CHAPTER 1

INTRODUCTION

1.1. Existing System

In traditional home environments, weather and power monitoring systems are often separate, limited in functionality, and lack real-time communication capabilities. Electrical energy usage is typically tracked through analog meters or basic digital displays with no provisions for detailed analytics or remote control. Similarly, environmental parameters such as temperature and humidity are monitored using standalone devices that do not support data logging or networked communication.

Most existing smart home systems rely on cloud platforms or proprietary ecosystems, which may incur additional costs and reduce user control over data privacy and system customization. Furthermore, real-time feedback and automation features are minimal or nonexistent, resulting in inefficient energy usage and limited user engagement.

1.2. Proposed System

The proposed system addresses the limitations of existing solutions by integrating environmental monitoring and electrical energy management into a unified, real-time control platform. The system uses an ESP32-WROOM (38-pin) microcontroller to collect data from a DHT22 sensor (temperature and humidity) and a PZEM-004T module (voltage, current, power and energy).

This data is

- Displayed locally via a 20x4 I2C LCD
- Transmitted to a smartphone MQTT Panel app using the MQTT protocol

A 2-channel relay module allows users to control connected appliances remotely based on sensor inputs or manual commands, enabling automation and energy optimization. The system is compact, cost-effective, and powered by a 9V 1A transformer, making it suitable for smart home applications.

This proposed system enhances user control, promotes energy efficiency, and supports environmental awareness through real-time data acquisition and automation.

1.3. Overview of the Report

This report is structured to provide a comprehensive understanding of the Smart Weather and Power Consumption Monitoring System with Real-Time Control. The following chapters outline the development and implementation details

- **Chapter 1 Introduction**
  Discusses the background, objectives, scope, existing limitations, and the proposed solution.
- **Chapter 2 Literature Survey**
  Reviews existing technologies and related works in the domain of smart home automation, MQTT-based systems, and environmental monitoring.
- **Chapter 3 System Design**
  Describes the hardware components, software tools, block diagram, and system architecture used in the project.

- **Chapter 4 Implementation**

  Details the configuration, integration of sensors and modules, MQTT setup, and mobile application interfacing.

- **Chapter 5 Results and Discussion**

  Presents data outputs, system performance, advantages, and limitations.

- **Chapter 6 Conclusion and Future Work**

  Summarizes the project outcomes and suggests possible enhancements for scalability and advanced features.

CHAPTER 2

LITERATURE SURVEY

The integration of Internet of Things (IoT) technologies into smart home environments has gained significant attention in recent years. This chapter presents a review of existing studies and solutions related to environmental monitoring, power consumption tracking, and MQTT-based communication systems.

1 **Banks, A., & Gupta, R. (2014).** ***MQTT Version 3.1.1 Protocol Specification*****, OASIS Standard, pp. 1–84.**

This foundational specification outlines the MQTT protocol, designed for lightweight communication in constrained networks. It publish/subscribe architecture enables efficient, real-time message delivery ideal for IoT systems. The proposed project employs MQTT to facilitate seamless data exchange between the ESP32 microcontroller and remote user interfaces, ensuring low-latency and minimal overhead communication.

2 **Gupta, K., & Singh, S. (2019).** ***Weather Monitoring using Raspberry Pi*****, Proceedings of the International Conference on IoT and Smart Systems, pp. 22–26.**

This study implemented a weather monitoring system using Raspberry Pi and cloud services for data visualization. While functional, the system consumed significant power and incurred high deployment costs, limiting its suitability for low-power smart homes. The proposed project addresses

these issues by using the more efficient ESP32 microcontroller and MQTT, reducing both cost and power consumption.

3 **Kale, S., Jadhav, A., & Patil, M. (2020). *Home Automation using MQTT Protocol*, International Journal of Computer Applications, Vol. 176(18), pp. 1–5.**

This paper explored the use of ESP8266 and MQTT for automating basic home appliances. Although it successfully demonstrated remote control using MQTT, it lacked integration with environmental or power monitoring systems. The proposed project expands on this work by incorporating DHT22 and PZEM-004T sensors for comprehensive monitoring alongside automation features.

4 **Khan, A., & Iqbal, M. (2021). *Multi-parameter Smart Home Monitoring System*, Journal of Electronics and Communication Engineering, Vol. 10(3), pp. 50–54.**

The authors developed a multi-functional smart home system using Blynk for remote access. It monitored temperature, gas levels, and security breaches, but did not incorporate power monitoring or MQTT communication. The proposed system improves upon this by adding electrical monitoring with PZEM-004T and adopting MQTT for more scalable and efficient communication.

5 **Kumar, P., & Reddy, R. (2020). *Smart Energy Meter using PZEM Module*, Proceedings of the IEEE Conference on Power Systems, pp. 131–135.**

This research introduced an energy monitoring system using the PZEM-004T module, which effectively measured voltage, current, and power consumption. However, it lacked remote control and integration with environmental monitoring. The proposed system includes both electrical

and environmental data tracking, along with MQTT-based control features.

6 **Mehta, P., & Rana, D. (2017). *Zigbee-Based Energy Meter Monitoring System*, International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol. 6(11), pp. 65–69.**

In this study, Zigbee was used for wireless energy monitoring. While Zigbee offered low-power communication, its limited range and scalability reduced its effectiveness for modern smart home use. The proposed system replaces Zigbee with Wi-Fi and MQTT, offering greater scalability and performance for real-time monitoring and control.

7 **Sharma, A., & Patel, R. (2018). *IoT-based Temperature and Humidity Monitoring System*, International Journal of Engineering Research & Technology (IJERT), Vol. 7(4), pp. 1–4.**

This work focused on basic environmental monitoring using a DHT11 sensor. While successful in data collection, it lacked automation and remote monitoring features. The proposed project addresses this by using a more accurate DHT22 sensor and enabling real-time data access and control via MQTT.

Summary of Findings

The reviewed literature demonstrates growing interest in smart systems for home automation, environmental monitoring, and energy management. However, gaps exist in

- Combining environmental sensing and electrical monitoring in a single system.

- Using MQTT for low-latency, real-time control without dependence on cloud platforms.
- Enabling user-friendly interfaces such as mobile MQTT panels for intuitive control.

The proposed project bridges these gaps by developing a unified, MQTT-enabled system using ESP32, capable of both monitoring and controlling in real time, while ensuring low power consumption and system scalability.

Research Gap and Proposed Solution

The proposed project addresses these shortcomings by

- Integrating environmental monitoring (via DHT22) and electrical parameter tracking (via PZEM-004T).
- Utilizing the ESP32 microcontroller for low-power, scalable processing.
- Implementing an MQTT-based communication system for real-time data updates and remote control.

CHAPTER 3

PROJECT OVERVIEW

3.1 BLOCK DIAGRAM OF THE PROPOSED IOT-BASED MONITORING AND CONTROL SYSTEM

- The proposed system is designed to monitor both weather parameters and electrical power consumption in real time using IoT. The main controller of the system is the ESP32 microcontroller, which acts as the central hub for data acquisition, processing, display, and communication.
- The AC power supply feeds the system, where an AC to DC converter provides regulated power to the ESP32 and associated components. A DHT22 sensor is connected to the ESP32 for measuring ambient temperature and humidity. In parallel, a PZEM-004T power sensor is used to measure voltage, current, power, and energy consumed by the connected load. This sensor communicates with the ESP32 through the UART protocol.
- The ESP32 processes the incoming sensor data and displays it on a 20×4 LCD module, which is interfaced using the I2C protocol to minimize wiring. A 2-channel relay module is also connected to the ESP32, enabling control of connected electrical appliances (load). Manual control can be done via a push button, while automated or remote control is possible through the cloud.
- The ESP32 is configured to connect to the internet using Wi-Fi. It communicates with a remote MQTT broker, allowing data to be published and subscribed through MQTT topics. A mobile device or web interface can then be used to monitor real-time data and control appliances remotely through this MQTT communication.

- This setup creates a complete real-time monitoring and control system that is cost-effective, scalable, and energy-efficient, making it suitable for smart home and energy management applications.

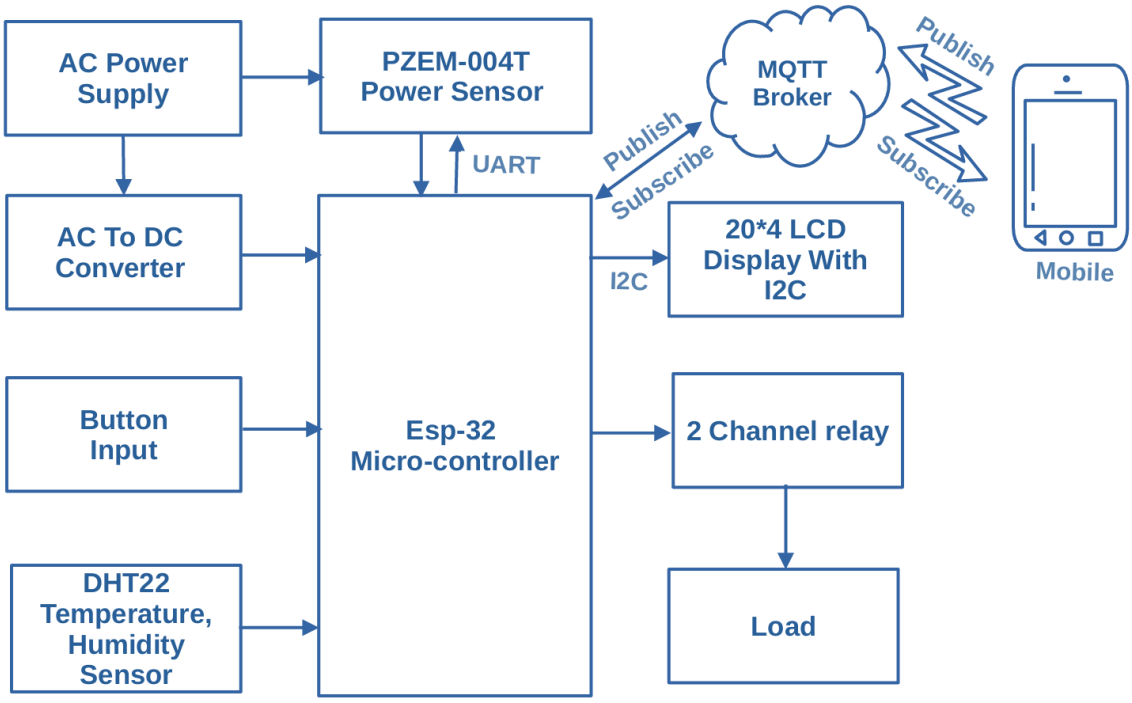

Figure 3.1 Block Diagram of The Proposed Iot-Based Monitoring and Control System

3.2 CIRCUIT DIAGRAMOF THE PROPOSED SMART IOT MONITORING SYSTEM

- The circuit diagram represents the complete hardware setup of the Smart Weather and Power Consumption Monitoring System with Real-Time Control. It uses an ESP32 microcontroller as the central controller that interfaces with various sensors, display units, power supplies, and output control devices.

- The system is powered through a 12V 1A step-down transformer, which converts the AC mains voltage to 12V AC. This AC voltage is rectified using a bridge rectifier and further filtered and regulated using a buck converter to provide a stable 5V DC supply for powering the ESP32, sensors, and modules.

- A DHT22 temperature and humidity sensor is connected to the ESP32 to monitor environmental conditions. A PZEM-004T sensor is used to measure power parameters such as voltage, current, power, and energy. It works along with a CT (current transformer) and is connected to the ESP32 via a UART interface.

- For output display, a 20x4 LCD with an I2C module is used. This reduces wiring complexity and allows real-time sensor data and system status to be shown clearly. The 5V 2-channel relay module connected to the ESP32 is used to control two electrical appliances, in this case represented as Bulb-1 and Bulb-2.

- All components are properly grounded and wired to ensure safe operation. The ESP32 communicates the collected data to an external device (like a smartphone or cloud dashboard) via Wi-Fi using MQTT protocol, enabling remote monitoring and control of appliances.

- This circuit demonstrates a complete low-cost IoT-based solution for smart home energy monitoring and automation.

