# IoT-Based Smart Energy Meter

## SONARGAON UNIVERSITY (SU)

**Supervised By**

**Rudaina Tasnuva**

Lecturer
Deoertment Of EEE
Sonargaon University (SU)

**Submitted By**


| | |
|---|---|
| **Md. Nahid Hasan** | **EEE2201025019** |
| **Md. Abul Kalam Azad** | **EEE2201025021** |
| **Md. Tanbir Mollah** | **EEE1902017035** |
| **Md. Sunam Hossain** | **EEE2201025012** |
| **Md. Rion Ali** | **EEE2201025036** |

**Deperment of Electrical & Electrinics Engineering (EEE)**
Sonargaon University (SU) 147/I, Panthapath
Dhaka- 1215, Bangladesh


Date of Submission: September 2025

# Declaration

We certify that the research/project-based thesis titled, **"IoT-Based Smart Energy Meter"** is entirely our group work. All sources and knowledge for this paper that were obtained from other researchers are duly acknowledged through references. Materials used in this work, such as images, figures, tables, and citations, have been approved by our supervisor. We hereby declare that this thesis has not been previously submitted, either in whole or in part, for any other degree or publication.


**Md.Nahid Hasan**
EEE2201025019

**Md.Sunam Hossain**
EEE2201025012

**Md.Abul Kalam Azad**
EEE2201025021

**Md.Rion Ali**
EEE2201025036

**Md.Tanbir Molla**
EEE1902017035


**Under Supervision of**


**Rudaina Tasnuva**
Lecturer
**Depertment of EEE**
Sonargaon University (SU)

# Certification

This is to certify that this project entitled **"IoT-Based Smart Energy Meter"** is done by the following students under my direct supervision. This project work has been carried out by them in the laboratories of the **Department of Electrical and Electronics Engineering** under the Faculty of Engineering, **Sonargaon University (SU)** in partial fulfillment of the requirements for the degree of **Bachelor of Science in Electrical and Electronics Engineering**.

Supervised By


**Rudaina Tasnuva**
**Designation:** Lecturer
**Department of Electrical & Electronics Engineering (EEE)**
**Sonargaon University (SU)**

# Acknowledgement

The report titled as **"IoT-Based Smart Energy Meter"** has been prepared to fulfill the requirement of our practicum program. In the process of doing and preparing our practicum report, we would like to express our gratitude to several persons for their valuable help and continuous cooperation.

First, we would like to express our heartfelt gratitude to the University authority for permitting us to carry out this practicum. In particular, we are deeply thankful to our honorable teacher **Rudaina Tasnuva**, Lecturer,**Department of Electrical & Electronics Engineering (EEE), Sonargaon University**, for her valuable and patient guidance, sympathetic assistance, continuous cooperation, and contribution of innovative ideas. Her deep theoretical and hardware knowledge, along with her keen interest in this field, inspired us to complete this project. Her endless patience, scholarly supervision, continual encouragement, constructive criticism, and valuable advice — including reviewing multiple drafts and providing corrections at every stage — have made it possible to successfully complete this work.

We would also like to extend our sincere thanks to the respected Vice-Chancellor of Sonargaon University, **Professor Dr. Shamim Ara Hossain**, and to the Head of the **Department of Electrical & Electronics Engineering (EEE)**, **Assistant Professor Md. Rais Uddin Mollah**, for creating such an academic environment through which we gained the opportunity to acquire knowledge under the Bachelor of Science in Computer Science and Engineering program — knowledge that will be immensely helpful for our future careers.

Finally, we are truly grateful to all those from whom we received sincere cooperation, advice, and help in preparing this report.

# Abstract

The IoT-Based Smart Energy Meter is an important innovation for our country. In the modern age, electricity is essential for almost every activity in daily life. However, we face challenges such as power wastage, inaccurate billing, and inefficient use of resources. Without proper monitoring, users often consume more electricity than necessary, and service providers struggle to maintain fair and efficient distribution.

Our project addresses these issues by introducing an IoT-Based Smart Energy Meter that measures voltage, current, power, and total energy consumption in real time. Using the PZEM004T sensor with an ESP8266 Wi-Fi module, the system collects data and transmits it to the Blynk mobile application. This allows users to monitor their electricity usage from anywhere, receive usage statistics, and take corrective actions to reduce unnecessary consumption.

This solution increases billing accuracy, promotes energy savings, and encourages responsible usage. By enabling real-time data access and control, it supports more efficient power management and improves the quality of electrical services in our country.

