground source to a storage tank. When water is needed, the operator switches on the pump, monitors the tank's filling level visually, and turns off the pump when sufficient water is collected. The stored water is then distributed to users through basic distribution systems. This manual process lacks automation, sensors, or monitoring capabilities, making it prone to inefficiencies like overflow, pump damage, and water wastage, while requiring constant human supervision for operation and maintenance.

3.2 PROPOSED SYSTEM

This system is designed using an **ESP32 micro controller**, a **water flow sensor**, a **relay module**, and a **motor** to automate water distribution based on predefined consumption limits. The system continuously monitors water flow rates and displays real-time data on the **Blynk app**, where users can track their consumption and control the motor remotely. When the water consumption exceeds the set threshold, the motor automatically turns off to prevent excessive usage, and an LED indicator.

3.3 BLOCK DIAGRAM

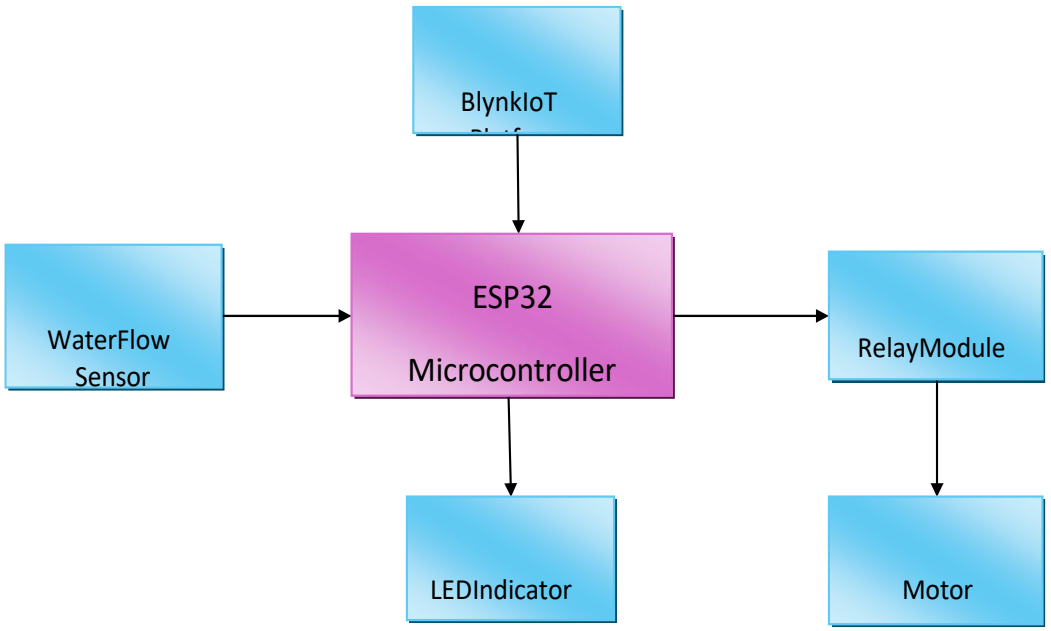


Fig3.1:Block Diagram

3.4 POWERSUPPLY

The input to the circuit is applied from the regulated power supply. The a.c. inputi.e.,230V from the mains supply is step down by the transformer to12Vand is fed to a rectifier. The output obtained from the rectifier is a pulsating d.c voltage. So in order to get a pure d.c voltage, the output voltage from the rectifier is fed to a filter to remove any a.c components present even after rectification. Now, this voltage is given to a voltage regulator to obtain a pure constant dc voltage.

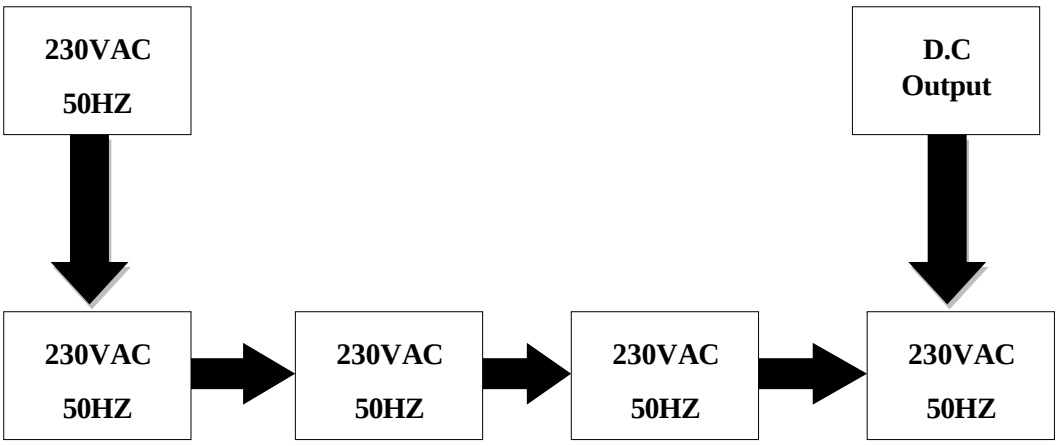


Fig3.2: Block diagram of power supply

3.4.1 TRANSFORMER

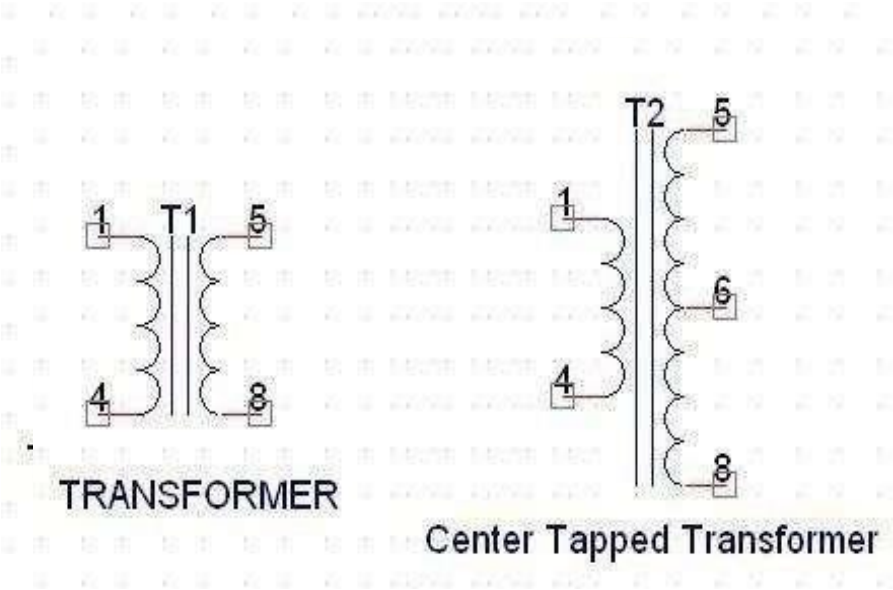


Fig3.3: Transformer

A transformer consists of two coils also called as "WINDINGS" namely PRIMARY & SECONDARY.

They are linked together through inductively coupled electrical conductors also called as CORE. A changing current in the primary causes a change in the Magnetic Field in the core& this in turn induces an alternating voltage in the secondary coil. If load is applied to the secondary then an alternating current will flow through the load. If we consider an ideal condition then all the energy from the primary circuit will be transferred to the secondary circuit through the magnetic field.

$$P_{primary} = P_{secondary}$$

so,

$$I_p V_p = I_s V_s$$

$$\frac{V_s}{V_p} = \frac{N_s}{N_p}$$

3.4.2 RECTIFIER

A rectifier is a device that converts an AC signal into DC signal. For rectification purpose we use a diode, a diode is a device that allows current to pass only in one direction i.e. when the anode of the diode is positive with respect to the cathode also called as forward biased condition& blocks current in the reversed biased condition.

Rectifier can be classified as follows:

1) Half Wave rectifier

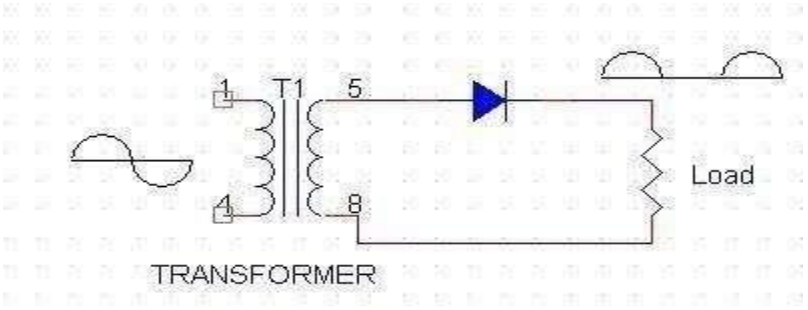


Fig3.4: Half Wave Rectifier

This is the simplest type of rectifier as you can see in the diagram a half wave rectifier consists of only one diode. When an AC signal is applied to it during the positive half cycle the diode is forward biased & current flows through it. But during the negative half cycle diode is reverse biased &no current flows through it. Since only one half of the input reaches the output, it is very inefficient to be used.

1) Full wave rectifier

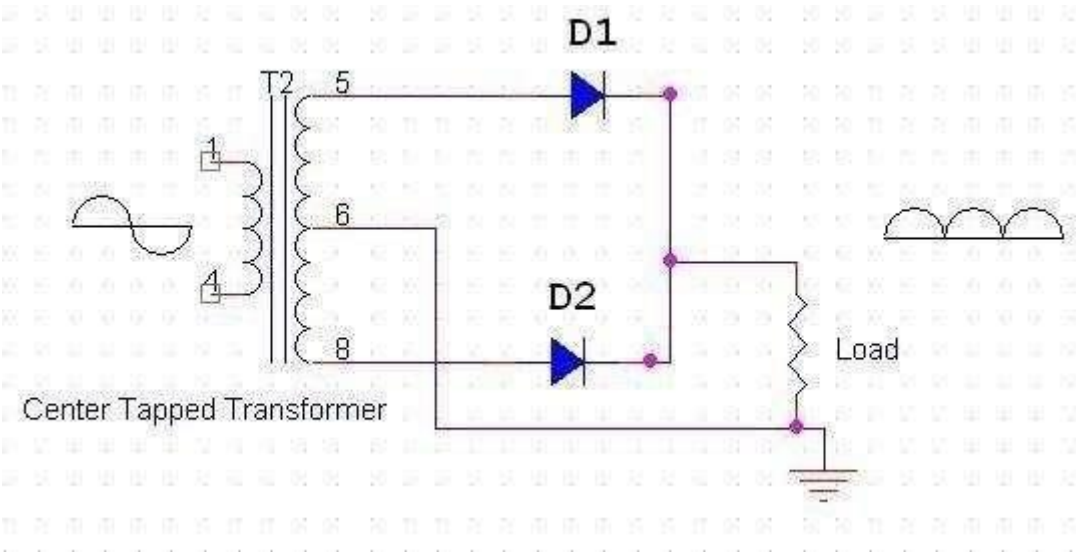


Fig3.5:Full Wave Rectifier

Half wave rectifier is quite simple but it is very inefficient, for greater efficiency we would like to use both the half cycles of the AC signal. This can be achieved by using a center tapped transformer i.e. we would have to double the size of secondary winding &provide connection to the center. So during the positive half cycle diode D1conducts& D2 is in reverse biased condition. During the negative half cycle diode D2 conducts & D1 is reverse biased. Thus we get both the half cycles across the load.