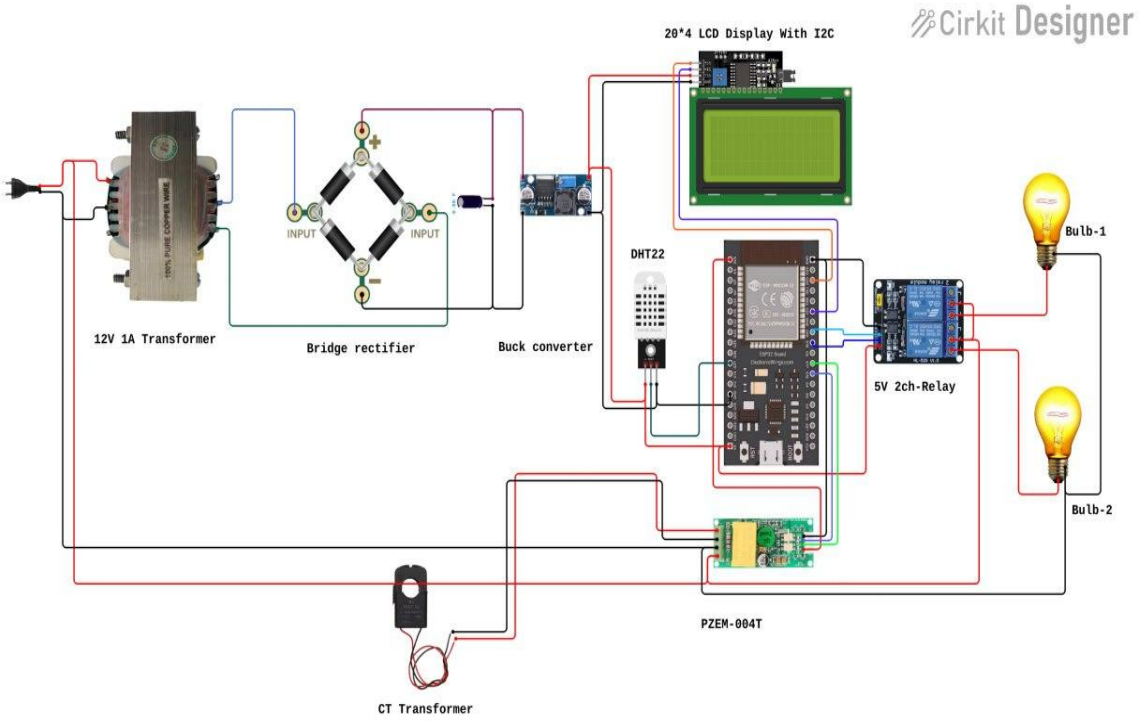

Figure 3.2 Circuit Diagram of The Proposed Smart IOT Monitoring System

CHAPTER 4

HARDWARE USED

4.1 ESP32-WROOM-32 (38-Pin Module)

- The ESP32-WROOM-32 is a powerful, general-purpose Wi-Fi and Bluetooth microcontroller module based on the ESP32 SoC from Espressif Systems. This 38-pin module is highly integrated with 2.4 GHz Wi-Fi, Bluetooth (Classic and BLE), and a dual-core Xtensa® 32-bit LX6 microprocessor. It is optimized for low-power and IoT applications, supporting both Arduino IDE and the native ESP-IDF platform for firmware development.
- The ESP32-WROOM-32 module includes an onboard PCB antenna, voltage regulators, and a USB-to-serial interface (CP2102 chip), which allows easy programming and debugging through a standard micro-USB connection. With 4 MB of external flash memory and a rich set of peripherals, it is widely used in embedded systems and smart devices such as IoT weather stations, power monitoring systems, and home automation.

Overview

The ESP32-WROOM-32 is a powerful Wi-Fi + Bluetooth dual-core MCU module developed by Espressif Systems. It integrates all the core components required for many applications such as smart energy meters, home automation, remote sensor networks, and wireless data logging. This model features robust wireless connectivity, a wide range of input/output (I/O) options, and an ultra-low-power co-processor, making it a flexible platform for IoT solutions.

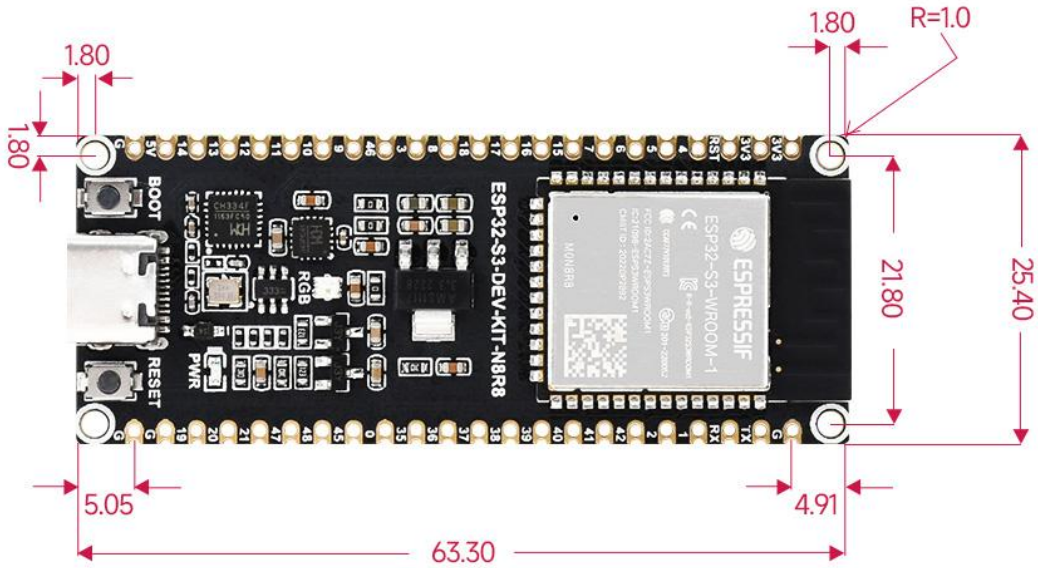

Figure 4.1: Esp 32

Table 4.1: Board Specifications

Feature	Specification
Power Supply (USB)	5V DC
Operating Voltage (I/O)	3.3V DC
Recommended Current	≥ 500 Ma
Microcontroller SoC	ESP32-WROOM-32
CPU	Xtensa® dual-core 32-bit LX6
Clock Frequency	80 – 240 MHz
SRAM	512 KB

Feature	Specification
Flash Memory	4 MB
GPIO Pins	34
ADC Channels	18 (12-bit resolution)
DAC Channels	2 (8-bit resolution)
Communication Interfaces	UART, SPI, I2C, I2S, CAN
Wi-Fi	802.11 b/g/n (up to 150 Mbps)
Bluetooth	v4.2, Classic + BLE
Antenna	Onboard PCB
Dimensions	56 x 28 x 13 mm

Pin Configuration and Descriptions

The ESP32-WROOM-32 (38-pin) module provides a wide range of functional I/O pins, suitable for digital input/output, analog input, touch sensing, and communication.

Pin Overview

- GPIO Pins (Digital I/O) Operate at 3.3V logic level. Avoid applying 5V directly to any GPIO pin, as it may damage the chip.
- Input-only Pins GPIOs 34–39 is input-only (no internal pull-up/down resistors).
- ADC Pins The module supports 18 channels of 12-bit ADC, including GPIOs 32–39 (ADC1) and GPIOs 0–15, 25–27 (ADC2).
- DAC Pins GPIO25 (DAC1) and GPIO26 (DAC2) for 8-bit DAC output.
- Capacitive Touch Pins GPIOs 0, 2, 4, 12–15, 27, 32, and 33.
- RTC (Real-Time Clock) Pins Several GPIOs (e.g., 0, 2, 4, 12–15, 25–27, 32–39) support RTC functions for wake-up from deep sleep.
- PWM (Pulse Width Modulation) Available on all output-capable GPIOs (except GPIOs 34–39).
- I2C Interface Configurable on any GPIO; default is GPIO21 (SDA) and GPIO22 (SCL).
- SPI Interface Common pin mapping for VSPI is GPIO23 (MOSI), GPIO19 (MISO), GPIO18 (CLK), and GPIO5 (CS).
- Strapping Pins GPIOs 0, 2, 5, 12, and 15 affect boot mode. Ensure correct logic levels during power-up for successful flashing.
- Boot behaviour Some GPIOs output PWM or go HIGH during boot (e.g., GPIOs 1, 3, 5, 14, 15); avoid connecting relays/LEDs directly to these without buffers.

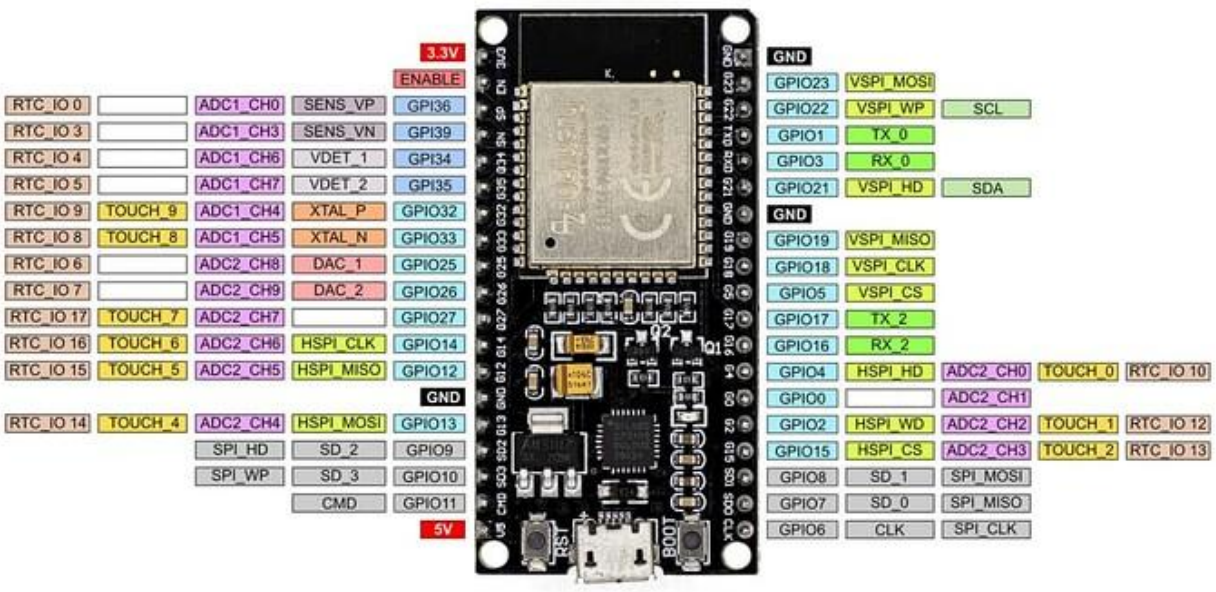

Figure 4.2: ESP 32 Pin Out

Table 4.2: Functions and Notes

GPIO	Function(s)	Notes
0	Boot, Input, ADC2_CH1	Must be LOW during boot (strapping pin)
2	ADC2_CH2, DAC1	Often used for onboard LED
4	I²C (SDA), ADC2_CH0, Touch	Used commonly for sensors or buttons
5	SPI (CS), PWM	Must be HIGH during boot strapping
12	ADC2_CH5, Touch	Must be LOW during boot strapping
13	SPI, ADC2_CH4, Touch	Good general-purpose pin
14	SPI, PWM, Touch	Often used with motor drivers
15	Boot, PWM, Touch	Must be HIGH during boot strapping

GPIO	Function(s)	Notes
16–17	UART2	Safe for general use
18–19	SPI Bus (VSPI)	Fast SPI applications
21	I²C (SDA)	Default SDA pin
22	I²C (SCL)	Default SCL pin
23	SPI MOSI	Used in SPI communications
25–27	DAC2, ADC2, PWM	Excellent analog and PWM outputs
32–39	ADC1 Only (Input Only)	Best for analog sensors; input-only pins

Key Features and Specifications

- Wi-Fi 802.11 b/g/n
- Bluetooth v4.2 BR/EDR and BLE
- CPU Dual-core 32-bit Xtensa® LX6 microprocessor, up to 240 MHz
- Flash Memory 4MB
- SRAM 520KB internal
- Operating Voltage 3.0–3.6V
- Input Supply 5V (via onboard voltage regulator)
- GPIOs 34 programmable I/O pins
- ADC Channels 18 (12-bit)
- DAC Channels 2 (8-bit)
- Touch Sensors 10 capacitive touch GPIOs
- Timers 4 × 64-bit general-purpose timers
- Interfaces 3 × UART, 3 × SPI, 2 × I2C, 2 × I2S, CAN, PWM
- RTC Yes, with ULP co-processor

- Security AES, SHA-2, RSA, ECC, RNG
- Operating Temperature Range -40°C to +125°C

I²C Communication Interface

The ESP32 supports two I²C bus interfaces that can be mapped to any GPIO

➢ **Default I²C Pins**

- SDA GPIO21
- SCL GPIO22

➢ **I²C Features**

- Multi-master capability
- Clock stretching
- Standard (100 kbps) and Fast mode (400 kbps)
- Slave mode available

This makes it highly versatile for connecting peripherals like LCD displays, sensors, and EEPROMs.

SPI Communication Protocol

➢ **Number of SPI Buses** 4 (SPI0, SPI1, HSPI, VSPI)

- **SPI0 and SPI1** Used internally for flash
- **HSPI and VSPI** Available for user applications

- **Default VSPI Pins**

Table 4.3: VSPI Pins

Function	Pin
MISO	GPIO19
MOSI	GPIO23

Function	Pin
SCLK	GPIO18
CS	GPIO5

- **Default HSPI Pins**

Table 4.4: HSPI Pins

Function	Pin
MISO	GPIO12
MOSI	GPIO13
SCLK	GPIO14
CS	GPIO15

- **Max Clock Speed** Up to 80 MHz
- **Mode**
- Master/Slave
- **Use Cases** High-speed displays (TFT), SD card readers, external ADCs/DACs, flash storage

Note All SPI signals are remappable to any GPIO using `SPI.begin()` with parameters.