# Table of Contents

**Declaration** ..... **ii**
**Certification** ..... **iii**
**Acknowledgement** ..... **iv**
**Abstract** ..... **v**
**Table of Contents** ..... **vi**
**List of Figures** ..... **ix**
**1. Introduction** ..... **1**
1.1 Introduction ..... 1
1.2 Problem Statement ..... 2
1.3 Background of the Study: ..... 2
1.4 Objective ..... 3
1.5 Methodology ..... 4
**2. Theory** ..... **5**
2.1 Overview of Traditional Energy Meters ..... 5
2.2 Importance of energy monitoring in smart grids ..... 5
2.3 Role of microcontrollers in IoT systems ..... 6
**3. STUDY FOR DESIGN AND DEVELOPMENT** ..... **7**
3.1 Introduction ..... 7
3.2 Multimeter ..... 7
3.3 Combination Pliers ..... 8
3.4 Star Screw driver ..... 8
3.5 Flat Screw driver ..... 9
3.6 Soldering Iron ..... 9
3.7 Jumper Wires ..... 9
3.8 Breadboard ..... 10
3.9 ESP8266 NodeMCU Module ..... 10
ESP8266 NodeMCU Details ..... 11
Applications of ESP8266 NodeMCU ..... 13
Why NodeMCU is used in this project ..... 13
3.10 PZEM004T V3 Energy Meter Module ..... 14

PZEM-004T V3 Details ..... 14
Applications of PZEM-004T V3 ..... 15
Why PZEM-004T V3 is used in this project ..... 15
3.11 20x4 Liquid Crystal Display (LCD) ..... 16
How the 20x4 LCD Works: ..... 16
3.12 I2C LCD Interface Module ..... 17
Key Technical Details of I²C Module ..... 17
Applications ..... 18
Why This Module is Important for the Project ..... 18
3.13 18650 Battery (Rechargeable Lithium Cell) ..... 18
3.14 18650 Charging Module (Type-C) ..... 19
Key Technical Details ..... 19
Why This Module is Used in the Project ..... 20
3.15 USB Output Module ..... 20
Key Technical Details ..... 20
Why This Module is Used in the Project ..... 21
3.17 Designed Flowchart ..... 22
3.16 Circuit Diagram ..... 22
3.18 Blynk App Integration (Mobile UI) ..... 23
Features of Blynk Integration in This Project ..... 23
Why Blynk App is Used in the Project ..... 25
3.19 Wi-Fi Network Configuration (IoT Setup) ..... 25
3.20 Arduino IDE Software ..... 25
Key features include: ..... 26
**4. RESULT AND DISCUSSION ..... 27**
4.1 Introduction ..... 27
4.2 Result ..... 27
4.3 System Functionality ..... 28
4.4 Performance Metrics ..... 28
4.5 Hardware Integration ..... 29
4.6 Software Logic ..... 29
4.7 User Interface ..... 30
4.8 Result Summary ..... 30

**5. CHALLENGES, IMPROVEMENTS, LIMITATION & FUTURE SCOPE...31**
5.1 Challenges and Improvements....31
5.2 Limitations of the Work....31
Hardware Limitations: ....31
Software Limitations:....31
System-Level Limitations:....32
5.3 Lack of Advanced Communication Protocols ....32
5.4 Safety Concerns ....32
5.5 Scalability ....32
5.6 Cost and Maintenance Limitations ....32
5.7 User-Related Limitations....33
5.8 Regulatory and Compliance Issues....33
5.9 Certification ....33
5.10 Future Scope ....33
**6. Conclusion ....35**
6.1 References....36
6.2 Appendix....38

# List of Figures

Figure 3. 1: Multimeter ..... 8
Figure 3. 2 : Combination Pliers ..... 8
Figure 3. 3 : Star Screw driver ..... 8
Figure 3. 4 : Flat Screw driver ..... 9
Figure 3. 5 : Soldering Iron ..... 9
Figure 3. 6 : Jumper Wires ..... 10
Figure 3. 7 : Breadboard ..... 10
Figure 3. 8 : ESP8266 NodeMCU Module ..... 11
Figure 3. 9 : PZEM004T V3 Energy Meter Module ..... 14
Figure 3. 10 : 20x4 Liquid Crystal Display (LCD) ..... 16
Figure 3. 11 : I2C LCD Interface Module ..... 17
Figure 3. 12 : 18650 Battery (Rechargeable Lithium Cell) ..... 18
Figure 3. 13 : 18650 Charging Module (Type-C) ..... 19
Figure 3. 14 : USB Output Module ..... 20
Figure 3. 15 : Designed Flowchart ..... 22
Figure 3. 16 : Circuit Diagram ..... 22
Figure 3. 17 : Blynk App Integration (Mobile UI) ..... 23
Figure 3. 18 : Real-Time Data Display ..... 23
Figure 3. 19 : Virtual Pins Usage ..... 24
Figure 3. 20: User Interface (UI) Design ..... 24
Figure 3. 21: Arduino IDE Software programe ..... 26

Figure 4. 1 : Result of the Project ..... 27

# 1. Introduction

## 1.1 Introduction

The outline of the project is to design and develop an IoT-based system capable of monitoring and displaying electrical parameters such as voltage, current, power, and total energy consumption in real time. This is achieved by using an ESP8266 (NodeMCU) microcontroller and the PZEM004T energy measurement sensor. The measured data is transmitted wirelessly through Wi-Fi to the Blynk mobile application, enabling remote monitoring and improved control over electricity usage.