One of the disadvantages of Full Wave Rectifier design is the necessity of using a center tapped transformer ,thus increasing the size & cost of the circuit. This can be avoided by using the Full Wave Bridge Rectifier.

3.4.3 FILTER CAPACITOR

Even though half wave & full wave rectifier give DC output, none of them provides a constant output voltage. For this were quire to smoothen the waveform received from the rectifier .This can be done by using a capacitor at the output of the rectifier this capacitor is also called as “FILTER CAPACITOR” or “SMOOTHINGCAPACITOR” or “RESERVOIR

CAPACITOR". Even after using this capacitor a small amount of ripple will remain.

We place the Filter Capacitor at the output of the rectifier the capacitor will charge to the peak voltage during each half cycle then will discharge its stored energy slowly through the load while the rectified voltage drops to zero, thus trying to keep the voltage as constant as possible.

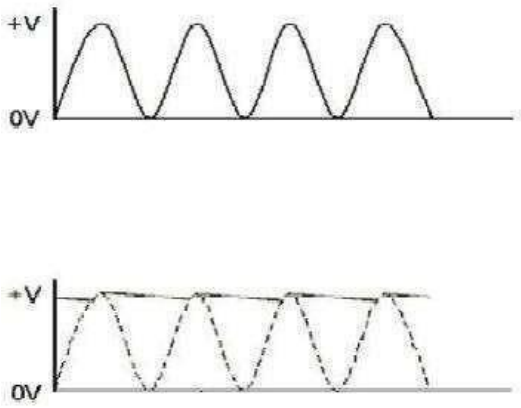


Fig3.6:Wave forms of Filter Capacitor

If we go on increasing the value of the filter capacitor then the Ripple will decrease. But then the costing will increase. The value of the Filter capacitor depends on the current consumed by the circuit, the frequency of the waveform &the accepted ripple.

$$C = \frac{V_r F}{I}$$

Where,

Vr=accepted ripple voltage (shouldnotbemorethan10%ofthevoltage)

I=current consumed by the circuit in Amperes.

F=frequency of the waveform. A halfwave rectifier has only one peak in one cycle so F=25hz Where as a full wave rectifier has Two peaks in one cycles oF=100Hz.

3.4.4 VOLTAGEREGULATOR

A Voltage regulator is a device which converts varying input voltage into a constant regulated output voltage. Voltage regulator can be of two types

1) Linear Voltage Regulator

Also called as Resistive Voltage regulator because they dissipate the excessive voltage resistively as heat.

2) Switching Regulators.

They regulate the output voltage by switching the Current ON/OFF very rapidly. Since their output is either ON or OFF it dissipates very low power thus achieving higher efficiency as compared to linear voltage regulators. But they are more complex & generate high noise due to their switching action. For low level of output power switching regulators tend to be costly but for higher output wattage they are much cheaper than linear regulators.

The most commonly available Linear Positive Voltage Regulators are the 78XX series where the XX indicates the output voltage. And 79XX series is for Negative Voltage Regulators.

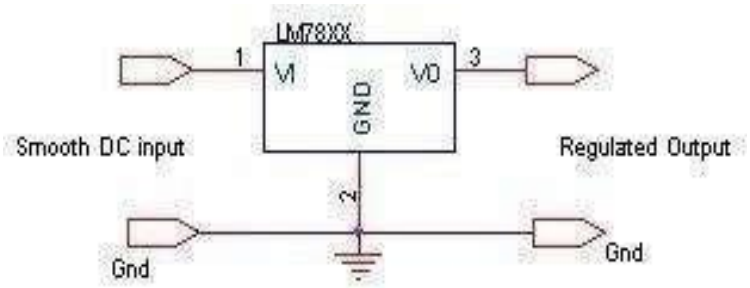


Fig3.7:pin diagram of voltage regulator

After filtering the rectifier output the signal is given to a voltage regulator. The maximum input voltage that can be applied at the input is 35V. Normally there is a 2-3 Volts drop across the regulator so the input voltage should be at least 2-3 Volts higher than the output voltage.

If the input voltage gets below the Vmin of the regulator due to the ripple voltage or due to any other reason the voltage regulator will not be able to produce the correct regulated voltage.

3.5 CIRCUIT DIAGRAM

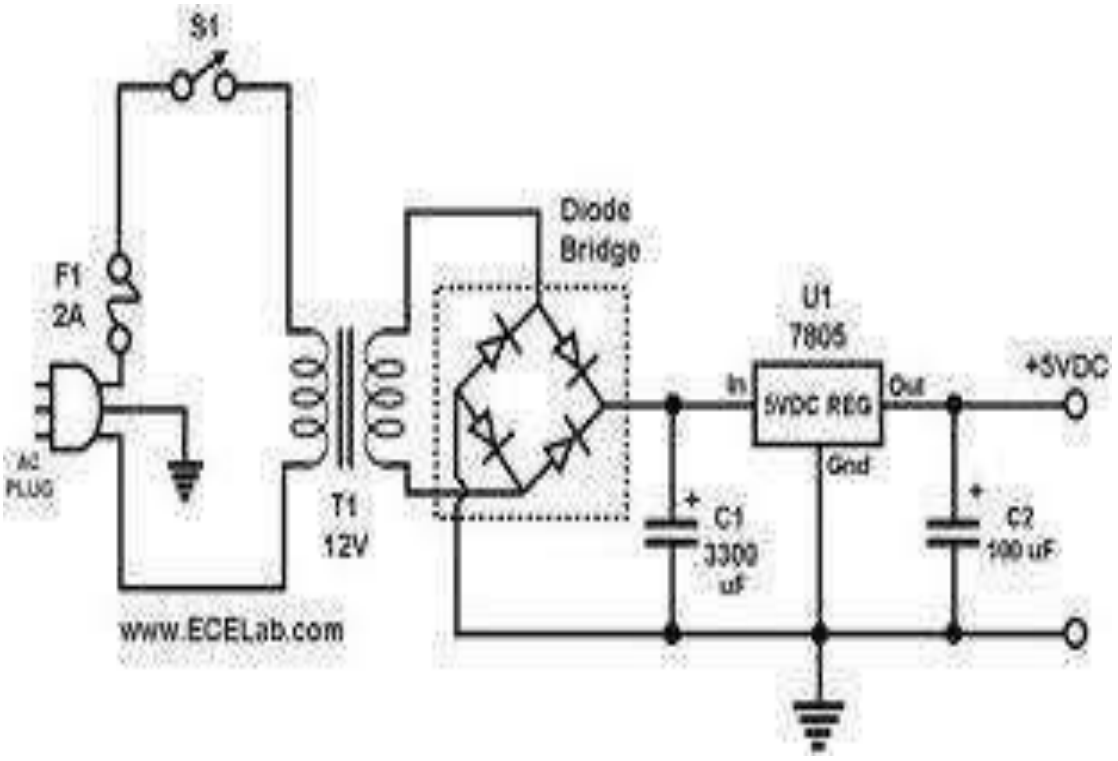


Fig3.8:**Circuit diagram**

3.5.1 IC7805

7805 is an integrated three-terminal positive fixed linear voltage regulator. It supports an input voltage of 10 volts to 35 volts and output voltage of 5 volts. It has a current ratingof1 amp although lower current models are available. Its output voltage is fixed at5.0V. The 7805 also has a built-in current limiter as a safety feature. 7805 is manufactured by many companies, including National Semiconductors and Fairchild Semiconductors.

The 7805 will automatically reduce output current if it gets too hot. The last two digits represent the voltage; for instance, the 7812 is a 12-volt regulator. The78xx series of regulators is designed to work in complement with the79xx series of negative voltage regulator sin systems that provide both positive and negative regulated voltages, since the 78xx series can't regulate negative voltages in such a system.

The7805&78 is one of the most common and well-known of the78xx series regulators, as it's small component count and medium-power regulated5V make it useful for powering TTL devices.

Table3.1:Specifications of IC7805

SPECIFICATIONS	**IC7805**
V_{out}	5V
V_{ein}-V_{out} Difference	5V-20V
Operation Ambient Temp	0-125°C
Output I_{max}	1A

3.6 ESP32 MICRO CONTROLLER

Fig3.10:ESP32 Microcontroller

The ESP32 is a highly integrated microcontroller known for its low cost, low power consumption, and built-in Wi-Fi and Bluetooth capabilities. It features a dual-core Ten silica X tens a LX6 processor running at up to 240 MHz, 520 KB SRAM, and support for external flash memory. With multiple GPIOs, ADC, DAC, PWM, and communication interfaces like UART, SPI, I2C, and I2S,it allows seamless interfacing with various sensors , relays, and other peripherals. Operating at
3.3V, the ESP32 also supports multiple low-power modes, making it suitable for battery-operated applications.

Widely used in IoT, smart home automation, industrial control, and wireless sensor networks, the ESP32 excel sin projects requiring real-time data processing and remote connectivity. Its ability to connect to cloud platforms and handle real-time monitoring makes it ideal for applications like underground water management systems, where precise control of water flow, motor operation, and consumption tracking are essential for efficient resource management.