UART Communication Protocol

- **Total UART Interfaces** 3 (UART0, UART1, UART2)
- **Features**
- Configurable baud rate up to 5 Mbps
- Hardware flow control (RTS/CTS)
- 9-bit data support
- DMA support

- **Default UART Pins**

Table 4.5: UART Pins

UART	TX Pin	RX Pin
UART0	GPIO1	GPIO3
UART1	GPIO10*	GPIO9*
UART2	GPIO17	GPIO16

*GPIO9 and GPIO10 are used for flash on some modules, so reassign if needed.

Use Cases Serial debugging, communication with GSM/GPS modules, serial sensors, RS-232/RS-485.

Power Supply and Regulation

- **USB Power Input** 5V via micro-USB
- **Onboard Regulator Output** 3.3V regulated for internal logic
- **Current Requirements**
    - Idle ~80 Ma
    - Peak Wi-Fi TX ~240 Ma
    - Deep Sleep <10 µA

Note Ensure regulated 3.3V on GPIOs. Connecting 5V directly can damage the chip.

Wireless Communication Wi-Fi and Bluetooth

- **Wi-Fi Specifications**
- Standards 802.11 b/g/n (2.4 GHz)
- Modes Station, Soft-AP, or both
- Max Speed 150 Mbps
- WPA/WPA2 security
- SmartConfig and WPS provisioning
- **Bluetooth Specifications**
- Bluetooth v4.2 (BR/EDR and BLE)
- BLE supports multiple simultaneous connections
- Bluetooth stack included in firmware

Real-Time Clock and Deep Sleep

The ESP32 features an RTC subsystem with ultra-low-power support. Deep sleep mode uses the RTC and ULP co-processor to minimize power.

- **RTC GPIOs**
    - GPIOs 0, 2, 4, 12–15, 25–27, 32–39
- Wake-up Sources
        - RTC timer
        - External GPIO (RTC IO)
        - Touch sensor
        - ULP co-processor
        - Deep Sleep power consumption can go as low as 10 μA.

Touch and Analog Sensors

- **Capacitive Touch Inputs**
    - GPIOs 0, 2, 4, 12–15, 27, 32, 33
    - Use cases Touch buttons, sliders, proximity detection
- **ADC**
- 18 channels, 12-bit resolution
- ADC1 GPIOs 32–39
- ADC2 GPIOs 0, 2, 4, 12–15, 25–27
- Note ADC2 is disabled when Wi-Fi is in use
- **DAC**
- GPIO25 (DAC1), GPIO26 (DAC2)

Hall Effect and Temperature Sensor

The ESP32 includes internal sensors

- Hall Sensor Detects magnetic field variations; connected to internal ADC.
- Temperature Sensor
    - Range -40°C to +125°C
    - Used for internal diagnostics; not accurate for precise ambient temperature**.**

PWM (Pulse Width Modulation)

- 16 PWM channels
- Resolution 1 to 16 bits
- Output on any GPIO except 34–39
- Ideal for motor control, LED dimming, etc.

ESP32 GPIO Pinout Summary Table

Table 4.6: Esp32 GPIO

Function	GPIO Pins
ADC1	32–39
ADC2	0, 2, 4, 12–15, 25–27
DAC	25 (DAC1), 26 (DAC2)
Touch Sensors	0, 2, 4, 12–15, 27, 32, 33
PWM	All GPIOs except 34–39
I²C	Any GPIO (Default 21-SDA, 22-SCL)
SPI	VSPI (23, 19, 18, 5), HSPI (13–15)
UART	1 (TX0), 3 (RX0), 16, 17
RTC Wakeup	0, 2, 4, 12–15, 25–27, 32–39

Application Scenarios for ESP32-WROOM-32

- Home Automation Lighting control, door locks, appliance monitoring
- Environmental Monitoring Temperature, gas and humidity sensors
- Industrial IoT Energy metering, machinery telemetry
- Wearables BLE-enabled health and fitness devices
- Smart Agriculture Soil moisture, weather monitoring

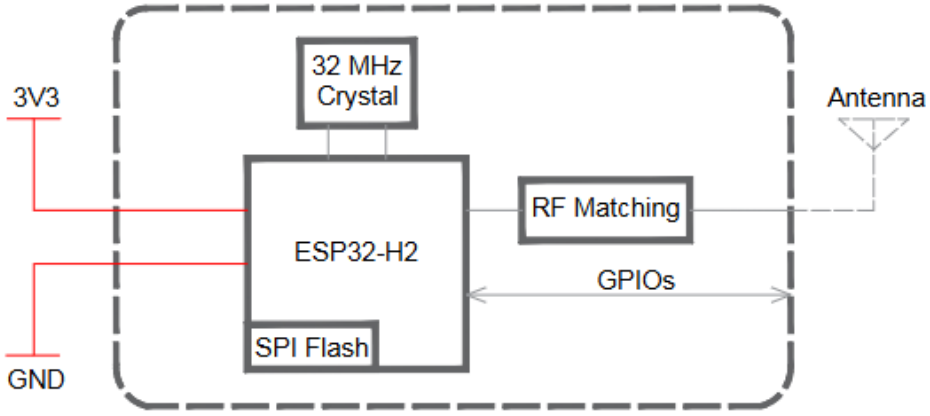

Figure 4.3: WROOM-32 Block Diagram

4.2 DHT22 Temperature and Humidity Sensor

Overview

The DHT22 (also known as AM2302) is a low-cost, digital-output sensor that provides accurate measurements of temperature and humidity. It is a commonly used sensor in embedded systems and IoT applications where environmental monitoring is required. The DHT22 uses a capacitive humidity sensor and a thermistor to measure the surrounding air and provides a digital signal on a single data line, ensuring simple interfacing with microcontrollers such as Arduino, ESP8266, ESP32, and others.

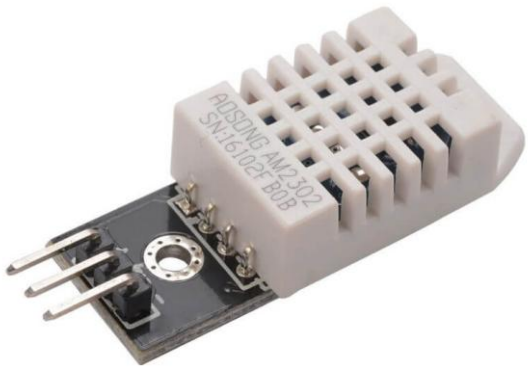

Figure 4.4 DHT22: Temperature and Humidity Sensor

Technical Specifications

Table 4.7: Technical Specifications-DHT22

Parameter	Value
Temperature Range	-40°C to +80°C
Temperature Accuracy	±0.5°C
Humidity Range	0% to 100% RH
Humidity Accuracy	±2% RH
Temperature Resolution	0.1°C
Humidity Resolution	0.1% RH
Operating Voltage	3.3V to 6V DC
Maximum Current during Data Transmission	2.5 Ma
Sampling Rate	Once every 2 seconds (0.5 Hz)
Output Signal	Digital (single wire, proprietary protocol)
Dimensions (typical)	15.1 × 25 × 7.7 mm

Note The DHT22 provides better accuracy and a wider measurement range than its predecessor, the DHT11.

Pin Configuration

The DHT22 sensor typically comes with 4 pins, but only 3 are used in most applications

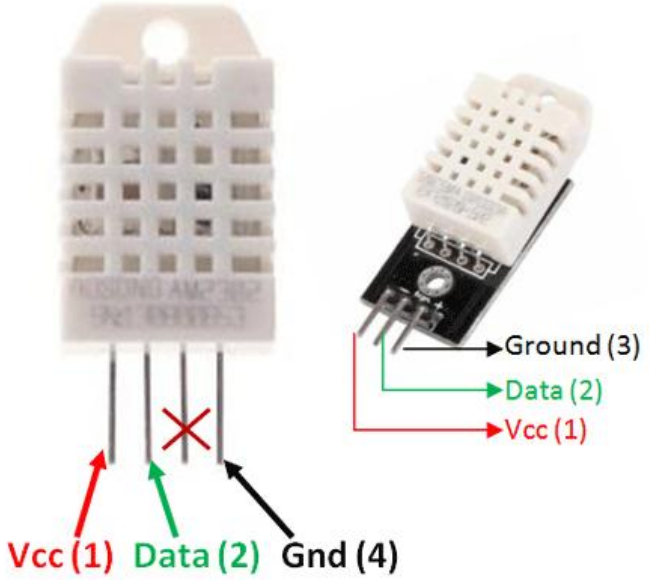

Figure 4.5: DHT-22 Pin Out

Pin	Name	Description
1	VCC	Connect to 3.3V or 5V power supply
2	DATA	Digital signal output (connect to MCU pin)
3	NC	Not connected (leave unconnected)
4	GND	Ground

A **10Kω pull-up resistor** is recommended between the **VCC and DATA** lines for stable data transmission.

Key Features

- Dual-function sensor Measures both temperature and humidity.
- Single-wire digital interface Easy integration with microcontrollers.
- High accuracy ±0.5°C for temperature and ±2% for humidity.
- Wide operating range Suitable for indoor and outdoor environments.
- Low power consumption Ideal for battery-powered applications.
- Compact form factor Easy to embed in portable and fixed systems.
- Long-term stability Reliable measurements over extended periods.

Applications

- Weather stations
- Smart home automation systems
- HVAC monitoring and control
- Environmental data logging
- Greenhouse and agriculture control
- Laboratory instruments
- IoT-based climate monitoring systems

4.3 PZEM-004T Energy Monitoring Module

Overview

The PZEM-004T is a compact and reliable AC power monitoring module designed to measure key electrical parameters in real-time, including voltage, current, power, energy (kWh), frequency, and power factor. It is widely used in smart energy monitoring projects and IoT-based power management systems. The module communicates via UART (TX/RX), making it ideal for integration with microcontrollers like ESP32, ESP8266, or Arduino. It uses a non-invasive CT (Current Transformer) sensor, allowing current measurement without physically interrupting the power line.

Figure 4.6: PZEM-004T Sensor

Technical Specifications

Table 4.8: Technical Specifications-PZME-004t

Parameter	Value
Voltage Measurement	80V to 260V AC
Current Measurement	0 to 100A (using external CT sensor)
Frequency Range	45 Hz to 65 Hz
Power Factor	0.00 to 1.00
Active Power	0 to 23Kw
Energy (kWh) Range	0 to 9999.99 kWh
Communication Protocol	UART (9600 baud rate, 8N1)
Supply Voltage	5V DC
Power Consumption	< 1W
Measurement Accuracy	±0.5%
Interface Isolation	Optical isolation
Dimensions	Approx. 48mm × 29mm × 22mm
Operating Temperature	-10°C to +60°C

Note For current measurement, the module must be used with the included CT sensor (usually 100A/1V).

Pin Configuration

The PZEM-004T module typically has 4 pins for UART communication and power supply

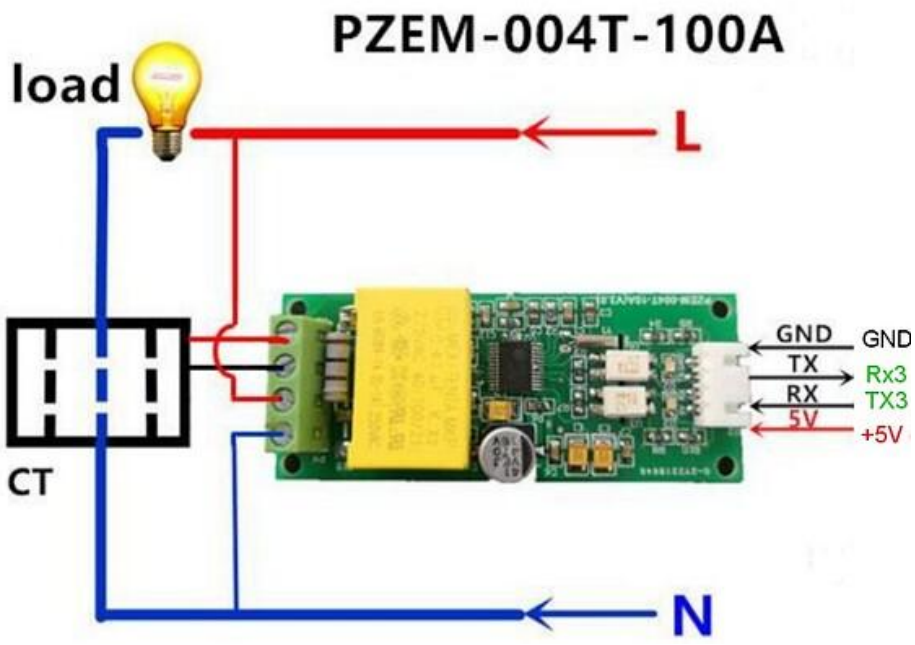

Figure 4.7: PZEM-004T Pin Out

Pin	Name	Description
1	VCC	Connect to 5V DC power supply
2	GND	Ground
3	TX	Transmit data from PZEM to microcontroller
4	RX	Receive data from microcontroller to PZEM

Connect TX of PZEM to RX of the microcontroller and RX of PZEM to TX of the microcontroller.