In this project, the ESP8266 acts as the central controller, interfacing with the PZEM004T sensor to collect electrical data. The system also features a 20x4 Liquid Crystal Display (LCD) to show the readings locally, allowing both on-site and remote monitoring. The IoT connectivity ensures that users can access their energy usage statistics anytime and from anywhere via their smartphone.

The system is powered by a regulated supply. The incoming AC supply is converted to DC using an appropriate power adapter and regulated to +5V and +3.3V as required by the microcontroller and other components. This ensures stable and reliable operation of the device.

The need for automated, real-time monitoring of electricity consumption is growing rapidly. Traditional meters require manual reading and do not provide instant data, leading to inefficiencies and possible billing disputes. This IoT-based solution provides transparency, helps in identifying unusual consumption patterns, and promotes energy savings by enabling users to make informed decisions.

Earlier methods of tracking energy usage involved physical meter reading and manual record keeping. With the advancement of embedded systems and IoT technology, it is now possible to automate this process entirely, saving time, improving accuracy, and ensuring better energy management. This project serves as a working prototype for smart metering systems, with potential applications in both residential and industrial energy monitoring.

## 1.2 Problem Statement

The regular activities of people may get disrupted when electricity usage is not monitored properly or when billing is inaccurate. In our daily usage, the voltage supply is typically 230–240V AC. In our country, the demand for electrical power is very high, and as consumption continues to grow, efficient monitoring and management of electricity have become essential.

Currently, most electricity meters are analog or basic digital models, which only record total consumption and require manual reading by utility providers. This method does not provide real-time data to consumers, making it difficult to detect wastage, monitor usage patterns, or prevent overconsumption. As a result, electricity bills may be higher than expected, and resources may be wasted unnecessarily.

Without proper monitoring, specific areas may consume more electricity than planned, causing an imbalance in the distribution network. The lack of instant feedback to users also means that inefficient appliances or unusual power usage may go unnoticed. Moreover, the absence of remote monitoring capabilities prevents users from controlling their electricity usage while away from home or managing it more effectively.

Our IoT-Based Smart Energy Meter aims to solve these problems. By continuously measuring voltage, current, power, and energy consumption and transmitting this data to a mobile application, the system provides real-time visibility and control over electricity usage. This ensures accurate billing, reduces wastage, and promotes more efficient energy management.

## 1.3 Background of the Study:

Nowadays, the efficient use and monitoring of electricity is very important for people. Accurate and real-time energy usage information is needed in almost every sector of our lives — for home applications, industries, research centers, hospitals, and many other areas. With the increasing demand for energy and the limited nature of natural resources, it has become essential to adopt smarter and more efficient methods for managing electricity consumption.

Traditional energy metering systems only record total consumption over time and require manual reading, which is often slow, inconvenient, and prone to errors. These systems do not allow consumers to track their energy usage in real time, making it

difficult to prevent wastage or identify unusual consumption patterns. As a result, electricity bills may be inaccurate or unexpectedly high, and energy resources may not be used efficiently.

This project focuses on the design and development of an IoT-Based Smart Energy Meter that provides real-time monitoring and remote access to consumption data. By using the ESP8266 microcontroller and the PZEM004T energy measurement module, along with the Blynk mobile platform, the system ensures that users can view their electricity usage at any time and from anywhere. This enables better decision-making for reducing wastage, improving energy efficiency, and ensuring accurate billing.

The system increases reliability by providing continuous and updated consumption data. It is resilient to data loss by storing readings locally when internet connectivity is unavailable, and then syncing once reconnected. This makes it a modern, low-cost, and practical solution for enhancing the efficiency and transparency of electricity usage.

## 1.4 Objective

- To ensure accurate and real-time monitoring of electricity usage for households and industries using IoT technology.
- To provide remote access to energy consumption data through the Blynk mobile application, allowing monitoring from anywhere.
- To improve billing accuracy and transparency between users and utility providers.
- To promote energy efficiency by identifying wastage and encouraging responsible usage.
- To reduce human effort by automating the process of electricity monitoring and data collection.

## 1.5 Methodology

This IoT-Based Smart Energy Meter operates on the principle of continuous measurement, data acquisition, and wireless transmission of electrical parameters to a remote monitoring platform. The system is designed to measure voltage, current, power, and total energy consumption in real time and present this information both locally on an LCD display and remotely through the Blynk mobile application.

The hardware of the project consists of an ESP8266 (NodeMCU) microcontroller, a PZEM004T energy measurement module, a 20x4 LCD display, and supporting components such as resistors, connectors, and power regulation modules. The ESP8266 acts as the central control unit, collecting readings from the PZEM004T sensor and processing the data before transmitting it via Wi-Fi to the cloud-based Blynk platform.