3.6.1 ESP32 Microcontroller–Specification

ESP32Microcontroller	Specification
CPU	Dual-coreXtensaLX6microprocessor(32-bit), operating at 160 or 240 MHz
Memory	520KBSRAM
Wireless	Wi-Fi:802.11 b/g/n(2.4GHz),upto150Mbps Bluetooth: v4.2 BR/EDR and BLE
GPIO	Upto36programmableGPIOpins
ADC	12-bitSARADC,upto18channels
DAC	2-channel8-bit DAC
Interfaces	SPI,I2C, I2S,UART,CAN,EthernetMAC,IR, PWM, capacitive touch sensors
Security	Hardwareencryption:AES,SHA-2,RSA,ECC, RNG
Operating temperature	-40°Cto+85°C
Power consumption	Active:~240mA(typical) Deep sleep: ~10 µA ULP sensor-monitored pattern recognition:~20µA
Input voltage	2.3Vto3.6V
Package dimensions	Varies by module
Flash memory	External,typically4MBto16MBdependingon module

Table3.2:ESP32 Micro controller

3.6.2 ESP32Microcontroller Pin Discription

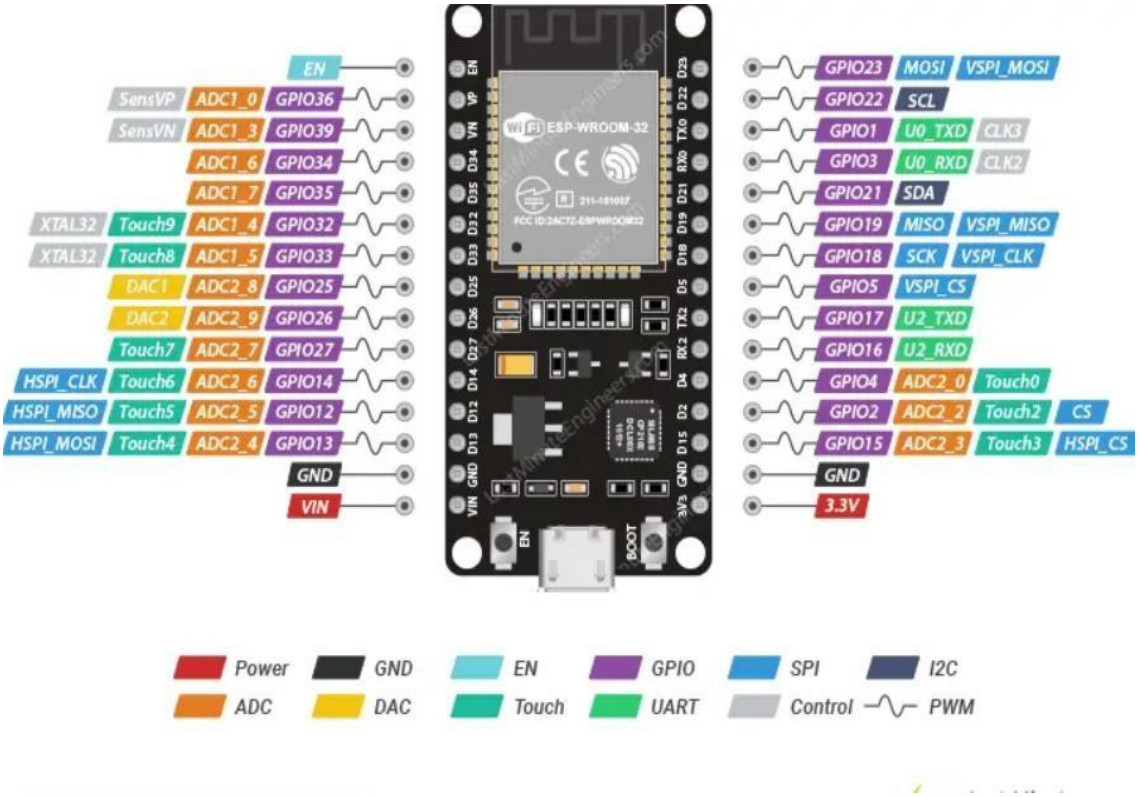


Fig3.11:ESP32MicrocontrollerPinDiagram

We can infer from the image thatESP32 Microcontroller got 38 pins in total.

We will see all the pins section wise as well as a detailed format at last.

- **Digital Pins:** All GPIOs except those used for flash memory can be used as digital I/O.
- **Analog Pins:**ESP32 has two ADCs (ADC1&ADC2)with 18 total analog inputs.
- **PWM Pins:** Any GPIO can be used for PWM as ESP32 has **16 PWM channels**.
- **DAC Pins:** GPIO 25& GPIO 26 can output analog voltage.
- **Power Pins :** VIN (for external power), 3.3V (regulated output) ,multiple GND.
- **Reset Pin :** EN pin is used to reset the board.

Pins 1 to 30

ESP32pins	**Pin name**	**Type**	**Function**
VIN	-	Power Input	Power supply(5V)
GND	-	Ground	Common ground
3.3V	-	Power Output	3.3VPowerOutput
EN	-	Enable	Chip Enable (Active HIGH)
GPIO36	ADC1_0	Input	Analog Input, Sens VP
GPIO39	ADC1_3	Input	Analog Input, Sens VN
GPIO34	ADC1_6	Input	Analog Input
GPIO35	ADC1_7	Input	Analog Input
GPIO32	ADC1_4	Input / Output	Analog Input, Touch Sensor(T9)
GPIO33	ADC1_5	Input / Output	Analog Input, Touch Sensor(T8)
GPIO25	DAC1	Input / Output	Digital I/O,DAC Output,ADC2_8
GPIO26	DAC2	Input / Output	Digital I/O,DAC Output,ADC2_9
GPIO27	ADC2_7	Input / Output	Digital I/O, Analog Input
GPIO14	HSPI_CLK	Input / Output	Digital I/O,Touch6, SPI Clock

GPIO12	HSPI_MISO	Input / Output	Digital I/O,Touch5, SPIMISO
ESP32pins	**Pin name**	**Type**	**Function**
GPIO13	HSPI_MOSI	Input / Output	Digital I/O,Touch4, SPIMOSI
GPIO23	MOSI	Input / Output	Digital I/O,VSPI MOSI
GPIO22	SCL	Input / Output	Digital I/O,I2C Clock
GPIO1	U0_TXD	Output	UARTTX(Serial Output)
GPIO3	U0_RXD	Input	UARTRX(Serial Input)
GPIO21	SDA	Input / Output	Digital I/O,I2C Data
GPIO19	MISO	Input / Output	Digital I/O,VSPI MISO
GPIO18	SCK	Input / Output	Digital I/O,VSPI Clock
GPIO5	CS	Input / Output	Digital I/O,VSPI Chip Select
GPIO17	U2_TXD	Output	UART2TX(Serial Output)
GPIO16	U2_RXD	Input	UART2RX(Serial Input)
GPIO4	ADC2_0	Input / Output	Digital I/O, Analog Input,Touch0
GPIO2	ADC2_2	Input / Output	Digital I/O,Analog Input,Touch2
GPIO15	ADC2_3	Input / Output	Digital I/O, Analog Input, HSPI_CS

Table3.3: ESP32 Micro controller Pin Description

ESP32 Detailed Pin Configuration

GPIO Pin Architecture

- The ESP32 has 48 pins in total, with up to 34 of them usable as GPIO pins
- The pins are divided into two groups:
  - **GPIO0-GPIO31**: These are part of the CPU's digital I/O pads
  - **GPIO32-GPIO39**: These are connected to the analog-to-digital converter (ADC)
- The pins support configurable interrupt modes:
  - Rising edge trigger
  - Falling edge trigger
  - Both edges trigger
  - Low-level trigger
  - High-level trigger

GPIO Pin Restrictions and Limitations

- **Input-Only Pins**: GPIO34, GPIO35, GPIO36, GPIO37, GPIO38, and GPIO39 can only be configured as input pins
- **Flash-Connected Pins**: GPIO6-GPIO11 are connected to the integrated SPI flash and are not recommended for general use
- **Strapping Pins**: The following pins have special bootstrapping functions:
  - GPIO0: Boot mode selection (0=Download mode, 1=Normal boot)
  - GPIO2: Must be high during boot
  - GPIO5: Must be high during boot
  - GPIO12: If held high during startup, the flash voltage is set to 1.8V (default is 3.3V)
  - GPIO15: Must be low during boot

Analog Capabilities

- **ADC (Analog-to-Digital Converter)**:

- **ADC1 Channel Assignment**:

- GPIO36 (ADC1_CH0)
- GPIO37 (ADC1_CH1)
- GPIO38 (ADC1_CH2)
- GPIO39 (ADC1_CH3)
- GPIO32 (ADC1_CH4)
- GPIO33 (ADC1_CH5)
- GPIO34 (ADC1_CH6)
- GPIO35 (ADC1_CH7)

ADC2 Channel Assignment:

- GPIO4 (ADC2_CH0)
- GPIO0 (ADC2_CH1)
- GPIO2 (ADC2_CH2)
- GPIO15 (ADC2_CH3)
- GPIO13 (ADC2_CH4)
- GPIO12 (ADC2_CH5)
- GPIO14 (ADC2_CH6)
- GPIO27 (ADC2_CH7)
- GPIO25 (ADC2_CH8)
- GPIO26 (ADC2_CH9)

- 12-bit resolution (0-4095 values)
- ADC2 pins cannot be used when WiFi is active

DAC (Digital-to-Analog Converter):

- GPIO25 (DAC_CH1)
- GPIO26 (DAC_CH2)

- 8-bit resolution (0-255 values)

Touch Sensors: 10 capacitive touch GPIOs

- GPIO0 (TOUCH_CH1)
- GPIO2 (TOUCH_CH2)
- GPIO4 (TOUCH_CH3)
- GPIO12 (TOUCH_CH5)
- GPIO13 (TOUCH_CH4)

- GPIO14 (TOUCH_CH6)
- GPIO15 (TOUCH_CH7)
- GPIO27 (TOUCH_CH7)
- GPIO32 (TOUCH_CH9)
- GPIO33 (TOUCH_CH8)

Communication Interfaces

UART (Universal Asynchronous Receiver-Transmitter):

- **UART0**:
  - TX: GPIO1
  - RX: GPIO3
  - Primary debug UART

- **UART1**:
  - TX: GPIO10
  - RX: GPIO9
  - Note: Usually connected to flash, not accessible on most dev boards

- **UART2**:
  - TX: GPIO17
  - RX: GPIO16

- Can be remapped to other pins

- **I2C (Inter-Integrated Circuit)**:

  - **I2C0** and **I2C1** interfaces
  - Can be assigned to any GPIO pins through software
  - Typical pins used (on most dev boards):

  - SDA: GPIO21
  - SCL: GPIO22

SPI (Serial Peripheral Interface):

- **VSPI** (Default SPI bus):

  - MOSI: GPIO23
  - MISO: GPIO19
  - CLK: GPIO18
  - CS: GPIO5

- **HSPI**:

  - MOSI: GPIO13
  - MISO: GPIO12
  - CLK: GPIO14
  - CS: GPIO15

  - Can be remapped to other pins through software

- **I2S (Inter-IC Sound)**:

  - Two I2S interfaces
  - Can be assigned to any GPIO pins through software

3.7 WATER FLOW SENSOR:

Water flow sensor can be used to measure the flow of liquids, i.e. the consumption of liquids in industrial or domestic usage. For example you can make a robotic cocktail dispensing machine, and can use this sensors to accurately measure components like Soda, Water, etc. Water flow sensor consists of a plastic valve body, a water rotor, and a hall- effect sensor. When water flows through the rotor, rotor rolls. Its speed changes with different rate of flow. The hall-effect sensor outputs the corresponding pulse Signal.