It also has terminals for

- AC input voltage connection (Live and Neutral terminals)
- CT sensor input (jack or screw terminal for clip-on current sensor)

Key Features

- 📊 Real-time monitoring of voltage, current, power, energy, frequency, and power factor.
- ⚡ Non-invasive current sensing via CT sensor (safe and easy to install).
- 🔌 AC voltage input range 80V–260V AC.
- 📡 Serial (UART) communication compatible with 5V logic microcontrollers.
- 🔒 Electrical isolation via optocoupler ensures safe operation.
- 🛠️ Compact design for easy embedding in custom enclosures.
- 💾 Energy (kWh) data retention even after power loss.
- 🔄 Supports reset of energy data via command through UART.

Applications

- Home energy monitoring systems
- Industrial power tracking and logging
- Smart grid and smart home applications
- Load analysis and energy optimization projects
- IoT-based power consumption dashboards
- Protective monitoring for high power equipment

4.4 LCD 20x4 With I^2C

An LCD (Liquid Crystal Display) is a display technology widely used in embedded systems and microcontrollers due to its low power consumption and compact design. LCDs are much thinner and lighter compared to CRTs and consume significantly less power than LED or gas-plasma displays because they operate by blocking light rather than emitting it.

LCDs can be built with either a passive matrix or an active-matrix configuration. The active matrix, also known as TFT (Thin Film Transistor), places a transistor at each pixel, allowing for faster switching and improved screen refresh rates compared to passive matrices.

Figure 4.8: 20x4 LCD Display

The 20x4 LCD display shows 20 characters across 4 lines, making it ideal for applications requiring more text output compared to a 16x2 LCD. It is commonly used with microcontrollers such as the Arduino for visual feedback like displaying sensor data, status messages, or debug information.

I^2C Module Interfacing

Microcontrollers like the Arduino have a limited number of I/O pins. A standard parallel LCD requires many digital pins (10 or more), which can be limiting. However, by using an I^2C interface, you can control the LCD with just two data lines (SDA and SCL), leaving more pins free for sensors and other peripherals.

The I2C 20x4 LCD uses an I^2C backpack module, which simplifies wiring and reduces pin usage

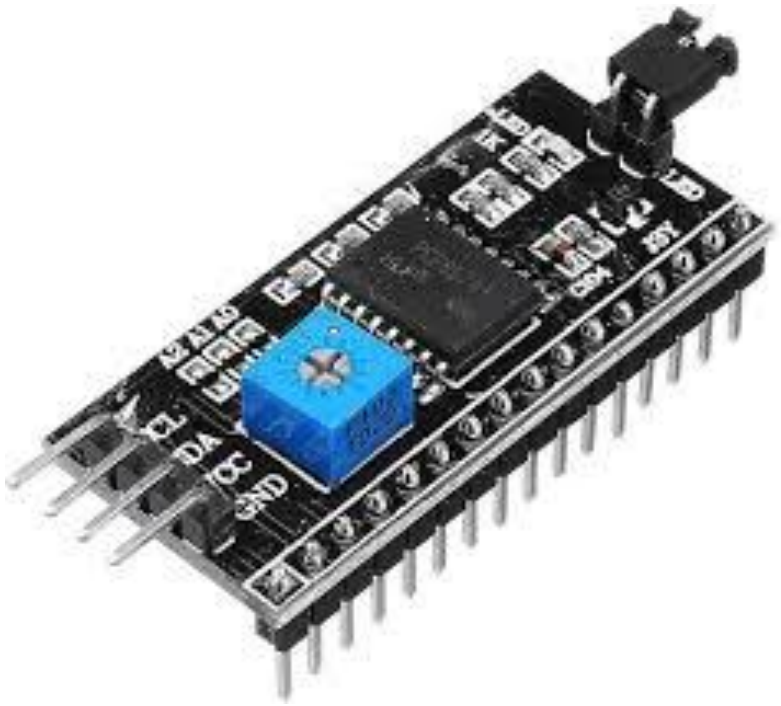

Figure 4.9: I^2C Module

I²C Pin Out

- SDA (Serial Data Line)
- SCL (Serial Clock Line)
- VCC (5V Power Supply)
- GND (Ground)

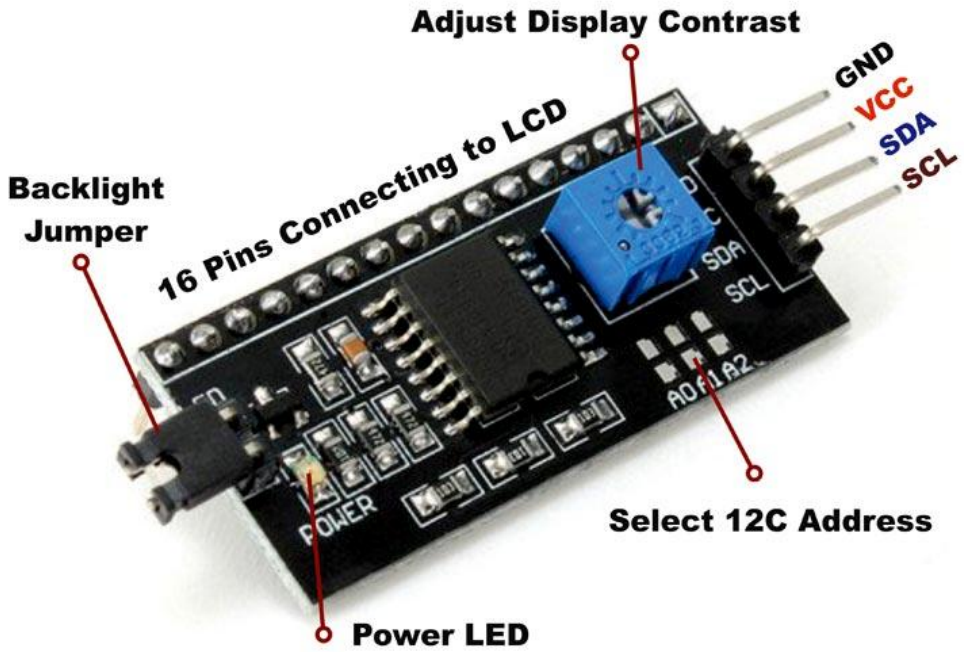

Figure 4.10: I²C Pin Out

These displays come with a PCF8574 I/O expander chip that handles I²C communication and includes an adjustable potentiometer to control display contrast. The default I²C address is usually 0x27, but this may vary.

Advantages of Using a 20x4 I^2C LCD

- Displays more data compared to 16x2 (80 characters total vs. 32).
- Frees up I/O pins using I^2C (only 2 data pins needed).
- Simple wiring with 4-pin connection.
- Adjustable backlight and contrast.
- Supports custom characters using a 5x8 dot matrix per character.

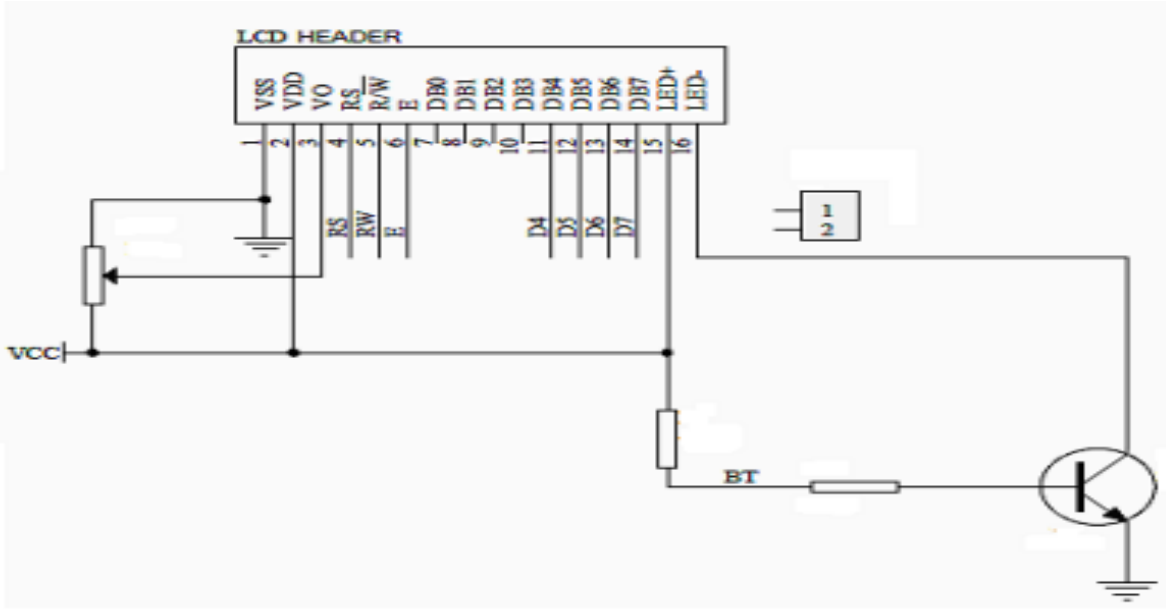

Figure 4.11: Block Diagram of LCD Display

Board Specifications (for 20x4 I^2C LCD)

- Compatible Boards Arduino UNO, Mega, Nano, etc.
- Display Size 20 characters × 4 lines
- Supply Voltage 5V
- Communication Interface I2C (2 wires – SDA, SCL)
- I^2C Address Typically, 0x27 (may vary)
- Character Colour Black
- Backlight Colour Yellow (or as per model)
- Size (Arduino.) 98 × 60 × 22 mm
- Adjustable contrast using onboard potentiometer

➢ I^2C Pin Details

- VCC – Connects to 5V
- GND – Ground
- SDA – Data line
- SCL – Clock line

4.5 Relay Module – 5V

A 5V relay is an automatic switch that is commonly used in an automatic control circuit and to control a high-current using a low-current signal. The input voltage of the relay signal ranges from 0 to 5V. The relay module with a single channel board is used to manage high voltage, current loads like solenoid valves, motors, AC load, and lamps. This module is mainly designed to interface with different microcontrollers like PIC, Arduino, etc.

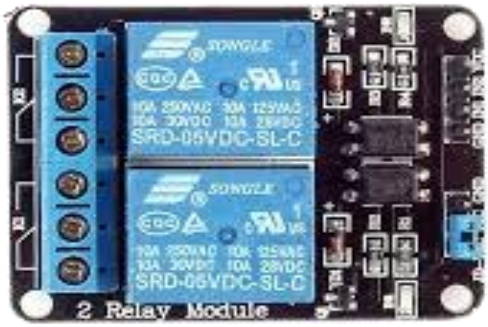

Figure 4.12: Picture of Relay

Specifications

- Voltage supply ranges from 3.75V – 6V
- Quiescent current is 2Ma
- Once the relay is active then the current is ~70Ma
- The highest contact voltage of a relay is 250VAC/30VDC
- The maximum current is 10A
- It includes 5-pins & designed with plastic material
- Operating time is 10msec
- Release time is 5msec
- Maximum switching is 300 operations per minute

Pin Configuration

- **Normally Open (NO)** This pin is normally open unless we provide a signal to the relay modules signal pin. So, the common contact pin smashes its link through the NC pin to make a connection through the NO pin.
- **Common Contact** This pin is used to connect through the load that we desire to switch by using the module.
- **Normally Closed (NC)** This NC pin is connected through the COM pin to form a closed circuit. However, this NC connection will break once the relay is switched through providing an active high/low signal toward the signal pin from a microcontroller.

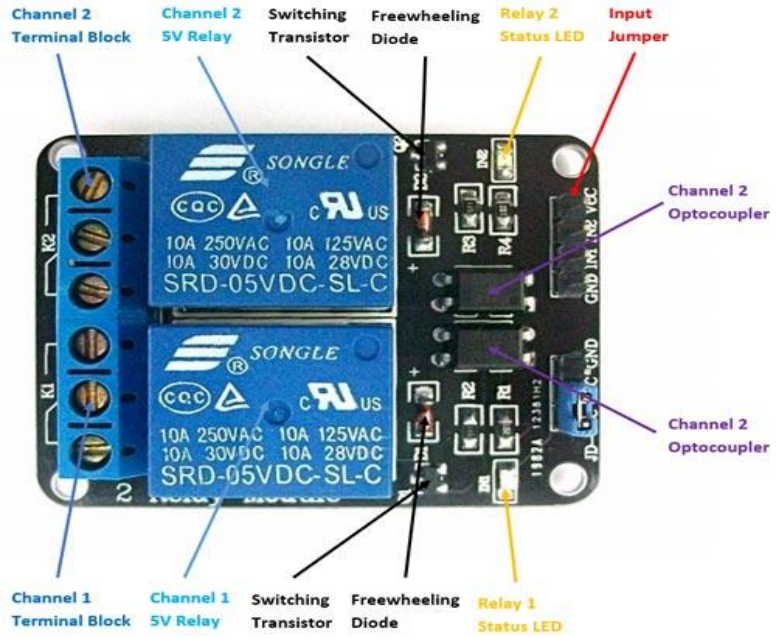

Figure 4.13 Pin Diagram of 5V Relay

- **Signal Pin** The signal pin is mainly used for controlling the relay. This pin works in two cases like active low or otherwise active high. So, in an active low case, the relay activates once we provide an active low signal

toward the signal pin, whereas, in an active high case, the relay will trigger once we provide a high signal toward the signal pin.