The system is powered by a regulated DC supply. The AC input is converted to DC through an adapter or charging module, with appropriate voltage regulation for both the microcontroller and the sensor. The meter is capable of storing consumption data locally if internet connectivity is temporarily unavailable, and synchronizes the data once the connection is restored.

The main reason for choosing this project is the growing need for accurate, real-time, and easily accessible energy usage data for both households and industries. Many users face challenges with high electricity bills, wastage, and the inability to monitor consumption patterns. With this IoT-based approach, electricity usage can be tracked instantly, allowing corrective actions to be taken for energy saving and cost reduction.

The suitability of this system lies in its reliability, affordability, and ease of implementation. It is a modern solution that combines embedded systems with IoT technology to provide a transparent, user-friendly, and effective method for energy monitoring and management.

# 2. Theory

## 2.1 Overview of Traditional Energy Meters

An IoT-based energy monitoring system is an electronic apparatus that provides continuous, real-time measurement and reporting of electrical parameters (such as voltage, current, power and energy) from a consumer load while connected to the main supply. An IoT energy monitor differs from a conventional utility meter in that it delivers near-instantaneous visibility and remote access via networked communication rather than only accumulating consumption for later manual reading. The system's operation is persistent (continuous) rather than time-limited, providing live data streams and short-term historical logs that are sufficient for demand response, anomaly detection, and automated or remote control actions. Typical IoT meter deployments use a microcontroller (e.g., ESP8266) interfaced with a precision energy sensor (e.g., PZEM004T), a local display for on-site viewing, and a cloud/mobile platform (e.g., Blynk) to ensure reliable monitoring, timely alerts, and actionable reporting.

## 2.2 Importance of energy monitoring in smart grids

Energy monitoring in smart grids is a critical component for ensuring the efficient, reliable, and sustainable distribution of electricity. Unlike traditional power systems, which rely on manual readings and limited feedback, smart grids integrate advanced metering systems to provide real-time information about energy consumption. This enables both consumers and utility providers to make informed decisions that improve efficiency and reduce wastage.

For consumers, smart energy monitoring ensures accurate billing, prevents unexpected high electricity costs, and encourages responsible usage by providing real-time visibility of consumption patterns. By identifying peak usage times and inefficient appliances, households and industries can reduce unnecessary power usage and save costs.

For utility providers, energy monitoring helps balance supply and demand across the grid, prevents overloading, and minimizes the risk of blackouts. Real-time data collection also allows providers to detect faults, theft, or irregular consumption early, ensuring system stability and reducing financial losses.

In the context of renewable energy integration, monitoring plays an even greater role. As solar, wind, and other renewable sources fluctuate depending on weather and environment, smart grids require precise data to maintain a stable power supply. Energy monitoring systems make it possible to optimize renewable energy usage while ensuring that consumers receive continuous and reliable electricity.

Overall, energy monitoring in smart grids is not just a convenience but a necessity. It enhances efficiency, ensures sustainability, supports renewable energy integration, and promotes transparency between consumers and utility providers.

## 2.3 Role of microcontrollers in IoT systems

A microcontroller plays a crucial role in an IoT-based smart energy monitoring system by acting as the brain of the entire setup. It manages the collection, processing, and transmission of energy data, ensuring accurate and reliable performance. The microcontroller integrates with sensors, communication modules, and display units to deliver real-time monitoring and remote accessibility. Below are the key roles of a microcontroller in such a system:

- **Data Acquisition** – It receives input signals from the energy measurement module (such as the PZEM004T), which measures voltage, current, power, and energy consumption.
- **Data Processing** – The microcontroller processes raw sensor values into meaningful information that can be displayed locally or transmitted to the cloud.
- **Communication Management** – With built-in Wi-Fi (in ESP8266), it establishes IoT connectivity, sending data to platforms like Blynk for remote monitoring through smartphones or computers.
- **Local Display Control** – It drives the LCD screen, showing live energy parameters for users on-site.
- **Error Handling & Reliability** – It ensures continuous monitoring, even during network interruptions, by storing readings temporarily and updating them once the connection is restored.
- **System Integration** – It coordinates between different modules (sensors, displays, communication) to create a seamless and automated monitoring system.Thus, the microcontroller serves as the core unit that connects hardware and IoT platforms, enabling the smart energy meter to function as an efficient, reliable, and user-friendly system.

# 3. STUDY FOR DESIGN AND DEVELOPMENT

## 3.1 Introduction

This chapter highlights the construction and assembly of the IoT-Based Smart Energy Meter, focusing on both hardware and software integration. For the purchased components, the specific product types and their performance characteristics are reviewed. The primary emphasis is placed on the ESP8266 NodeMCU microcontroller, the PZEM004T V3.0 energy monitoring sensor, the 20x4 I2C LCD module, and the Blynk IoT platform. Additional supporting modules include the SoftwareSerial interface, voltage regulation circuit, and the NTPClient for time synchronization.