3.7.1 Features:

- The simple and compact module.
- Easy to Install.
- High Sealing Performance.
- High-Quality Hall Effect Sensor.

3.7.2 Applications:

- Industrial process monitoring.
- Leak detection.
- Home water consumption tracking.
- Agricultural water management.

3.7.3 Advantages:

- Real-time flow measurement.
- Precise water usage tracking.
- Leak detection capabilities.
- Remote monitoring potential.

3.7.4 Challenges:

- Sensitivity to installation conditions.
- Some types affected by fluid properties.
- Maintenance needs.
- Cost variations between sensor types.

3.8 Relay

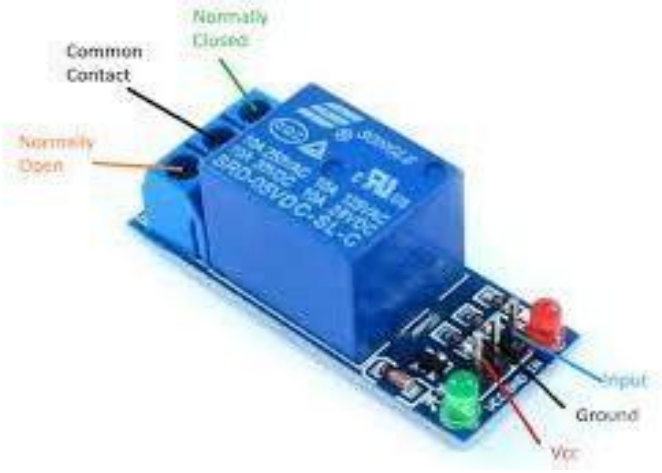


Fig3.13:Relay module

A relay is an electrical switch that opens and closes circuits electromechanically or electronically. It allows a low-power electrical signal to control a much higher-power circuit, essentially acting as an electrical control device that can switch or route electrical signals.

Relay is an electrically operated switch used to control high-power devices with a low-power signal. It consists of an electromagnet (coil), a movable armature, and a set of contacts. When a small voltage is applied to the coil, it generates a magnetic field that pulls the armature, causing the contacts to change position. This allows a low-power circuit (such as a microcontroller) to control high-voltage or high-current devices like motors, lights, or heaters. Relays can be **normally open (NO)** or **normally closed (NC)**, meaning they either complete or break a circuit when activated. In your **underground water management system**, the relay controls the motor based on signals from the ESP32, ensuring automatic water regulation based on consumption limits.

3.8.1 Working Principle

A relay module consists of:

1. **Electromagnetic Coil**–When energized, it creates a magnetic field.
2. **Switching Mechanism(Contacts)**–Opens or closes the circuit when the coil is activated.
3. **Diode(Fly back Diode)**–Protects the circuit from voltages pikes.
4. **Transistor&Optocoupler(Optional)**–Used in some modules for better control and isolation.

When a low-voltage signal is sent to the relay module, it energizes the coil, causing the switch contacts to close(or open),allowing current to flow to the connected high-power device(e.g.,motors, lights, pumps).

3.8.2 FEATURES:

- Protection Features.
- Durability.
- Fast Switching Speed.
- High Voltage & Current Handling.
- Low Power Control.
- Normally Open (NO) & Normally Closed (NC) Contacts.

3.8.3 Applications:

- Industrial automation.
- Motor control.
- Home appliance control.
- Computer and electronic devices witching.
- Protection circuits in electrical systems.

3.8.4 Advantages:

- Can control high-power circuits with low-power signals.
- Provide electrical isolation.
- Flexible circuit protection.
- Controllable through various input signals.

3.8.5 Disadvantages:

- Mechanical relay shave limited switching life cycle.
- Can experience contact bounce.
- Slower switching compared to solid-state alternatives.

3.9 LED

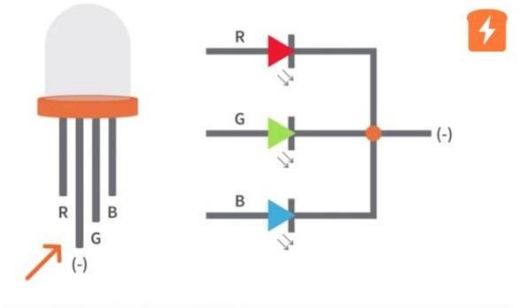


Fig3.14: LED Diagram

LEDs are semiconductor devices that emit light when an electric current passes through them. They have revolutionized lighting technology with their efficiency, durability, and versatility. An **LED (Light Emitting Diode)** is a semiconductor device that emits light when an electric current flows through it. It works on the principle of **electroluminescence**, where electrons recombine with holes in the semiconductor material, releasing energy in the form of light. LEDs are widely used for lighting, display panels, indicators, and electronic circuits due to their **energy efficiency, durability, and long lifespan**.

Unlike traditional incandescent bulbs, LEDs do not rely on a filament, which makes them more **efficient and heat-resistant**. They are available in various colors such as **red, green, blue, and white**, depending on the semiconductor material used. LEDs operate at **low voltage (typically 2V–3.5V) and consume very little power**, making them ideal for battery-powered and low-power applications.

Additionally, LEDs are highly **durable, shock-resistant, and environmentally friendly**, as they do not contain hazardous materials like mercury. Their fast response time allows them to be used in applications like **indicators, digital displays, remote controls, automotive lighting, and smart lighting systems**.

3.9.1 Basic Working Principle:

- Semiconductor diode that emits light when electrically charged.
- Uses electroluminescence phenomenon.
- Converts electrical energy directly into light.
- Consists of a chip of semiconducting material doped with impurities.

3.9.2 FEATURES

- Energy Efficient.
- Long Lifespan.
- Small Size.
- Fast Switching.
- Fast Switching.

3.9.3 Applications:

- Headlights(providing brighter, more energy-efficient illumination)
- Tail light sand brake lights.
- Dashboard indicators.
- Interior ambient lighting.

3.9.4 Advantages:

- Lower power consumption.
- Longer life span.
- Faster response time.
- Enhanced visibility and safety.

3.9.5 Disadvantages:

- Heat Sensitivity.
- Color Inconsistency.

CHAPTER 4

SOFTWARE DEVELOPMENT

4.1 INTRODUCTION

This tutorial will walk you through downloading, installing, and testing the Arduino software (also known as the Arduino IDE -short for Integrated Development Environment).Before you jump to the page for your operating system, make sure you've got all the right equipment. What you will need :A computer (Windows, Mac, or Linux). An Arduino-compatible microcontroller (anything from this guide should work) AUSB A-to-B cable, or another appropriate way to connect your Arduino compatible micro controller to your computer (check out this USB buying guide if you're not sure which cable to get).

Fig3.15: An A-to-BUS B Cable

If you're new to Arduino in general, you want to check out this tutorial to familiarize yourself with everyone's favorite microcontroller platform.

What is an Arduino?

If you're ready to get started, click on the link in the column on the left that matches up with your operating system, or you can jump to your operating system here.

- Windows
- Mac
- Linux

4.2 WINDOWS

This page will show you how to install land test the Arduino software with a Windows operating system (Windows 8, Windows 7, Vista, and XP).

4.2.1 WINDOWS 8,7,VISTA,ANDXP

- Go to the Arduino download page and download the latest version of the Arduino software for Windows.
- When the download is finished, un-zip it and open up the Arduino folder to confirm that yes, there are indeed some files and sub-folders inside. The file structure is important so don't be moving any files around unless you really know what you're doing.
- Power up your Arduino by connecting your Arduino board to your computer with a USB cable (or FTDI connector if you're using an Arduino pro). You should see the an LED labed 'ON' light up. (This diagram shows the placement of the power LED on the UNO).
- If you're running Windows8, you'll need to disable driver signing, so go see the Windows8 section. If you're running Windows 7, Vista, or XP, you'll need to install some drivers, so head to the Windows 7, Vista, and XP section down below.

4.2.2 WINDOWS8

Windows 8 comes with a nice little security 'feature' that 'protects' you from unsigned driver installation. Some older versions of Arduino Uno come with unsigned drivers, so in order to use your Uno, you'll have to tell Windows to disable driver signing. This issue has been addressed in newer releases of the Arduino IDE, but if you run into issues, you can try this fix first.

Foranice,step-by-steptutorialwithpicturesclickhere,otherwisethestepsareoutlinedbelow.

To temporarily disable driver signing:

- From the Metro Start Screen, open Settings (move your mouse to the bottom right-corner of the screen and wait for the pop-out bar to appear, then click the
- Click ‘More PC Settings’
- Click ‘General’.
- Scroll down, and click ‘Restart now’ under ‘Advanced start up’.
- Wait a bit.
- Click ‘Trouble shoot’.
- Click ‘Advanced Options’.
- Click ‘Windows Start up Settings’.
- Click Restart.
- When your computer restarts, select ‘Disable driver signature enforcement‘ from the list.