- However, these modules generally work on an active high signal which will strengthen the relay coil to make contact with the common terminal with the normally open terminal.
- **5V VCC** This pin needs 5V DC to work. So 5V DC power supply is provided to this pin.
- **Ground** This pin connects the GND terminal of the power supply.

Relay Module Components

- **Relay**

At the heart of the module is a 5V relay covered in blue-coloured plastic. Maximum operating current and voltage for both AC and DC load are also mentioned at the top of the relay cover. SRD-05VDC-SL-C is the part number and it shows the operating voltage. It is known as a 5V relay module. Because the relay operates at 5V DC. In other words, a 5V active high or low signal activates the relay by energizing its coil. As mentioned earlier, internally a 5V relay consists of NC, NO, COM terminals and a coil.

- **Output Terminal**

On the left-hand of this figure is an output terminal, which is used to connect a DC/AC load and DC/AC input power source. We will discuss the wiring diagram to connect a LOAD and power source with this terminal in later sections of this tutorial. Each terminal of the output connector is connected with NO, NC, and COM pins of 5V relay. Each point of the module has screws that make it easy to connect cables and wires with the relay module. This 5V relay module supports 10A maximum output current and maximum contact voltage

of 250V AC and 30V DC. If you are using a high AC voltage and high current load with this module, you should use thick main cables.

- **Status LED**

The status LED is SMD LED that is connected through current limiting resistor and it is available on the top right corner of the module. It shows the status of the relay. In other words, the status LED turns on when the relay is active and the coil is energized through a signal input pin. The DC current passes through a relay coil.

- **Power LED**

The power LED is also a SMD type and it shows the status of the power source connected with the 5V single channel relay module. Do not connect more than a 5V source to the VCC and GND pins of the module. Otherwise, higher voltage may damage the status and power LEDs.

- **Freewheeling Diode**

A freewheeling diode is connected across the coil to avoid the effect of back EMF. It is also known as a flyback diode. The coil used in the relay is an inductive type. When the current passes through an inductive load, it produces a back EMF voltage. This back EMF may damage the circuit. Therefore, a freewheeling diode is used to avoid this effect.

- **Input Connector**

On the right-hand side of the relay module is an input connector. It is used to provide an input signal and 5V power supply. Furthermore, it also provides power to the status LED, power LED, and relay coil.

- **Switching Transistor**

We usually provide an input signal to a relay from the general-purpose input-output pins of microcontrollers such as Arduino, TM4C123, ESP32, etc. But the maximum current sourcing capability of GPIO pins is generally less than

20 mA. Hence, a switching transistor is used in this relay module to amplify current to the level of the minimum current requirement of the relay coil. By using a switching transistor, we can control the relay from the GPIO pin of a microcontroller.

Working Principle

The relay uses the current supply for opening or closing switch contacts. Usually, this can be done through a coil to magnetize the switch contacts & drag them jointly once activated. A spring drives them separately once the coil is not strengthened.

By using this system, there are mainly two benefits. The first one is that the required current for activating the relay is less as compared to the current used by relay contacts for switching. The other benefit is that both the contacts & the coil are isolated galvanically, which means there is no electrical connection among them.

Advantages

- A remote device can be controlled easily
- It is triggered with less current but it can also trigger high power machines
- Contacts can be changed Easily
- At a time, several contacts can be controlled using a single signal
- Activating part can be isolated
- It can switch AC or DC
- At high temperatures, it works very well

Applications

- Used in over voltage/under voltage protection system
- Mains Switching
- Speed control of motors through start-delta converters
- Automatic electrical appliances
- Electrical isolation in between high & low power sources
- Lights
- AC voltage load switching using less voltage DC
- Delivery of isolated power
- Home automation projects
- Switching with High Current

4.6 12 1A Step Down Transformer

Overview

0-12 1A Step Down Transformer is a general-purpose chassis mounting mains transformer. Transformer has 230V primary winding and non-centre tapped secondary winding. The transformer has flying, coloured insulated connecting leads. It acts as step down transformer reducing AC – 230V to AC – 12V.

Figure 4.14: Transformer

The transformer gives outputs of 12V. a static electrical device that transfers energy by inductive coupling between winding circuits. A varying current in the primary winding creates a varying magnetic flux in the transformer's core and a varying magnetic flux through the secondary winding. This varying magnetic flux induces a varying electromotive force or voltage in the secondary winding. The cores are made of high permeability silicon steel. The steel has a permeability many times that of free space and the core thus serves to greatly reduce the magnetizing current and confine the flux to a path that closely couples the windings.

Specifications

- Input Voltage – 230V AC
- Output Voltage – 12V or 0V
- Output Current – 1 Amp
- Mounting – Vertical mount type

Applications

- DIY projects requiring in-application High current drain.
- On chassis AC/AC converter.
- Designing a battery charger.

4.7 LM2596 Power Supply Module

The LM2596 series of regulators are monolithic integrated circuits that provide all the active functions for a step-down (buck) switching regulator, capable of driving a 3A load with excellent line and load regulation. These devices are available in fixed output voltages of 3.3V, 5V, 12V, and an adjustable output version. The LM2596 series operates at a switching frequency of 150 kHz thus allowing smaller-sized filter components than what would be needed with lower-frequency switching regulators. Available in a standard 5-lead TO-220 package with several different lead bend options and a 5-lead TO-263 surface mount package

A standard series of inductors is available from several different manufacturers optimized for use with the LM2596 series. This feature greatly simplifies the design of switch-mode power supplies. Other features include a guaranteed ±4% tolerance on output voltage under specified input voltage and output load conditions and ±15% on the oscillator frequency. External shutdown is included, featuring typically 80 μA standby current. Self-protection features include a two-stage frequency-reducing current limit for the output switch and an over-temperature.

Features

- 3.3V, 5V, 12V, and adjustable output versions
- Adjustable version output voltage range, 1.2V to 37V ±4% max over line and load conditions
- Available in TO-220 and TO-263 packages
- Guaranteed 3A output load current
- Input voltage ranges up to 40V

- Requires only 4 external components
- Excellent line and load regulation specifications
- 150 kHz fixed frequency internal oscillator
- TTL shutdown capability
- Low power standby mode, IQ typically 80 µA
- High efficiency
- Uses readily available standard inductors
- Thermal shutdown and current limit protection.

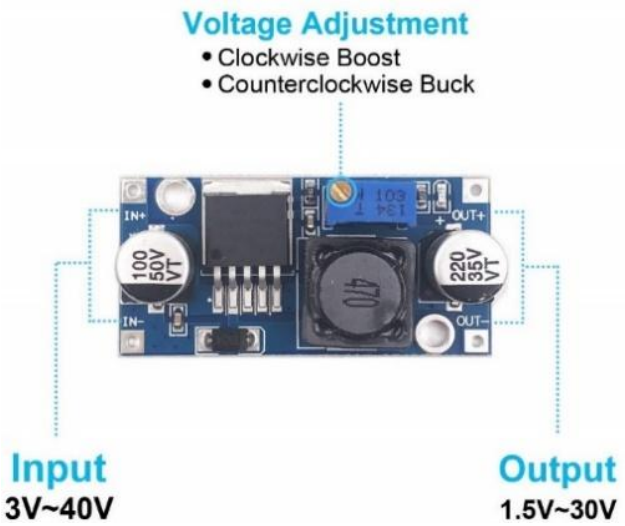

Figure 4.15: Pin Diagram of LM2596 Module

Pin Function Description

- **V in** – This pin is the positive input supply for the LM2596 step−down switching regulator. In order to minimize voltage transients and to supply the switching currents needed by the regulator, a suitable input bypass capacitor must be present.

- **Output** – This is the emitter of the internal switch. The saturation voltage V sat of this output switch is typically 1.5 V. It should be kept in mind that the PCB area connected to this pin should be kept to a minimum in order to minimize coupling to sensitive circuitry.

GND – Circuit ground pin. See the information about the printed circuit board layout.

- **Feedback** – This pin is the direct input of the error amplifier and the resistor network R2, and R1 is connected externally to allow programming of the output voltage.

- **ON/OFF** – It allows the switching regulator circuit to be shut down using logic level signals, thus dropping the total input supply current to approximately 80 A. The threshold voltage is typically 1.6 V. Applying a voltage above this value (up to +Vin) shuts the regulator off. If the voltage applied to this pin is lower than 1.6 V or if this pin is left open, the regulator will be in the “on” condition.

Figure 4.16 Picture of LM2596 Module

Application

- Simple High−Efficiency Step−Down (Buck) Regulator
- Efficient Pre−Regulator for Linear Regulators
- On−Card Switching Regulators
- Positive to Negative Converter (Buck−Boost)
- Negative Step−Up Converters
- Power Supply for Battery Chargers
- At least 4001 or more.

CHAPTER 5

SOFTWARE REQUIREMENTS

5.1 ARDUINO IDE

The Arduino integrated development environment or Arduino software (IDE) is a cross-platform application (for Arduino, Windows, macOS, and Linux) that is written in the java programming language. It originated from the IDE for the languages 'processing and wiring'. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board.

It contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino and Arduino hardware to upload programs and communicate with them. The source code for the IDE is released under the GNU General Public License, version 2.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main () into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program argued to convert the executable code into a text file in hexadecimal

encoding that is loaded into the Arduino board by a loader program in the board's firmware.

Arduino pro ide (alpha preview) was released. Later, on March 1, 2021, the beta preview was released, renamed ide 2.0. The system still uses Arduino CLI (command line interface), but improvements include a more professional development environment, autocompletion support, and git integration. The application frontend is based on the Eclipse Theia Open-Source IDE.

Writing Sketches

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension is (.ino). The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors.

The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.NB Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension (.pde). It is possible to open these files with version 1.0; you will be prompted to save the sketch with the .ino extension on save.

Verify

Check your code for errors compiling it.

Upload

Compiles your code and uploads it to the configured board. See uploading below for details.

Note If you are using an external programmer with your board, you can hold down the “shift” key on your computer when using this icon. The text will change to “Upload using Programmer.”

New

Creates a new sketch.

Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window, overwriting its content.

Note due to a bug in Java, this menu doesn’t scroll; if you need to open a sketch late in the list, use the File Sketchbook menu instead.

Save

Saves your sketch.

Serial Monitor

Opens the serial monitor.

Additional commands are found within the five menus File, Edit, Sketch, Tools and Help. The menus are Context-Sensitive, which means only those items relevant to the work currently being carried out are available.

File

New

Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.

Open

Allows loading a sketch file by browsing through the computer drives and folders.

Open Recent

Provides a short list of the most recent sketches, ready to be opened.

Sketchbook

Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.

Examples

Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.

Close

Closes the instance of the Arduino Software from which it is clicked.

Save

Saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as." window.

Save as…

Allows you to save the current sketch with a different name.

Page Setup

It shows the Page Setup window for printing.

Print

Sends the current sketch to the printer according to the settings defined in Page Setup.

Preferences

Opens the Preferences window where some settings of the IDE may be customized, such as the language of the IDE interface.

Quit

Closes all IDE windows. The same sketches that where open when quit was chosen will be automatically reopened the next time you start the IDE.

Undo/Redo

Go back one or more steps you did while editing; when you go back, you may go forward with Redo.

Cut

Removes the selected text from the editor and places it into the clipboard.

Copy

Duplicates the selected text in the editor and places it into the clipboard.

Copy for Forum

Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax colouring.

Copy as HTML

Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.

Paste

Puts the contents of the clipboard at the cursor position in the editor.

Select All

Selects and highlights the whole content of the editor.

Comment/Uncomment

Puts or removes the // comment marker at the beginning of each selected line.

Increase/Decrease Indent

Adds or subtracts a space at the beginning of each selected line, moving the text one space to the right or eliminating a space at the beginning.

Find

Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.

Find Next

Highlights the next occurrence – if any – of the string specified as the search item in the Find window, relative to the cursor position.

Find Previous

Highlights the previous occurrence – if any – of the string specified as the search item in the Find window relative to the cursor position.

Verify/Compile

Check your sketch for errors compiling it; it will report memory usage for code and variables in the console area.

Upload

Compiles and loads the binary file onto the configured board through the configured Port.

Upload Using Programmer

This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to upload to the USB serial port again. However, it allows you to use the full capacity of the flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so, a Tools -> Burn Bootloader command must be executed.

Export Compiled Binary

Saves a .hex file that may be kept as an archive or sent to the board using other tools.

Show Sketch Folder

Opens the current sketch folder.

Include Library

Add a library to your sketch by inserting #include statements at the start of your code. For more details, see libraries below. Additionally, from this menu item, you can access the Library Manager and import new libraries from .zip files.

Add File...

Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible by clicking on the small triangle icon below the serial monitor one on the right side of the toolbar.