To achieve proper functionality, a detailed schematic diagram was designed to define the interconnection between all components. The hardware implementation ensures accurate energy parameter measurement, while the software handles data processing, real-time billing, and synchronization with the Blynk cloud server. Hardware assembly was carried out systematically using jumper wires and a breadboard for modular testing before final integration.

The system workflow combines sensor data acquisition, wireless data transmission, and user visualization via both the LCD and mobile application. The design was inspired by an extensive review of literature on IoT-based metering systems and tailored to the specific needs of residential energy monitoring in Bangladesh. The developed system demonstrates the practicality of synchronized hardware-software operation to provide efficient, real-time energy management.

## 3.2 Multimeter

An instrument designed to measure Electric Current, voltage, Frequency, Capacitance and Resistance.

- Two type of Multi meter available such as being:

    3.2.1 Digital Multi meter

    3.2.2 Analog Multi meter




**Figure *3. 1:* Multimeter**

## 3.3 Combination Pliers

Pliers are a hand tool used to hold objects firmly, possibly developed from tongs used to handle hot metal. They are also useful for bending and compressing a wide range of materials.




**Figure 3. 2 : Combination Pliers**

## 3.4 Star Screw driver

Electrical works a lot of importance to star screw driver. Without star screw driver it is impossible to connect a lot circuitry. Many sizes are available in the market for star screw driver. There are used depending on the work. There for a lot of importance to screw driver.



**Figure *3. 3* : Star Screw driver**

## 3.5 Flat Screw driver

Electrical works a lot of importance driver. Flat screw driver used in some electrical circuit. All goods that's required for the flat screw driver. Flat screw driver in a variety of market depending on the use for their work. There for a lot of importance of flat screw driver.




**Figure 3. 4 : Flat Screw driver**

## 3.6 Soldering Iron

A soldering iron is a tool used to melt solder for joining electronic components. It provides strong electrical and mechanical connections on circuit boards. In this project, the soldering iron was used to connect the ESP8266, PZEM004T sensor, and LCD module securely. This ensured stable performance and reduced loose wiring problems.



**Figure 3. 5 : Soldering Iron**

## 3.7 Jumper Wires

Jumper wires are small insulated wires used to make quick and flexible connections between components on a breadboard or circuit. They allow easy prototyping without soldering. In this project, jumper wires were used to connect the ESP8266 with the PZEM004T sensor, LCD module, and power supply, enabling modular testing and troubleshooting during the hardware assembly.

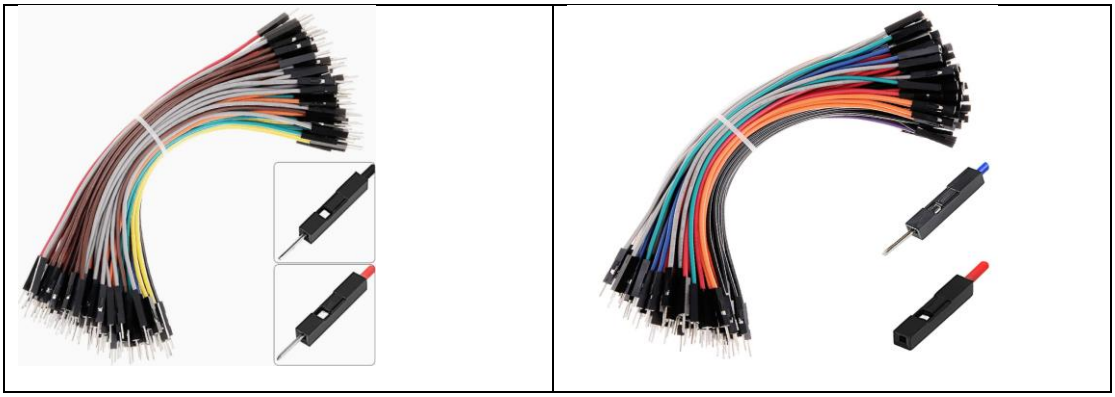


**Figure 3.6 : Jumper Wires**

## 3.8 Breadboard

A breadboard is a tool used for prototyping electronic circuits without the need for soldering. It allows components and jumper wires to be easily inserted and rearranged for testing. In this project, the breadboard was used to assemble the ESP8266, PZEM004T sensor, and LCD connections during the initial hardware implementation, making circuit testing and modifications simple.