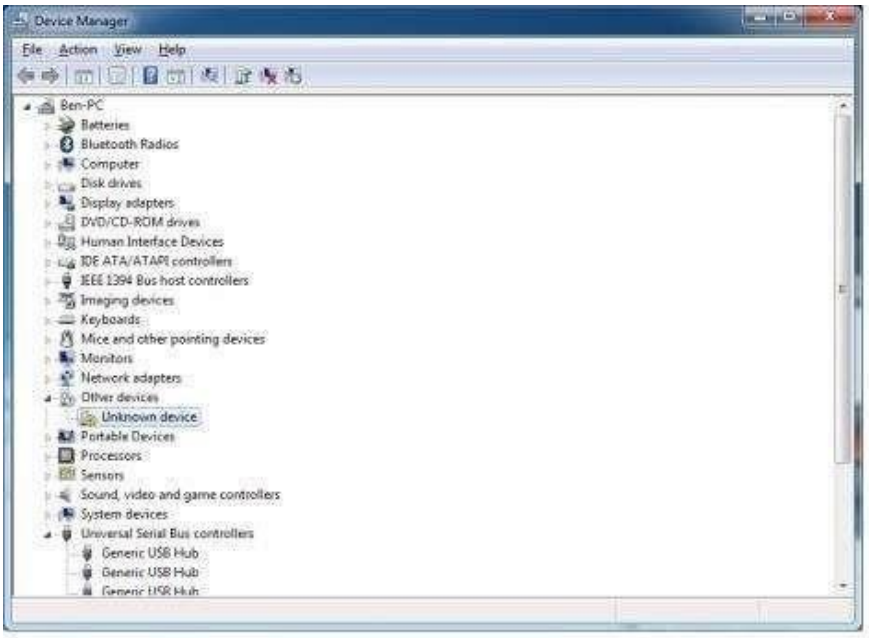
To permanently disable driver signing (recommended, but has some minor security implications):

- Go to the metro start screen.
- Type in“cmd”.
- Right click “Command Prompt” and select “Run as Administrator” from the buttons on the bottom of your screen.
- Type/paste in the following commands: bcdedit -set load options DISABLE_INTEGRITY_CHECKS bcdedit - set TESTSIGNING ON.
- Reboot!

4.2.3 WINDOWS7,VISTA,ANDXP

Installing the Drivers for the Arduino Nano (from Arduino.cc)

- Plug in your board and wait for Windows to begin it’s driver installation process.
- After a few moments, the process will fail, despite its best efforts.
- Click on the Start Menu, and open up the Control Panel.
- While in the Control Panel, navigate to System and Security. Next, clickon.
- System Once the System window is up, open the Device Manager.



- Look under Ports(COM& LPT).You should see an open port named "ESP32".If there is no COM & LPT section, look under 'Other Devices' for 'Unknown Device'
- Rightclickonthe"ESP32"or"UnknownDevice"portandchoosethe"Update Driver Software" option
- Next, choose the "Browse my computer for Driver software" option.

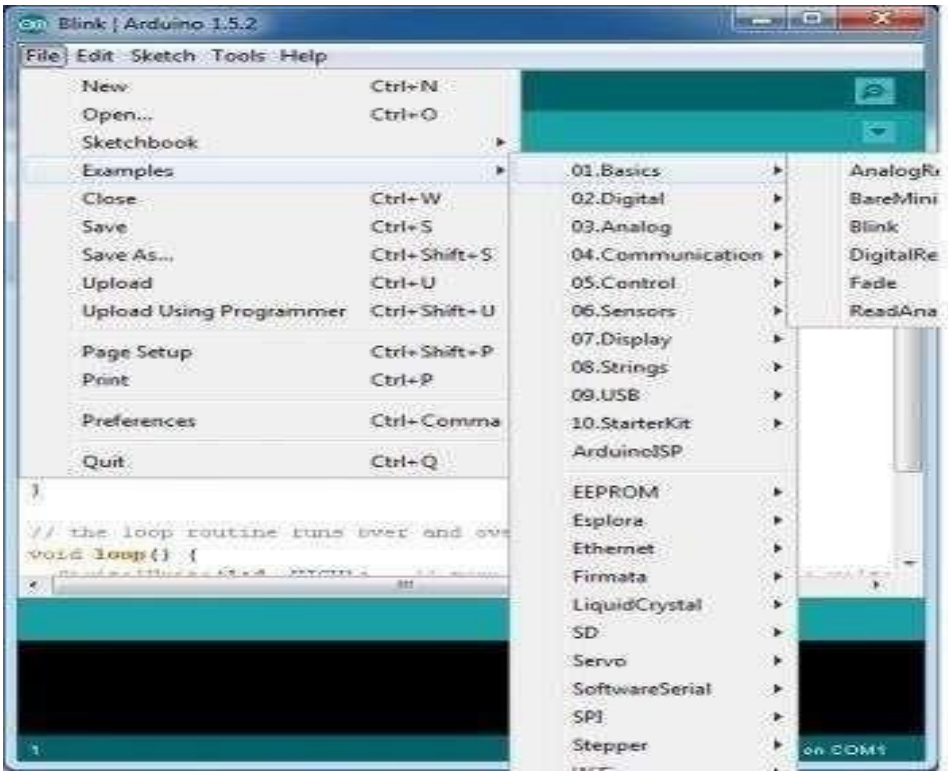

Update Driver Software - Unknown Device
How do you want to search for driver software?
Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.
Browse my computer for driver software
Locate and install driver software manually.

- Finally, navigate to and select the Uno's driver file, named "Arduino UNO. inf", located in the "Drivers" folder of the Arduino Software download(not the"FTDI USB Drivers" sub-directory). If you cannot see the.inf file, it is probably just hidden. You can select the 'drivers' folder with the 'search sub- folders' option selected instead.
- Windows will finish up the driver installation from there
- For earlier versions of the Arduino boards (e.g.Arduino Duemilanove, Nano,or Diecimila) check out this page for specific directions.

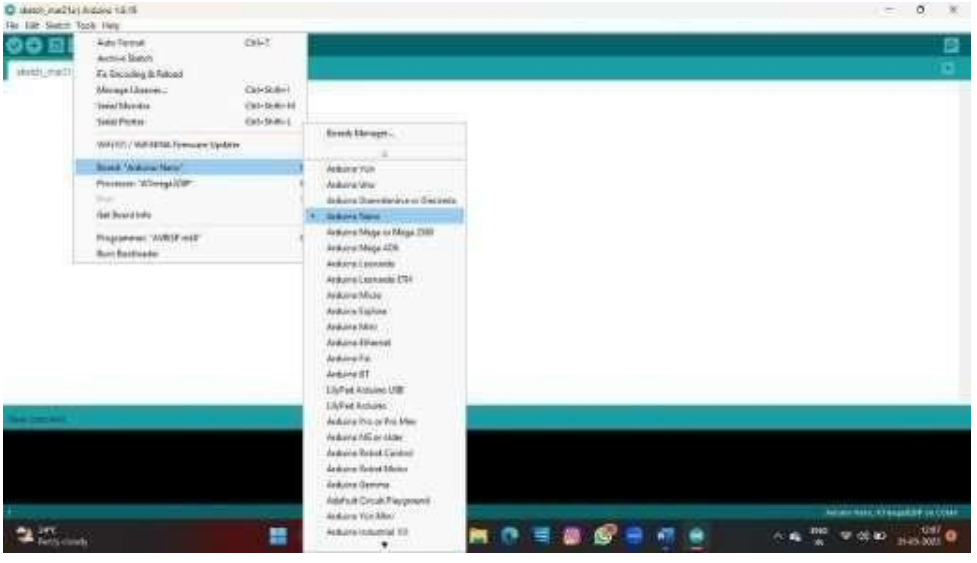
LAUNCH AND BLINK!

After following the appropriate steps for your software install, we are now ready to test your first program with your ESP32 board!

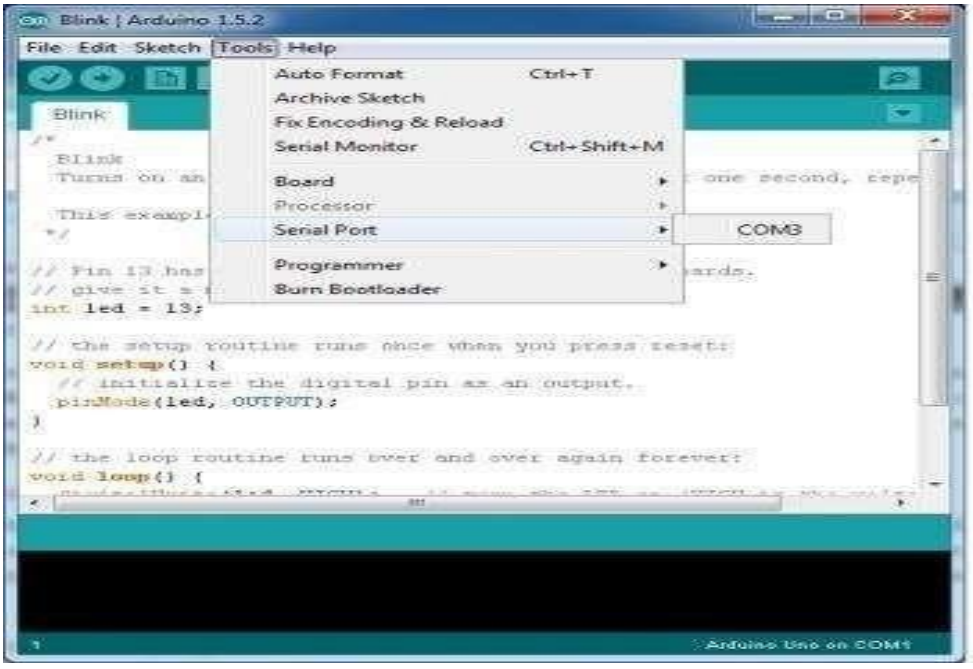
- Launch the ESP32 application
- If you disconnected your board, plug it back in.
- Open the Blink examples ketch by going to: File>Examples>1.Basics>Blink



- Select the type of ESP32 board you ’reusing:Tools>Board>yourboardtype



- Select the serial/COM port that your ESP32 is attached to:Tools>Port>COMxx



- If you're not sure which serial device is your Arduino, take a look at the available ports, then unplug your Arduino and look again. The one that disappeared is your Arduino.
- With your Arduino board connected, and the Blink sketch open, press the 'Upload' button.

- After a second , you should see some LEDs flashing on your ESP32, followed by the message 'Done Uploading' in the status bar of the Blink sketch.
- If everything worked, the on board LED on your ESP32 should now be blinking! You just programmed your first ESP32!