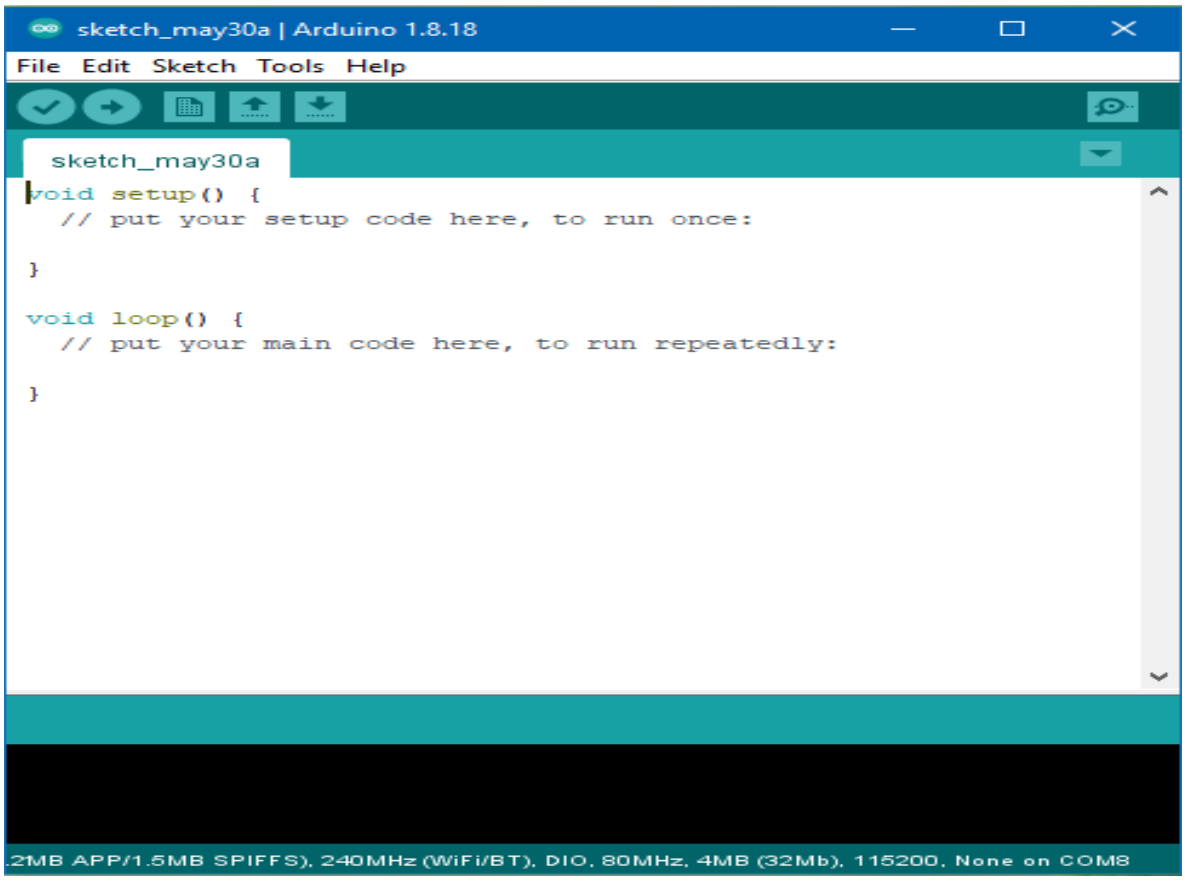

Figure 5.17: Arduino IDE

Tools

Auto Format

This formats your code nicely, i.e., indents it so that opening and closing curly braces line up and that the statements inside curly braces are indented more.

Archive Sketch

Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

Fix Encoding & Reload

Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.

Serial-Monitor

Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected port. This usually resets the board if the board supports reset over serial port opening.

Programmer

For selecting a hardware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.

Board

Select the board that you're using. See below for descriptions of the various boards.

Port

This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

Burn Bootloader

The items in this menu allow you to burn a bootloader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino board but is useful if you purchase a new AT mega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the Boards menu before burning the bootloader on the target board. This command also set the right fuses.

Features

- Modern, fully featured development environment
- Dual Mode, Classic Mode (identical to the Classic Arduino IDE) and Pro Mode (File System view)
- New Board Manager
- New Library Manager
- Board List
- Basic Auto-Completion (Arm targets only)
- Git Integration
- Serial Monitor
- Dark Mode.

5.2 MQTT

Overview of MQTT

MQTT (Message Queuing Telemetry Transport) is a lightweight, publish-subscribe network protocol designed for efficient communication between devices, particularly in constrained environments such as IoT (Internet of Things). It enables real-time data exchange with minimal bandwidth and power consumption.

Key Features of MQTT

- Lightweight and Efficient Ideal for low-bandwidth, high-latency, or unreliable networks.
- Publish/Subscribe Model Decouples message producers (publishers) from consumers (subscribers) using a broker.
- Asynchronous Communication Ensures fast and non-blocking messaging.
- Quality of Service (QoS) Levels
    - *QoS 0* At most once (fire and forget).
    - *QoS 1* At least once (acknowledged delivery).
    - *QoS 2* Exactly once (no duplication).
- Low Power Consumption Suitable for battery-powered or embedded systems.
- Small Code Footprint Ideal for microcontrollers and embedded devices.
- Security Supports TLS/SSL encryption and authentication mechanisms.

Working Principle of MQTT

1. Core Components of MQTT

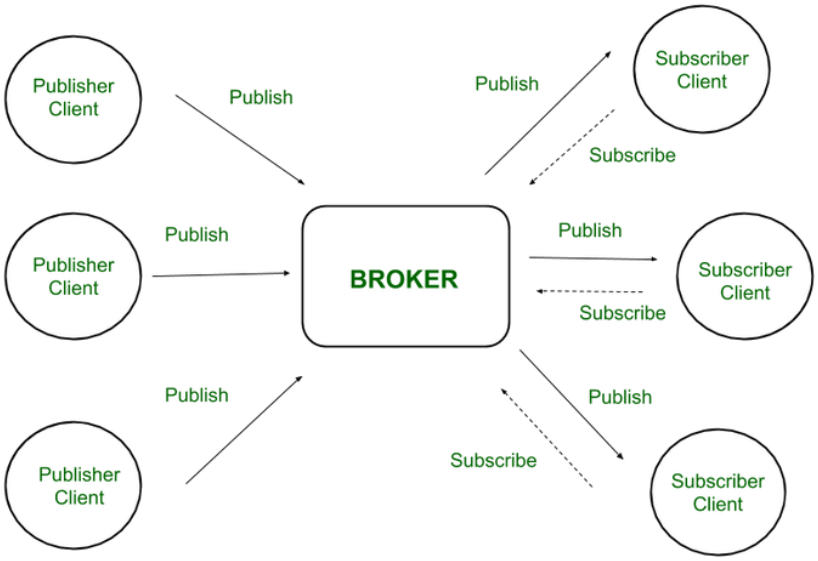

Figure 5.2: MQTT Publish/Subscribe Model

- **Client** Any device or application that connects to the MQTT broker. It can be a *publisher*, a *subscriber*, or both.
- **Broker** A central server that handles all message routing, filtering, and delivery between clients.
- **Topics** Structured strings (e.g., home/living room/temperature) used to label and route messages.
- **Messages** Data payloads (text, JSON, etc.) sent by publishers and received by subscribers.

2. Communication Model Publish/Subscribe

MQTT uses a publish/subscribe model instead of the traditional client-server model. Here's how it works

Step-by-Step Process

a. Client Connection

- Each client establishes a TCP/IP connection with the MQTT broker.
- Optionally, it can use TLS/SSL for secure communication.
- Upon connection, the client sends a CONNECT message with credentials (if required).

b. Topic Subscription

- A client subscribes to one or more topics using the SUBSCRIBE message.
- The broker stores these subscriptions and listens for messages on these topics.

c. Publishing a Message

- A client (publisher) sends a PUBLISH message to the broker, with
    - Topic name
    - Message payload (data)
    - Quality of Service (QoS) level

d. Message Routing by Broker

- The broker receives the message and checks which clients have subscribed to the topic.
- It then routes the message to each subscriber based on their QoS preference.

e. Receiving Messages

- Subscribers receive messages in real-time as long as they are connected.
- For higher QoS levels, the broker ensures message delivery and retransmission if needed.

3. Quality of Service (QoS) Levels

QoS determines the guarantee level of message delivery

- QoS 0 (At most once) No acknowledgment; message may be lost.
- QoS 1 (At least once) Message is guaranteed to arrive but may be duplicated.
- QoS 2 (Exactly once) Ensures one-time message delivery; most reliable and complex.

4. Retained Messages

- When a message is marked as "retained", the broker stores it.
- Any new subscriber to that topic immediately receives the last retained message.

5. Last Will and Testament (LWT)

- When a client connects, it can register a “Last Will” message.
- If the client disconnects unexpectedly, the broker sends this message to a specified topic, notifying other clients of the failure.

6. Keep Alive Mechanism

- MQTT clients send PINGREQ messages periodically to inform the broker they’re still connected.
- If no message is received within a certain time, the broker assumes the client is disconnected.

Applications of MQTT

- Smart homes and buildings
- Industrial automation (IoT)
- Remote monitoring and control
- Real-time analytics in logistics and transportation
- Healthcare and wearable devices

Advantages of MQTT

- It requires minimal resources since it is lightweight and efficient
- Supports bi-directional messaging between device and cloud
- Can scale to millions of connected devices
- Supports reliable message delivery through 3 QoS levels
- Works well over unreliable networks