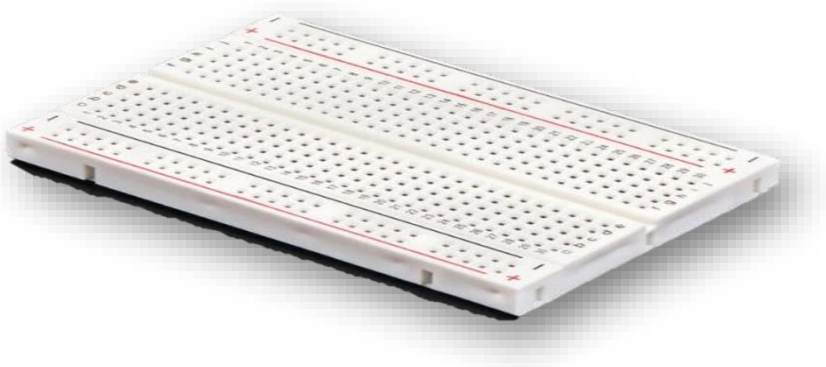


**Figure 3.7 : Breadboard**

## 3.9 ESP8266 NodeMCU Module

In a nutshell, the **ESP8266 NodeMCU** is an open-source development board based on the **ESP8266EX SoC**, a low-cost and powerful microcontroller with integrated Wi-Fi. It acts as the "brain" of many IoT projects, providing not only general-purpose I/O but

also built-in wireless networking. Unlike traditional microcontrollers such as the Arduino Uno, which require an external module for connectivity, the NodeMCU comes with **native Wi-Fi capability**, making it an ideal choice for IoT-based systems like smart energy meters.

The NodeMCU board can be programmed directly through a **Micro-USB cable** using popular platforms such as the **Arduino IDE**, **PlatformIO**, or the **NodeMCU firmware (Lua scripting language)**. It includes an on-board USB-to-Serial converter (CP2102 or CH340), a voltage regulator, and automatic reset circuitry, which makes it plug-and-play for developers.

For power, the NodeMCU can run on **USB 5V input** or an external **5V supply to the VIN pin**, regulated down to **3.3V logic**, which is the operating voltage of the ESP8266 chip. With its compact, breadboard-friendly design and wide range of features, the NodeMCU provides everything required for real-time energy monitoring, data logging, and cloud integration.

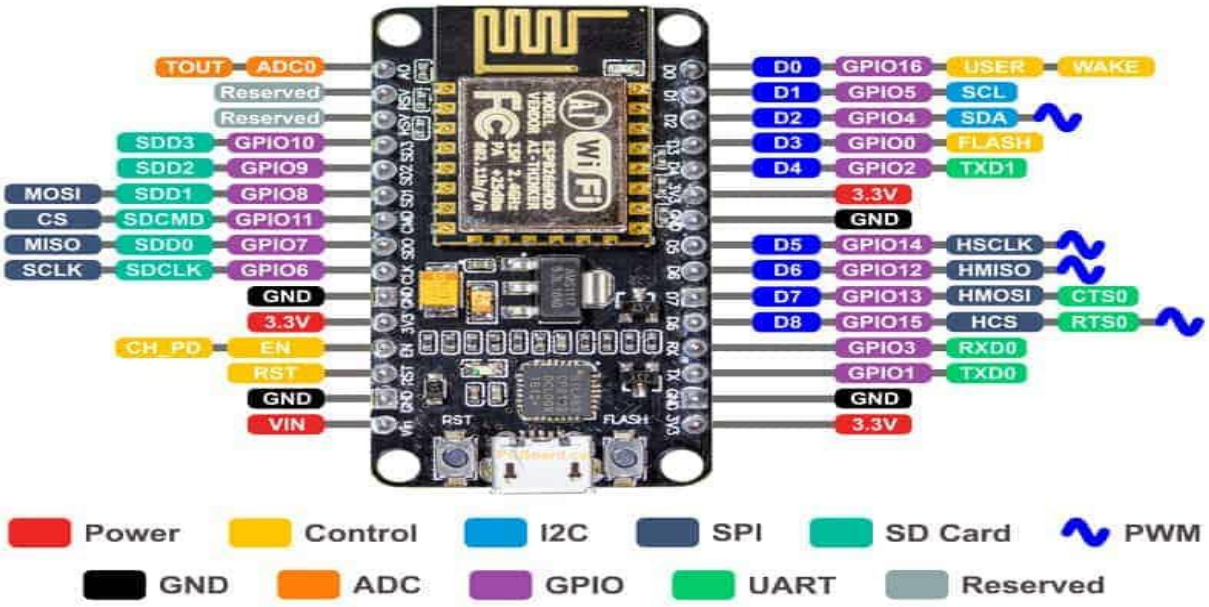



**Figure 3. 8 : ESP8266 NodeMCU Module**

### ESP8266 NodeMCU Details

#### 1. Microcontroller / Core

- Chipset: **ESP8266EX** (32-bit Tensilica L106 RISC processor).
- Clock Speed: **80 MHz (default)**, can be overclocked to **160 MHz**.
- Instruction Set: Harvard architecture, optimized for low-power IoT tasks.