4.2.4 TROUBLE SHOOTING

This guide from ESP32 has some more details and trouble shooting tips if you get stuck.

- Go to the ESP32download page and download the latest version of the ESP32 software for Mac.
- When the download is finished, un-zip it and open up the ESP32 folder to confirm that yes, there are indeed some files and sub-folders inside. The file structure is important so don't be moving any files around unless you really know what you're doing.
- Power up yourESP32 by connecting your ESP32 board to your computer with a USB cable(or FTDI connect or if you 'reusing an ESP pro). You should see the

LED labed ‘ON’ light up. (this diagram shows the placement of the power LED on the UNO).

- Move the ESP32 application into your Applications folder.

4.2.5 FTDI DRIVERS

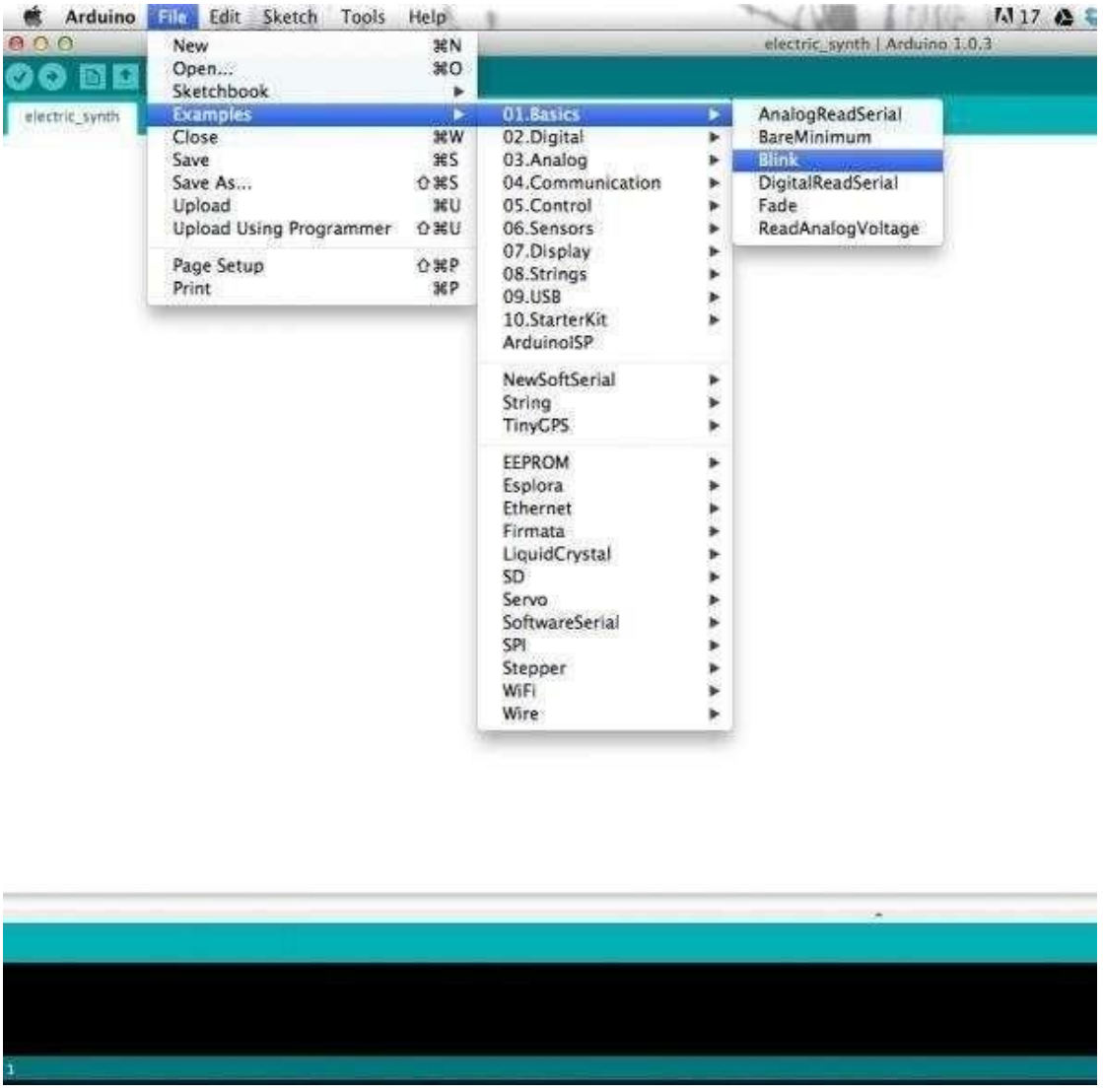
If you have an UNO, Mega 2560, or Red board, you shouldn’t need this step, so skipit!

- For other boards, you will need to install drivers for the FTDI chip on your ESP32.
- Go to the FTDI website and download the latest version of the drivers.
- Once you ’red one downloading, double click the package and follow the instructions from the installer.
- Restart your computer after installing the drivers.

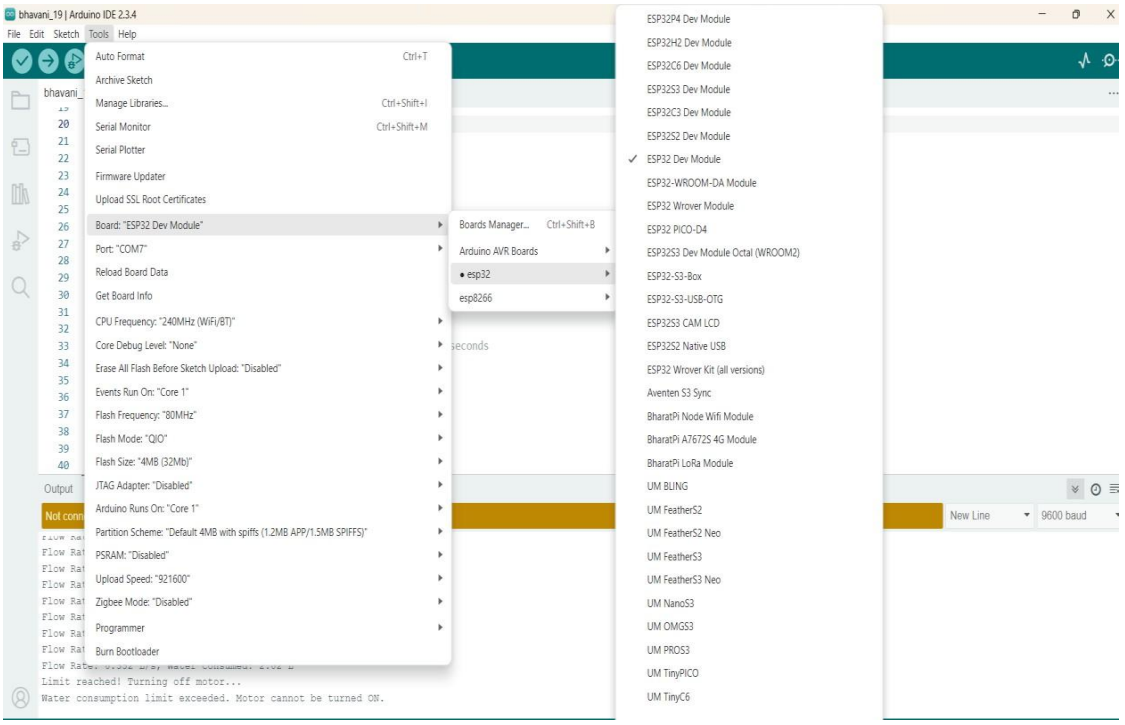
LAUNCH AND BLINK!

After following the appropriate steps for your software install, we are now ready to test your first program with your ESP32 board!

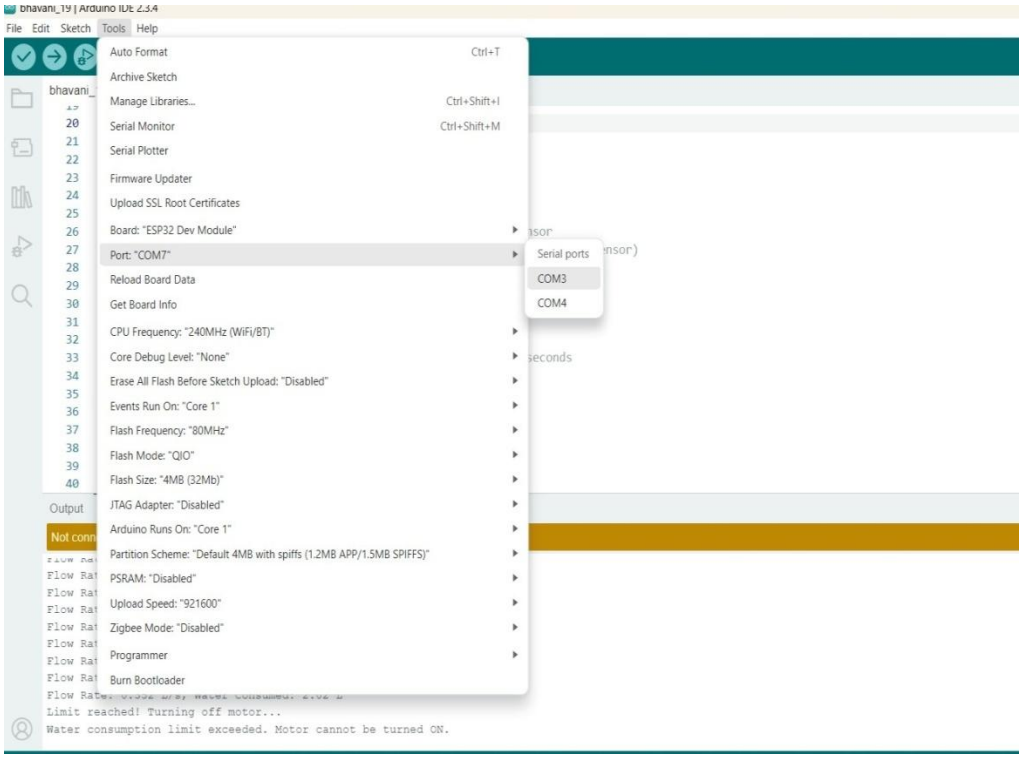
- Launch the ESP32 application.
- If you disconnected your board, plug it back in.