CHAPTER 6

RESULT AND OUTPUT

6.1 PROJECT LAYOUT

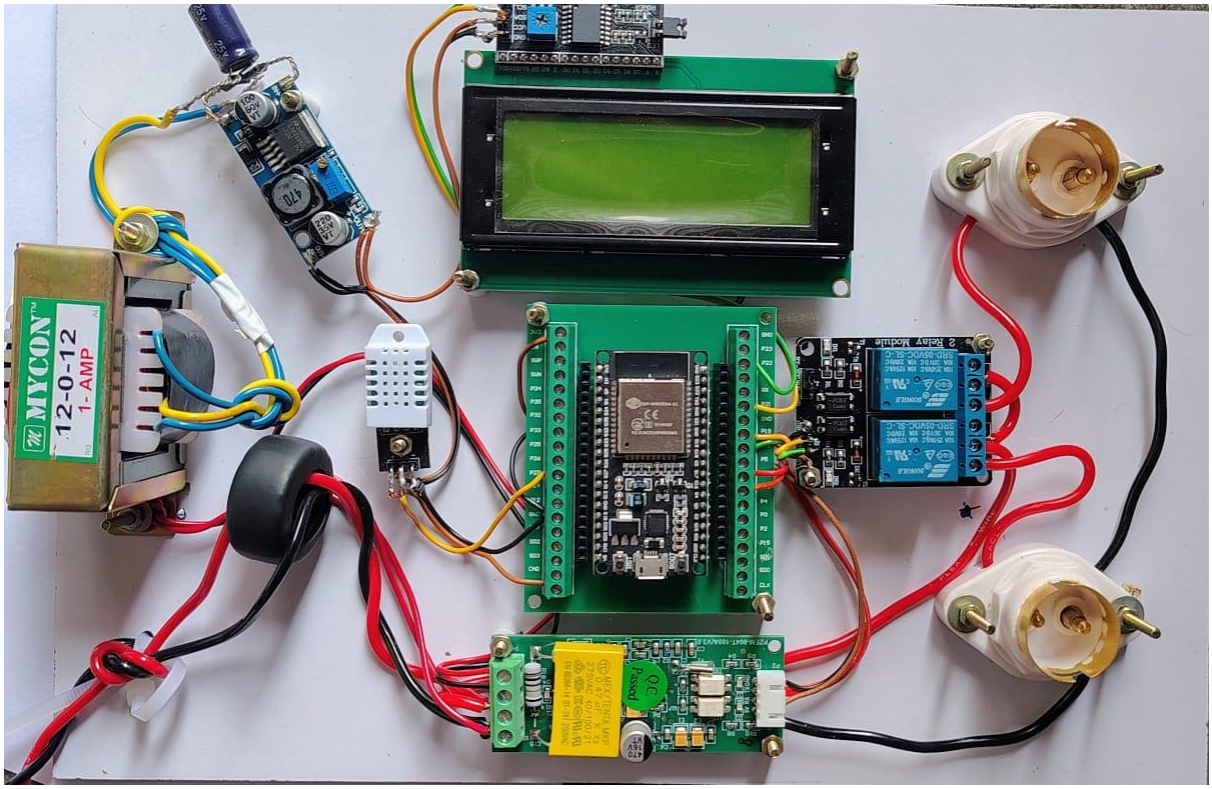

Figure 6.1: Layout

6.2 MQTT PANEL INTERFACE

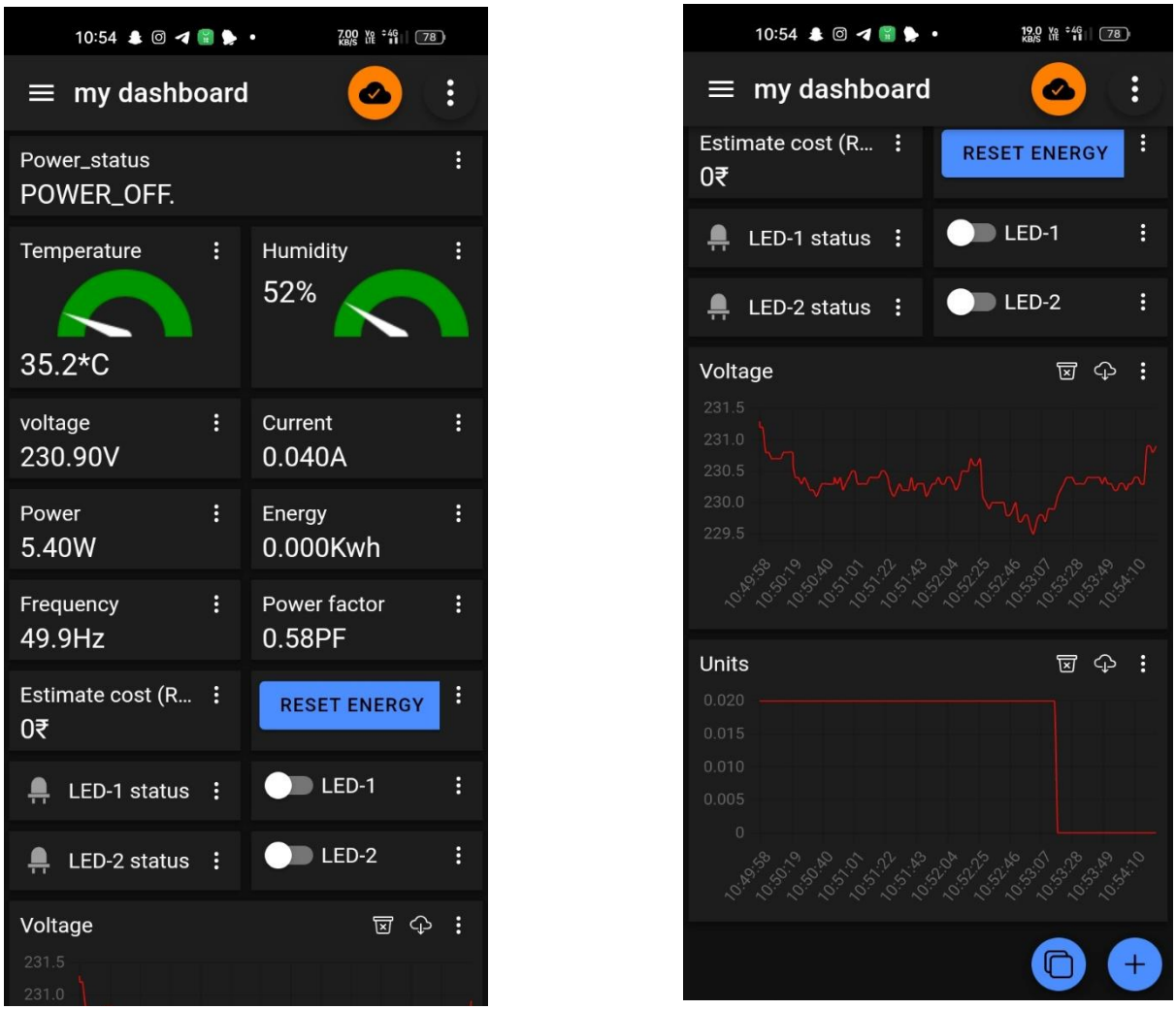

Figure 6.2: MQTT Panel Interface

6.3 WORKING

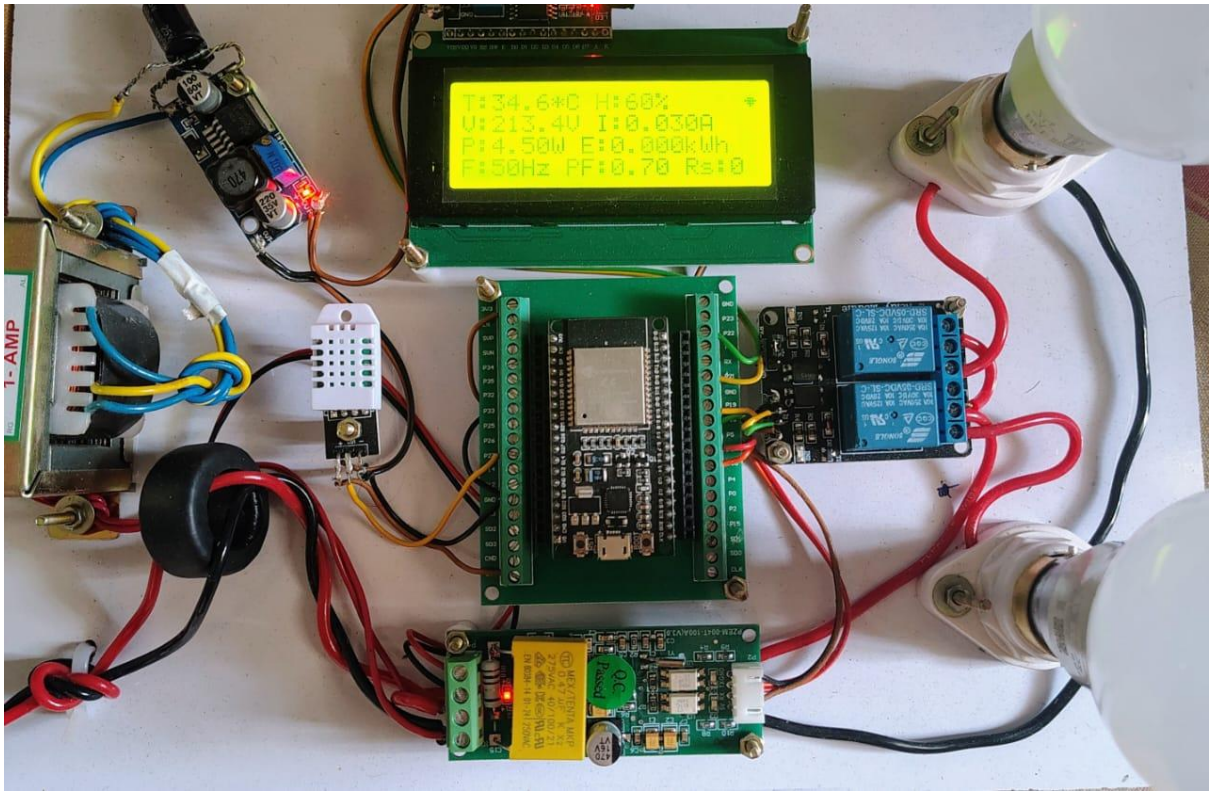

Figure 6.3: Working

CHAPTER 7

TESTING AND VALIDATION

The developed system was rigorously tested under standard home conditions to verify the functionality of environmental monitoring, energy consumption tracking, MQTT communication, and appliance control. Each component performed reliably, and the integrated system responded accurately to both sensor inputs and user commands.

7.1 Observed Outputs

1. Environmental Monitoring

- The DHT22 sensor accurately measured temperature and humidity.
- The readings were displayed in real-time on the 20x4 I^2C LCD.
- MQTT data packets were published to a predefined topic and displayed on the MQTT Panel mobile app.

Parameter	Observed Value Range
Temperature (°C)	24.3°C – 31.2°C
Humidity (%)	40% – 63%

2. Power Monitoring (PZEM-004T)

- Voltage, current, active power, and energy readings were successfully obtained and transmitted.

Parameter	Range Observed
Voltage (V)	224V – 238V
Current (A)	0.12A – 1.9A
Power (W)	25W – 340W

Parameter	Range Observed
Energy (kWh)	0.002 – 0.224 kWh

3. Relay Control via MQTT Panel

- Appliances connected to the 2-channel relay module could be controlled remotely.
- Switching ON/OFF commands from the mobile app showed instantaneous response.
- MQTT retained message support ensured that the state persisted after reconnection or reset.

7.3 Performance Metrics

Metric	Result
MQTT Data Latency	< 1 second
Sensor Reading Frequency	Every 2 seconds
Relay Response Time	Immediate (<0.5 seconds)
Power Consumption (System)	< 500mA during normal operation

7.4 Analysis

- The system maintained stable performance during continuous operation over 24 hours.
- MQTT communication was reliable over a local Wi-Fi network with minimal data loss.
- Power readings correlated well with manual measurements from a digital multimeter.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 Conclusion

The development of the Smart Weather and Power Consumption Monitoring System with Real-Time Control has successfully demonstrated the integration of environmental sensing, energy monitoring, and real-time control functionalities within a single, efficient, and cost-effective IoT platform. Leveraging the capabilities of the ESP32-WROOM microcontroller and the lightweight MQTT communication protocol, the system provides users with continuous insights into key environmental parameters (temperature and humidity) and electrical variables (voltage, current, power, and energy usage).

By utilizing the DHT22 and PZEM-004T sensors, alongside a 20x4 I2C LCD for local display and a smartphone-based MQTT Panel for remote access, the system ensures real-time data visibility and control. The inclusion of a 2-channel relay module enhances automation capabilities, enabling users to remotely operate electrical appliances based on data feedback or user input.

This system is particularly suited for smart home environments, where energy efficiency, remote monitoring, and intelligent control are increasingly critical. Its modular architecture and use of open protocols make it adaptable and scalable for future needs.

8.2 Future Enhancement

While the current system provides a strong foundation for smart home automation and monitoring, several enhancements can be considered to extend its functionality and user experience

1. **Cloud Integration & Data Logging**
   Integration with cloud platforms such as ThingSpeak, Firebase, or AWS IoT can allow long-term data storage, analysis, and visualization, enabling trends, anomaly detection, and energy audits.
2. **Mobile App Development**
   A custom mobile application with a more intuitive GUI could replace the MQTT Panel, offering features like historical data graphs, notifications, and scheduling of relay operations.
3. **AI-Based Automation**
   Implementing basic machine learning algorithms to predict user behavior or optimize appliance usage based on environmental conditions and energy consumption patterns.
4. **Solar Energy Monitoring**
   The system can be expanded to monitor solar panel output and battery levels, making it suitable for hybrid or off-grid energy systems.
5. **Voice Assistant Integration**
   Integration with voice platforms like Amazon Alexa or Google Assistant can enhance user accessibility and convenience.
6. **Security Improvements**
   Adding encryption for MQTT messages and implementing secure user authentication would improve system security, especially for internet-facing deployments.
7. **Multi-Node Support**
   Expanding the system to support multiple nodes (rooms or floors) connected via a central hub could make the setup scalable for larger buildings.