**2. Memory**

- **Flash Memory:** Typically 4 MB (some variants up to 16 MB). Stores program code, web pages, OTA updates.
- **SRAM:** 64 KB instruction RAM, 96 KB data RAM (usable ~50 KB).
- **EEPROM (Emulated):** Stored in flash, available for saving small amounts of persistent data.

**3. Input / Output Pins**

- **Digital I/O Pins:** ~11–13 GPIOs available, configurable as input/output.
- **PWM Support:** Software-based PWM on most pins.
- **Analog Input:** 1 ADC channel (A0), 10-bit resolution (0–1 V range, often extended via onboard divider to 3.3 V).
- **Special Pins:**
  - GPIO0, GPIO2, GPIO15 → used for boot modes (must be carefully configured).
  - TX/RX pins for UART.
- **Maximum Current per Pin:** ~12 mA.

**4. Connectivity**

- Integrated **Wi-Fi (IEEE 802.11 b/g/n)**.
- Modes: **Station, Soft-AP, or both simultaneously**.
- Security: WPA/WPA2.
- Protocols supported: TCP, UDP, HTTP, MQTT.

**5. Power Supply**

- Operating Voltage: **3.3 V logic**.
- Input Power: **5 V via Micro-USB or VIN pin**.
- Onboard 3.3 V LDO regulator (capable of ~500 mA).
- Supports **Deep Sleep** (down to ~20 µA), ideal for battery-powered projects.

**6. Communication Interfaces**

- **UART (Serial):** For programming, debugging.
- **I²C (Software):** For sensors/displays (like the 20x4 LCD I²C backpack).
- **SPI:** For fast peripherals or external flash.
- **OneWire:** For temperature sensors (e.g., DS18B20).

**7. Programming & Software**

- **Arduino IDE** support with ESP8266 core installed.
- NodeMCU firmware (Lua-based scripting).
- Supports OTA (Over-The-Air) programming via Wi-Fi.
- Wide ecosystem of libraries for sensors, cloud integration (Blynk, MQTT, ThingSpeak).

**8. Indicators & Hardware Features**

- Onboard **blue LED** (often mapped to GPIO2 or GPIO16).
- **Reset button** and **Flash/Boot button**.
- **USB-to-Serial converter:** CP2102 or CH340G.
- Compact PCB with breadboard-compatible headers.

**Applications of ESP8266 NodeMCU**

- **IoT-based smart energy meters** (as in this project).
- Home automation (lights, fans, appliances).
- Real-time monitoring systems (temperature, humidity, environment).
- Data logging and wireless sensor networks.
- Wireless communication bridges and servers.
- Smart agriculture and industrial monitoring.

**Why NodeMCU is used in this project**

- **Built-in Wi-Fi:** Eliminates need for extra communication modules.
- **Low cost and small form factor:** Ideal for compact smart energy meters.
- **Flexible I/O:** Supports sensors, displays, and relays with ease.
- **Cloud/Mobile integration:** Works seamlessly with Blynk, MQTT, and custom dashboards.
- **Reliable for real-time monitoring:** Can log and transmit live energy data.

## 3.10 PZEM004T V3 Energy Meter Module

In a nutshell, the **PZEM-004T V3.0** is a compact and affordable energy monitoring module designed for measuring **AC electrical parameters** such as voltage, current, power, energy consumption, frequency, and power factor. It is widely used in smart energy monitoring systems because of its accuracy, low cost, and easy integration with microcontrollers like the ESP8266 NodeMCU.



**Figure 3. 9 : PZEM004T V3 Energy Meter Module**

The module works by using a **non-invasive current transformer (CT)** and a direct AC input for voltage measurement. It communicates digitally with a microcontroller via **UART (TX/RX serial communication)**, which makes it simple to retrieve data for further processing, storage, or cloud integration. In this project, the PZEM-004T V3 plays a central role in measuring household electricity consumption and transmitting the data to the NodeMCU for IoT-based monitoring through the Blynk platform.

**PZEM-004T V3 Details**

**1. Measurement Capabilities**

- **Voltage:** Measures AC voltage in the range of **80–260 V**.
- **Current:** Measures load current up to **100A** using the external CT sensor.
- **Active Power:** Calculates real-time power (in watts).
- **Energy (kWh):** Monitors total energy consumption over time.
- **Frequency:** Measures AC supply frequency (50/60 Hz).
- **Power Factor (PF):** Determines how efficiently power is being consumed.

**2. Hardware Features**

- External **current transformer (CT) clamp** for measuring high current safely.
- Compact PCB with easy terminal connections for AC input and load.
- Built-in isolation and protection circuits to improve safety and stability.
- Operates directly from the AC mains (no separate DC supply needed).