- Open the Blink example sketch by going to: File>Examples>1.Basics>Blin



- Select the serial port that your ESP32 is attached to: Tools > Port > xxxxxx (it'll probably look something like "/dev/tty.usbmodemfd131" or "/dev/tty.usbserial-131" but probably with a different number)



- If you're not sure which serial device is your ESP32,take a look at the available ports, then unplug your ESP32 and look again. The one that disappeared is your ESP32.
- With your ESP32 board connected and the Blink sketch open, press the 'Upload' button

- After a second, you should see some LEDs flashing on your ESP32, followed by the message ‘Done Uploading’ in the status bar of the Blink sketch.
- If everything worked, the on board LED on your ESP32 should now be blinking! You just programmed your first ESP32!

ESP32 is a popular open-source single-board micro controller. Learn how to program one and let the possibilities take shape.

STEP1

ESP32 micro controllers come in a variety of types. The most common is the ESP32, but there are specialized variations. Before you begin building, do a little research to figure out which version will be the most appropriate for your project.

STEP2

To begin, you'll need to install the ESP32 Programmer, the integrated development environment(IDE).

STEP3

Connect yourESP32 to the USB port of your computer. This may require a specific USB cable. Every ESP32 has a different virtual serial-port address, so you'll need to reconfigure the port if you're using different ESP32s.

STEP4

Set the board type and the serial port in the ESP32 Programmer.

STEP5

Test the micro controller by using one of the pre loaded programs, called sketches, in the ESP32 Programmer. Open one of the example sketches, and press the upload button to load it. The ESP32 should begin responding to the program: If you've set it to blink an LED light, for example, the light should start blinking.

STEP6

To upload new code to the ESP32, either you'll need to have access to code you can paste into the programmer, or you'll have to write it yourself, using the ESP32 programming language to create your own sketch. An ESP32 sketch usually has five parts: a header describing the sketch and its author ;a section defining variables; a set up routine that sets the initial conditions of variables and runs preliminary code;a loop routine, which is where you add the main code that will execute repeatedly until you stop running the sketch.

STEP7

Once you've uploaded the new sketch to your ESP32, disconnect it from your computer and integrate it into your project as directed.

4.3 Blynk APP:

The Blynk app can be used in the context of an automatic movable platform for crossing railway tracks to provide control and monitoring capabilities. With the Blynk app, users can have a mobile interface to interact with the platform and perform various functions.

For example, the Blynk app can allow users to remotely activate or deactivate the platform when they are about to cross the railway tracks. This can be done by tapping on buttons or switches within the app's interface. The app communicates with the platform's control system, sending the necessary commands to initiate the platform's movement. Additionally, the Blynk app can provide real-time monitoring of the platform's status.

Users can view information such as the current position of the platform, any safety alerts or notifications, and other relevant data. This allows users to have a visual representation of the platform's movement and ensures that they are aware of any changes or issues. Overall, the Blynk app acts as a user- friendly interface that enables convenient control and monitoring of the automatic movable platform for crossing railway tracks.

It enhances the user experience by providing easy access to platform functions and real-time information.

Users can view information such as the current position of the platform, any safety alerts or notifications, and other relevant data. This allows users to have a visual representation of the platform's movement and ensures that they are aware of any changes or issues. Overall, the Blynk app acts as a user- friendly interface that enables convenient control and monitoring of the automatic movable platform for crossing railway tracks.

This can be done by tapping on buttons or switches within the app's interface. The app communicates with the platform's control system, sending the necessary commands to initiate the platform's movement. Additionally, the Blynk app can provide real-time monitoring of the platform's status.

Users can view information such as the current position of the platform, any safety alerts or notifications, and other relevant data. This allows users to have a visual representation of the platform's movement and ensures that they are aware of any changes or issues. Overall, the Blynk app acts as a user- friendly interface that enables convenient control and monitoring of the automatic movable platform for crossing railway tracks.

It enhances the user experience by providing easy access to platform functions and real-time information.

4.3.1 CONSTRUCTION OF DATA STREAMS

Step1: Create a New Project.

1. Open the Blynk app and log into your account.
2. Tap on the "+" icon to create a new project.
3. Enter a name for your project and select the device type (e.g.,ESP32,Arduino,etc.).

Step2: Add a New Data Stream

1. Tap on the "Data Streams" tab in the project dashboard.
2. Tap on the "+" icon to add a new data stream.
3. Select the data stream type (e.g.,Virtual Pin, Physical Pin,etc.).

Step3:Configure the Data Stream

1. Enter a name for the data stream.
2. Select the data type (e.g., Integer, Float, String, etc.).
3. Set the data stream's properties, such as the update frequency and data range.

Step4: Add a Widget to the Data Stream

1. Tap on the "Widgets" tab in the project dashboard.
2. Select a widget type (e.g., Gauge, Chart, Button, etc.).
3. Drag and drop the widget on to the data stream.

Step5: Configure the Widget

1. Enter a name for the widget.
2. Set the widget's properties, such as the label, units, and color scheme.
3. Configure the widget's behavior, such as the update frequency and data range.

Step6:Connect the Data Stream to a Device

1. Tap on the "Devices" tab in the project dashboard.
2. Select the device that you want to connect to the data stream.
3. Configure the device's settings, such as the serial port and baudrate.

Step7:Test the Data Stream

1. Tap on the "Test" button to test the data stream.
2. Verify that the data stream is updating correctly.
3. Test the widget's behavior and verify that it is displaying the correct data.

Step8: Deploy the Data Stream

1. Tap on the "Deploy" button to deploy the data stream.
2. Verify that the data stream is deployed correctly.
3. Test the data stream in real-time to ensure that it is working as expected.

Here are some additional steps and details to help you construct data streams in Blynk

Step9 : Add Additional Data Streams

1. Repeat steps 2-8 to add additional data streams to your project.
2. You can add multiple data streams to a single widget or create separate widgets for each data stream.

Step10: Configure Data Stream Settings

1. Tap on the "Settings" icon next to each data stream to configure additional settings.

2. Set the data stream's update frequency, data range, and other properties as needed.

Step11: Use Data Stream Formulas

1. Tap on the "Formulas" icon next to each data stream to create custom formulas.

2. Use formulas to perform calculations, conversions, or other operations on your data streams.

Step12: Create Data Stream Alerts

1. Tap on the "Alerts" icon next to each data stream to create custom alerts.

2. Set up alerts to notify you when your data streams exceed certain thresholds or meet specific conditions.

Step13:Visualize Data Streams

1. Use Blynk's built-in visualization tools to create custom dashboards and graphs.

2. Visualize your data streams in real-time to gain in sights and make data-driven decisions.

Step14: Integrate with Other Services

1. Use Blynk's integrations with other services, such as IFTTT,Zapier,or Google Sheets.

2.Integrate your data streams with other services to automate work flows, send notifications, or store data.

Step15: Monitor and Analyze Data

1. Use Blynk's analytics tools to monitor and analyze your data streams.

2. Gain insights into your data streams, identify trends, and make data-driven decisions.

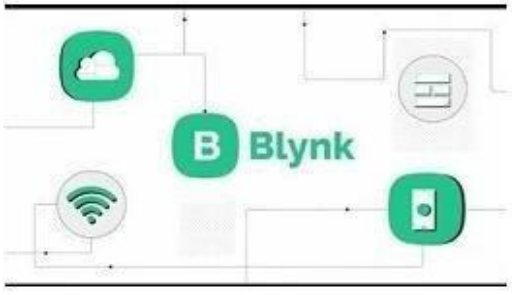


Fig3.15: Blynk App

Fig3.15: Blynk App Model

4.3.2 Benefits

- Easy to implement without extensive programming knowledge
- Cross-platform compatibility
- Scalable for both small personal projects and commercial applications
- Pre-built widgets to create interfaces without design skills.

4.3.3 Key Features

Blynk is a popular IoT platform that allows users to build mobile and web applications for controlling hardware devices remotely. Here are some key features of the Blynk app**:**

1. Cross-Platform Compatibility

- Works on iOS**,** Android**,** and web dashboards**.**
- Supports various microcontrollers and single-board computers like Arduino, ESP8266, ESP32, Raspberry Pi, and more.

2. Drag-and-Drop Interface

- No coding is required to design dashboards.
- Users can easily add buttons, sliders, graphs, and widgets**.**

3. Cloud Connectivity

- Blynk Cloud allows users to connect devices without setting up a local server.
- Secure and scalable cloud infrastructure.

4. Multiple Connectivity Options

- Supports Wi-Fi, Ethernet, GSM, LTE, Bluetooth, and BLE**.**
- Can work with MQTT for advanced IoT solutions.

5. Real-Time Data Monitoring & Control

- View live sensor data (temperature, humidity, etc.).
- Control actuators (relays, motors, LEDs) remotely.

6. Automation & Scheduling

- Create automations using simple rules (e.g., "Turn on light at 6 PM").
- Set up timers for devices to operate at specific times.

7. Notifications & Alerts

- Send push notifications**,** emails**,** and SMS alerts when a sensor detects a critical condition.

8. Multi-User Access & Sharing

- Share control of devices with multiple users.
- Assign roles and permissions.

9. Data Logging & Visualization

- Logs data for analysis.
- Use graphs, charts, and tables to visualize data trends.

10. Integration & API Support

- Supports integration with Google Assistant, Alexa, IFTTT, Node-RED, and Home Assistant.
- REST API for custom applications.

11. Edge & Offline Functionality

- Some devices can store and execute actions without an internet connection.

CHAPTER-5

EXPERIMENTAL RESULT

This system monitors water flow, controls a motor via a single phase double throw relay, and enforces consumption limits using the Blynk app. The system ensures efficient water usage by automatically switch off the motor when the limit is reached.