APPENDIX

```
// ----------------------- Libraries -----------------------
#include <DHT.h>
#include <WiFi.h>
#include <PZEM004Tv30.h>
#include <PubSubClient.h>
#include <LiquidCrystal_I2C.h>

// ----------------------- WiFi Credentials -----------------------
const char* ssid     = "your_ssid";
const char* password = "your_password";

// ----------------------- MQTT Settings -----------------------
const char* mqtt_server = "broker.emqx.io";
const int mqtt_port     = 1883;
const char* client_id   = "ESP32PowerClient";

// ----------------------- MQTAT Topics -----------------------
#define TEMP_TOPIC         "esp32/home584/temp"
#define HUMI_TOPIC         "esp32/home584/humi"
```

```
#define VOLTAGE_TOPIC      "esp32/home584/voltage"
#define CURRENT_TOPIC      "esp32/home584/current"
#define POWER_TOPIC        "esp32/home584/power"
#define ENERGY_TOPIC       "esp32/home584/energy"
#define FREQUENCY_TOPIC    "esp32/home584/frequency"
#define PF_TOPIC           "esp32/home584/pf"
#define COST_TOPIC         "esp32/home584/cost"
#define POWER_STATUS_TOPIC "esp32/home584/power_status"
#define LED1_CONTROL_TOPIC "esp32/home584/led1"
#define LED2_CONTROL_TOPIC "esp32/home584/led2"
#define LED1_STATUS_TOPIC  "esp32/home584/led1/status"
#define LED2_STATUS_TOPIC  "esp32/home584/led2/status"
#define RESET_ENERGY_TOPIC "esp32/home584/reset_energy"

// ----------------------- Pin Configuration -----------------------
#define LED1_PIN   19
#define LED2_PIN   18
#define DHTPIN     27
#define DHTTYPE    DHT22
#define PZEM_RX_PIN 16
```

```
#define PZEM_TX_PIN 17

// ----------------------- Global Variables -----------------------

bool led1State = LOW, led2State = LOW;

bool powerWasOn = true;


DHT dht(DHTPIN, DHTTYPE);

LiquidCrystal_I2C lcd(0x27, 20, 4);

PZEM004Tv30 pzem(Serial2, PZEM_RX_PIN, PZEM_TX_PIN);

WiFiClient espClient;

PubSubClient client(espClient);


// ----------------------- Custom Characters -----------------------

byte wifiConnectedChar[8] = {

  B00000, B00000, B01110, B11111,

  B01110, B00100, B00000, B00000

};


byte wifiDisconnectedChar[8] = {

  B00000, B00001, B01110, B11111,
```

```
  B01110, B10100, B00000, B00000
};

// ----------------------- Timing Variables -----------------------
unsigned long lastUpdate = 0;
const unsigned long updateInterval = 2000;
unsigned long lastWiFiCheck = 0;
const unsigned long wifiRetryInterval = 3000;

// ----------------------- Function Update WiFi Icon -----------------------
void updateWiFiSymbol() {
  lcd.setCursor(19, 0);
  lcd.write(WiFi.status() == WL_CONNECTED ? byte(0)  byte(1));
}

// ----------------------- Function Calculate Energy Cost -----------------------
float calculateCost(float energy) {
  float cost = 0;
  if (energy <= 500) {
    if (energy > 400) cost += (energy - 400) * 6.0, energy = 400;
```

```
    if (energy > 200) cost += (energy - 200) * 4.5, energy = 200;
    if (energy > 100) cost += (energy - 100) * 2.25, energy = 100;
  } else {
    if (energy > 1000) cost += (energy - 1000) * 11.0, energy = 1000;
    if (energy > 800)  cost += (energy - 800) * 10.0, energy = 800;
    if (energy > 600)  cost += (energy - 600) * 9.0,  energy = 600;
    if (energy > 500)  cost += (energy - 500) * 8.0,  energy = 500;
    if (energy > 400)  cost += (energy - 400) * 60.0, energy = 400;
    if (energy > 100)  cost += (energy - 100) * 4.5,  energy = 100;
  }
  return cost;
}

// ----------------------- MQTT Callback -----------------------
void callback(char* topic, byte* payload, unsigned int length) {
  String msg = "";
  for (uint8_t i = 0; i < length; i++) msg += (char)payload[i];

  if (String(topic) == LED1_CONTROL_TOPIC) {
    led1State = (msg == "1");
```

```
    digitalWrite(LED1_PIN, !led1State);
    client.publish(LED1_STATUS_TOPIC, led1State ? "1"  "0", true);
  }
  else if (String(topic) == LED2_CONTROL_TOPIC) {
    led2State = (msg == "1");
    digitalWrite(LED2_PIN, !led2State);
    client.publish(LED2_STATUS_TOPIC, led2State ? "1"  "0", true);
  }
  else if (String(topic) == RESET_ENERGY_TOPIC && msg == "1") {
    pzem.resetEnergy();
    Serial.println("Energy counter reset via MQTT");
    lcd.clear();
    lcd.setCursor(2, 1);
    lcd.print("Energy Reset Done!");
    delay(1500);
  }
}

// ------------------------ MQTT Reconnect ------------------------
void reconnect() {
```

```
if (WiFi.status() != WL_CONNECTED) return;

while (!client.connected()) {
  Serial.print("Connecting to MQTT...");
  lcd.clear();
  lcd.setCursor(0, 0); lcd.print("WiFi Connected!");
  lcd.setCursor(0, 1); lcd.print("IP"); lcd.print(WiFi.localIP());
  lcd.setCursor(0, 2); lcd.print("Online Mode");
  lcd.setCursor(0, 3); lcd.print("Connecting to MQTT..");

  updateWiFiSymbol();
  delay(2000);

  if (client.connect(client_id)) {
    Serial.println("connected.");
    client.subscribe(LED1_CONTROL_TOPIC);
    client.subscribe(LED2_CONTROL_TOPIC);
    client.subscribe(RESET_ENERGY_TOPIC);
  } else {
    Serial.println("MQTT failed, retrying in 5s");
```

```
    delay(5000);
   }

   lcd.clear();
  }
}

// ----------------------- Setup -----------------------
void setup() {
  Serial.begin(9600);
  dht.begin();
  lcd.begin();
  lcd.backlight();
  lcd.createChar(0, wifiConnectedChar);
  lcd.createChar(1, wifiDisconnectedChar);

  pinMode(LED1_PIN, OUTPUT);
  pinMode(LED2_PIN, OUTPUT);
  digitalWrite(LED1_PIN, HIGH);
  digitalWrite(LED2_PIN, HIGH);
```

```
  WiFi.begin(ssid, password);
  lcd.setCursor(0, 0); lcd.print("Smart Energy Meter");
  lcd.setCursor(0, 2); lcd.print("WiFi is");
  lcd.setCursor(0, 3); lcd.print("Connecting");
  updateWiFiSymbol();

  unsigned long wifiStart = millis();
  const unsigned long wifiTimeout = 30000;  // 30 seconds
  unsigned long lastDotUpdate = 0;
  int dotCount = 0;

  while (WiFi.status() != WL_CONNECTED && millis() - wifiStart < wifiTimeout) {
    int remainingTime = (wifiTimeout - (millis() - wifiStart)) / 1000;

    lcd.setCursor(0, 1);
    lcd.print("Offline Mode in   ");
    lcd.setCursor(17, 1);
    lcd.printf("%2ds", remainingTime);
    updateWiFiSymbol();
```

```
  if (millis() - lastDotUpdate >= 300) {
    lcd.setCursor(10 + dotCount, 3); lcd.print(".");
    dotCount++;
    if (dotCount >= 5) {
      for (int i = 0; i < 5; i++) lcd.setCursor(10 + i, 3), lcd.print(" ");
      dotCount = 0;
    }
    lastDotUpdate = millis();
  }

  if (WiFi.status() == WL_CONNECTED) break;
}

if (WiFi.status() != WL_CONNECTED) {
  lcd.clear();
  lcd.setCursor(0, 0); lcd.print("WiFi Failed!");
  lcd.setCursor(0, 1); lcd.print("Offline Mode");
  updateWiFiSymbol();
  delay(2500);
```

```
  }

  lcd.clear();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
}

// ----------------------- Loop -----------------------
void loop() {
  if (WiFi.status() != WL_CONNECTED && millis() - lastWiFiCheck >= wifiRetryInterval) {
    WiFi.begin(ssid, password);
    Serial.println("WiFi not connected. Attempting reconnection...");
    lastWiFiCheck = millis();
  }

  updateWiFiSymbol();
  if (!client.connected()) reconnect();
  client.loop();
  if (millis() - lastUpdate >= updateInterval) {
    lastUpdate = millis();
```

```
float v   = pzem.voltage();
float a   = pzem.current();
float w   = pzem.power();
float kWh = pzem.energy();
float hz  = pzem.frequency();
float pf  = pzem.pf();
float t   = dht.readTemperature();
float h   = dht.readHumidity();
float cost = calculateCost(kWh);
if (isnan(v) || isnan(a) || isnan(w) || isnan(kWh) || isnan(hz) || isnan(pf)) {
  if (powerWasOn) {
    client.publish(POWER_STATUS_TOPIC, "POWER_OFF", true);
    powerWasOn = false;
  }
  v = a = w = kWh = hz = pf = 0;
} else {
  if (!powerWasOn) {
    client.publish(POWER_STATUS_TOPIC, "POWER_ON", true);
    powerWasOn = true;
  }
```

```
    }
    // MQTT publish
    client.publish(TEMP_TOPIC,     String(t, 1).c_str(), true);
    client.publish(HUMI_TOPIC,     String(h, 0).c_str(), true);
    client.publish(VOLTAGE_TOPIC,  String(v, 2).c_str(), true);
    client.publish(CURRENT_TOPIC,  String(a, 3).c_str(), true);
    client.publish(POWER_TOPIC,    String(w, 2).c_str(), true);
    client.publish(ENERGY_TOPIC,   String(kWh, 3).c_str(), true);
    client.publish(FREQUENCY_TOPIC,String(hz, 1).c_str(), true);
    client.publish(PF_TOPIC,       String(pf, 2).c_str(), true);
    client.publish(COST_TOPIC,     String(cost, 0).c_str(), true);
    // LCD update
    lcd.setCursor(0, 0); lcd.printf("T%.1f*C H%.0f%%", t, h);
    lcd.setCursor(0, 1); lcd.printf("V%.1fV I%.3fA", v, a);
    lcd.setCursor(0, 2); lcd.printf("P%.2fW E%.3fkWh", w, kWh);
    lcd.setCursor(0, 3); lcd.printf("F%.0fHz PF%.2f Rs%.0f", hz, pf, cost);
  }
}
```

REFERENCES

[1] Banks, A., & Gupta, R. (2014). MQTT Version 3.1.1 Protocol Specification. OASIS Standard, pp. 1–84.

[2] Gupta, K., & Singh, S. (2019). Weather Monitoring using Raspberry Pi. In Proceedings of the International Conference on IoT and Smart Systems, pp. 22–26.

[3] Kale, S., Jadhav, A., & Patil, M. (2020). Home Automation using MQTT Protocol. International Journal of Computer Applications, Vol. 176(18), pp. 1–5.

[4] Khan, A., & Iqbal, M. (2021). Multi-parameter Smart Home Monitoring System. Journal of Electronics and Communication Engineering, Vol. 10(3), pp. 50–54.

[5] Kumar, P., & Reddy, R. (2020). Smart Energy Meter using PZEM Module. In Proceedings of the IEEE Conference on Power Systems, pp. 131–135.

[6] Mehta, P., & Rana, D. (2017). Zigbee-Based Energy Meter Monitoring System. International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol. 6(11), pp. 65–69.

[7] Sharma, A., & Patel, R. (2018). IoT-based Temperature and Humidity Monitoring System. International Journal of Engineering Research & Technology (IJERT), Vol. 7(4), pp. 1–4.

DHANALAKSHMI COLLEGE OF ENGINEERING

Approved by AICTE, New Delhi - Affiliated to Anna University, Chennai
NAAC Approved Institution | NBA Accredited Courses | An ISO Certified Institution

Associate Partners

Certificate of Presentation

This is to certify that Prof/Dr/Mr/Ms Bhavath.T, Dhanalakshmi College of Engineering participated and presented the paper titled Smart Weather and power Consumption Monitoring System with IoT Integration and Real-Time control using MATLAB and Node-RED in "International Conference on Innovation of Materials, Science and Engineering Technology (ICIMSET - 2025)", organised by the Department of Research and Development, in association with ICT Academy, MIT Square, London, IETE Student Forum and Dambi Dollo University, Ethiopia on 28th February 2025.

CONVENOR	PRINCIPAL	CHAIRMAN
Dr Sivakumar M	Dr Pradeep Kumar A R	Dr Ramamurthi V P

DHANALAKSHMI COLLEGE OF ENGINEERING

Approved by AICTE, New Delhi - Affiliated to Anna University, Chennai
NAAC Approved Institution | NBA Accredited Courses | An ISO Certified Institution

Associate Partners

Certificate of Presentation

This is to certify that Prof/Dr/Mr/Ms Hemath kumar N Nidhi R, Dhanalakshmi college of Engineering participated and presented the paper titled Smart Weather and Power Consumption Monitoring System with IoT Integration and Real-Time Control using MQTT and Node-RED in "International Conference on Innovation of Materials, Science and Engineering Technology (ICIMSET - 2025)", organised by the Department of Research and Development, in association with ICT Academy, MIT Square, London, IETE Student Forum and Dambi Dollo University, Ethiopia on 28th February 2025.

CONVENOR	PRINCIPAL	CHAIRMAN
Dr Sivakumar M	Dr Pradeep Kumar A R	Dr Ramamurthi V P

DHANALAKSHMI COLLEGE OF ENGINEERING

Approved by AICTE, New Delhi - Affiliated to Anna University, Chennai
NAAC Approved Institution | NBA Accredited Courses | An ISO Certified Institution

Associate Partners

Certificate of Presentation

This is to certify that Prof/Dr/Mr/Ms Yuvaraj. E, Dhanalakshmi college of Engineering, participated and presented the paper titled Smart Weather and Power Consumption Monitoring System with IoT Integration and Real-Time Control using MQTT and Node-RED in "International Conference on Innovation of Materials, Science and Engineering Technology (ICIMSET - 2025)", organised by the Department of Research and Development, in association with ICT Academy, MIT Square, London, IETE Student Forum and Dambi Dollo University, Ethiopia on 28th February 2025.

CONVENOR
Dr Sivakumar M

PRINCIPAL
Dr Pradeep Kumar A R

CHAIRMAN
Dr Ramamurthi V P

DHANALAKSHMI COLLEGE OF ENGINEERING

Approved by AICTE, New Delhi - Affiliated to Anna University, Chennai
NAAC Approved Institution | NBA Accredited Courses | An ISO Certified Institution

Associate Partners

Certificate of Presentation

This is to certify that Prof/Dr/Mr/Ms Esther Gandhimathi. S, Assistant Professor, Dhanalakshmi college of Engineering participated and presented the paper titled Smart Weather and Power Consumption Monitoring System with IoT Integration and Real-time Control Using MQTT and Node-RED in "International Conference on Innovation of Materials, Science and Engineering Technology (ICIMSET - 2025)", organised by the Department of Research and Development, in association with ICT Academy, MIT Square, London, IETE Student Forum and Dambi Dollo University, Ethiopia on 28th February 2025.

CONVENOR	PRINCIPAL	CHAIRMAN
Dr Dinesh S	Dr Pradeep Kumar A R	Dr Ramamurthi V P