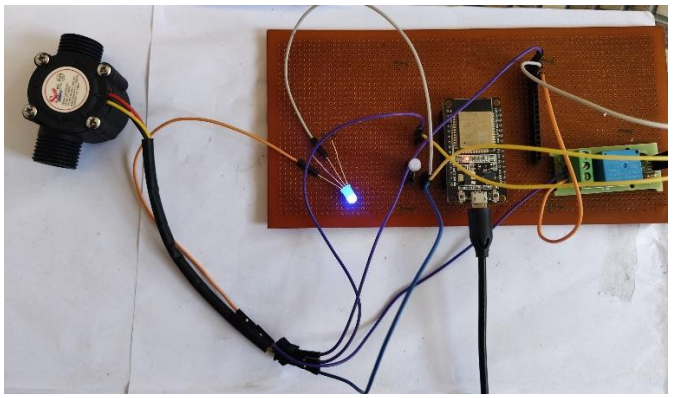


Fig3.15: L2 LED OFF indicates L1 motor ON

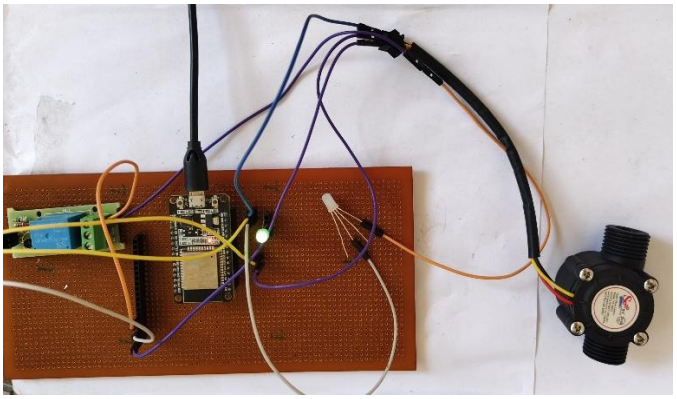


Fig3.15: L2 LED ON indicates L1 motor OFF

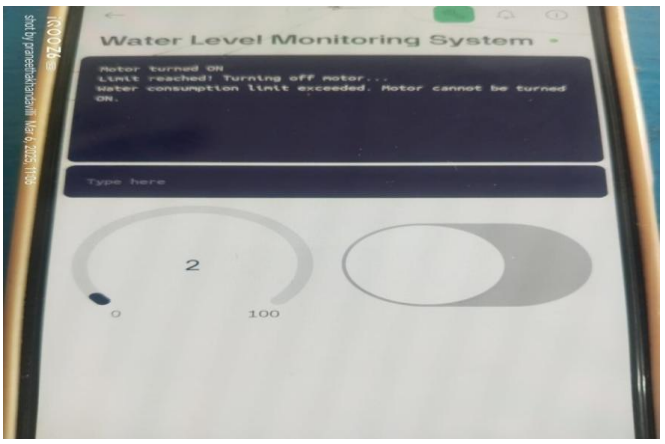


Fig3.15: Blynk Result

CHAPTER-6

ADVANTAGES AND APPLICATION

ADVANTAGES

A smart underground water management system offers several advantages, particularly in conserving water, improving efficiency, and ensuring sustainable usage. Here are some key benefits:

1. Water Conservation & Sustainability

- Helps prevent water wastage by monitoring consumption in real time.
- Automatically shuts off the motor when the water consumption exceeds a set limit.
- Supports sustainable water usage for households, industries, and agriculture.

2. Automation & Remote Monitoring

- Users can control the motor remotely using a smart phone app (e.g., Blynk).
- Automated on/off functionality based on water consumption and predefined limits.
- Reduces the need for manual intervention.

3. Real-Time Data Monitoring & Alerts

- Displays real-time water flow rate using sensors.
- Provides instant alerts if abnormal water usage is detected.
- Helps identify leaks or excessive consumption quickly.

4. Energy & Cost Efficiency

- Reduces electricity costs by running the motor only when needed.
- Prevents over-pumping**,** extending the lifespan of motors and pumps.
- Helps in budgeting water usage, reducing utility bills.

5. Smart Integration & IoT Connectivity

- Compatible with Blynk, cloud platforms, and smart home systems.
- Allows data logging, making it easy to analyze water usage patterns.
- Can be integrated with weather forecasting systems for optimized irrigation.

6. Leak Detection & Prevention

- Helps detect hidden leaks in underground pipes.
- Prevents water loss by automatically stopping the supply in case of abnormal flow.

7. Environmental Benefits

- Reduces groundwater depletion by ensuring responsible consumption.
- Lowers the risk of water scarcity in drought-prone areas.
- Supports eco-friendly water management practices**.**

8. Scalability & Customization

- Can be used in homes, apartments, industries, and agricultural fields**.**
- Allows users to set consumption limits based on needs**.**
- Can be expanded by adding more sensors and control features.

APPLICATIONS

- **Residential Water Management** –Controls and monitors household underground water usage efficiently.
- **Industrial Water Systems**–Manages water consumption in factories and plants to reduce waste and operational costs.
- **Municipal Water Supply**–Helps cities track underground water usage and detect leaks in pipelines.
- **Smart Buildings& Apartments** – Automates water distribution and monitors consumption in multi-story buildings.
- **Hotels & Commercial Spaces** – Ensures efficient water use in hospitality and business establishments.
- **Remote& Rural Areas**–Provides efficient water distribution where manual monitoring is difficult

CHAPTER-7

CONCLUSION AND FUTURESCOPE

CONCLUSION

A smart underground water management system is an efficient solution for monitoring and controlling water consumption. By integrating IoT technology, sensors, and automated controls, it helps conserve water, reduce wastage, and optimize motor operation. The system ensures sustainable water use in residential, agricultural, industrial, and municipal applications. With real-time monitoring and remote access via the Blynk app, it enhances efficiency, lowers costs, and promotes responsible water management for a sustainable future.

FUTURESCOPE

- **AI &Machine Learning Integration** –Predictive analytics can optimize water consumption and detect anomalies in usage patterns.
- **Cloud-Based Data Storage**– Enables long-term water usage analysis for better planning and resource management.

REFERENCES

Books:

- **"Smart Water Resource Management"**

  Authors: C. M .Hussain, A.K.Haghi

- **"Smart Water Technologies and Techniques: Data Capture and Analysis for Sustainable Water Management"**
  Editor*:* David A. Lloyd Owen

- **"Sustainable Water Engineering: Smart and Emerging Technologies"**

  Editors: Ali Fares, K.N.Ninan

Online Resources:

- "Applications of Smart Water Management Systems: A Literature Review".
- "Managing Urban Flooding through Smart Underground Drainage Systems: Technologies and Applications"
  "Open Storm: A Complete Frame work for Sensing and Control of Urban Watersheds"

Websites:

- **International Water Association(IWA)**

  Description: Provides resources and publications on water management, including smart technologies and sustainable practices.
- **Water Environment Federation(WEF)**

  Description: Offers technical resources, conferences, and publications related to water quality and management.

These resources offer valuable insights into the development, implementation, and benefits of smart underground water management systems.

CHAPTER-8 APPENDIX

SOURCESCODE

```
#define BLYNK_TEMPLATE_ID "TMPL3a_cYCKxG"
#define BLYNK_TEMPLATE_NAME "water level monitoring system"
#define BLYNK_AUTH_TOKEN "U2be0JuKP9923r1nbvkxnTiOM_VEow7O"
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
#include <BlynkSimpleEsp32.h>

char auth[] = "U2be0JuKP9923r1nbvkxnTiOM_VEow7O";  // Replace with your
Blynk Auth Token


// Define pin numbers
const int waterFlowPin = 34;  // Flow sensor pin
const int relayPin = 13;      // Relay control pin (motor)
const int ledPin = 14;        // LED indicator pin

// Blynk Virtual Pins
#define VIRTUAL_GAUGE V0


WidgetTerminal terminal(V2);

// Water consumption settings
const float consumptionLimit = 2.0;  // Limit (liters per day)
float waterConsumed = 0.0;           // Total liters consumed
volatile int pulseCount = 0;         // Pulse count from flow sensor
const float flowPerPulse = 0.0022;   // Liters per pulse (adjust for
your sensor)

// Timing variables
unsigned long previousMillis = 0;
const unsigned long interval = 1000;      // Update every 1 second
unsigned long lastResetTime = 0;
const unsigned long resetInterval = 300000; // 24 hours in milliseconds

// WiFi credentials
const char* ssid = "POCO M4 Pro";
const char* password = "1234567890";

// Web server setup
AsyncWebServer server(80);
```

```
// Interrupt service routine for pulse counting
void IRAM_ATTR pulseCounter() {
  pulseCount++;
}

void setup() {
  Serial.begin(9600);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, password);

  // Initialize sensor, relay, and LED
  pinMode(waterFlowPin, INPUT_PULLUP);
  pinMode(relayPin, OUTPUT);
  pinMode(ledPin, OUTPUT);  // Set LED as output

  digitalWrite(relayPin, HIGH);  // Initially, motor is OFF
  digitalWrite(ledPin, LOW);     // Initially, LED is OFF (motor ON)

  // Attach interrupt to flow sensor
  attachInterrupt(digitalPinToInterrupt(waterFlowPin), pulseCounter,
RISING);

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

  // Setup web server route
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    String html = "<h1>Smart Water Management</h1>";
    html += "<p>Water Consumed: " + String(waterConsumed, 2) + " L</p>";
    request->send(200, "text/html", html);
  });

  server.begin();
  lastResetTime = millis();
}
// Function to handle Blynk switch control (V1)
```

```
void loop() {
  Blynk.run();
  unsigned long currentMillis = millis();



// Reset water consumption daily
  if (currentMillis - lastResetTime >= resetInterval) {
    waterConsumed = 0.0;
    lastResetTime = currentMillis;
    Serial.println("Daily water consumption reset!");
    terminal.println("Daily water consumption reset! Turning on the
motor");
  }

  // Calculate water flow and consumption
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    float flowRate = (pulseCount * flowPerPulse) / (interval / 1000.0);
    pulseCount = 0;
    waterConsumed += flowRate;

     Blynk.virtualWrite(V0, waterConsumed);

    Serial.print("Flow Rate: ");
    Serial.print(flowRate, 3);
    Serial.print(" L/s, Water Consumed: ");
    Serial.print(waterConsumed, 2);
    Serial.println(" L");
    int switchState=0;

    Blynk.virtualWrite(VIRTUAL_GAUGE, waterConsumed);
    if ((waterConsumed >= consumptionLimit) ||(switchState==0)) {
      Serial.println("Limit reached! Turning off motor...");
      digitalWrite(relayPin, HIGH);
      digitalWrite(ledPin, HIGH);
      Blynk.logEvent("water_limit_exceeded", "Water consumption limit
reached!");
      Blynk.logEvent("reach_the_limit", "Water consumption limit
reached!");
       terminal.println("Limit reached! Turning off motor...");
    } else {
```

```
            digitalWrite(relayPin, LOW);
            digitalWrite(ledPin, LOW);

        }

    }
}
```