**3. Communication Interface**

- Uses **UART (Serial Communication)** with microcontrollers.
- Default baud rate: **9600 bps**.
- Supports multiple modules via unique addressing (Modbus RTU protocol).
- Can be interfaced with **ESP8266, ESP32, Arduino, Raspberry Pi**, etc.

**4. Electrical Specifications**

- Input Voltage Range: **80–260V AC**.
- Maximum Current: **100A (with CT sensor)**.
- Measurement Accuracy: ±0.5% (approx., varies with load).
- Power Supply: **From the measured AC line** (self-powered).

**5. Advantages**

- **Non-invasive measurement** using CT clamp (no need to cut wires).
- **Accurate and real-time monitoring** of all major AC parameters.
- **Compact size** makes it easy to integrate in IoT projects.
- **Digital communication** reduces noise and error compared to analog sensors.

**Applications of PZEM-004T V3**

- Smart energy meters (home and industrial).
- Load monitoring and analysis.
- Smart grid and IoT energy management systems.
- Preventive maintenance in electrical systems.
- Real-time power monitoring for solar inverters and UPS systems.

**Why PZEM-004T V3 is used in this project**

- **Comprehensive measurement:** Provides voltage, current, power, energy, and frequency in one module.
- **Safe measurement:** Uses CT sensor for current, avoiding direct high-current wiring.
- **Easy integration with ESP8266 NodeMCU** via UART.
- **Accurate and reliable readings** suitable for billing and monitoring.
- **Low cost and compact design,** making it ideal for IoT-based smart meters.

## 3.11 20x4 Liquid Crystal Display (LCD)

LCD stands for **Liquid Crystal Display**, a flat-panel display technology widely used in electronic devices for showing text, numbers, and symbols. The 20x4 LCD module can display **20 characters per line across 4 lines**, making it suitable for projects that need to show multiple parameters simultaneously. In this project, the LCD is used to present real-time electrical data such as voltage, current, power, energy, frequency, power factor, and calculated billing information.

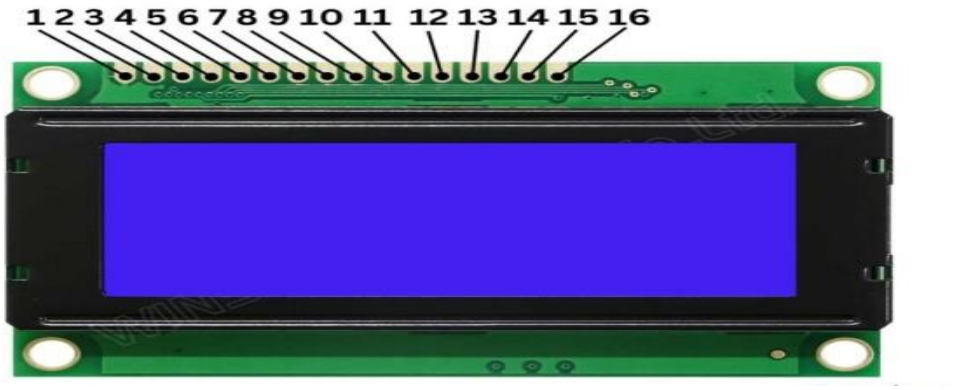



**Figure 3.10 : 20x4 Liquid Crystal Display (LCD)**

**How the 20x4 LCD Works:**

1. **Backlight:** The module uses an LED backlight to illuminate the screen, as the liquid crystals themselves do not emit light.
2. **Liquid Crystal Layer:** A thin layer of liquid crystals is placed between two transparent surfaces. When an electric signal is applied, the crystals align to control the flow of light.
3. **Polarizers:** Polarizing filters on both sides of the liquid crystal layer regulate the light, either allowing characters to appear or blocking them to create contrast.
4. **Character Matrix:** The 20x4 LCD uses a dot-matrix layout, where each character is formed from a 5x8 pixel grid, enabling it to display letters, numbers, and custom symbols.
5. **I²C Interface (in this project):** Instead of using multiple data pins, the LCD is connected to the NodeMCU through an I²C adapter, requiring only two pins (SDA and SCL) for communication, which simplifies wiring and saves GPIO pins.

## 3.12 I2C LCD Interface Module

In short, the I²C LCD interface module is a useful adapter that allows a microcontroller to control an LCD display using only two wires. Normally, a character LCD requires many digital pins for operation, but the I²C module simplifies this by converting serial data into parallel signals. This makes the circuit wiring easier, saves space, and leaves more GPIO pins available for other sensors and modules. In this project, the I²C interface is connected to the 20x4 LCD to show energy data such as voltage, current, power, energy consumption, frequency, power factor, and billing results.

The module is typically built around the **PCF8574 I/O expander chip**, which handles the conversion between I²C signals and the LCD's data/control pins. It also includes a small trimmer potentiometer to adjust the screen contrast and a jumper to enable or disable the backlight. Together, these features make the module very convenient for real-time monitoring systems like IoT-based smart meters.

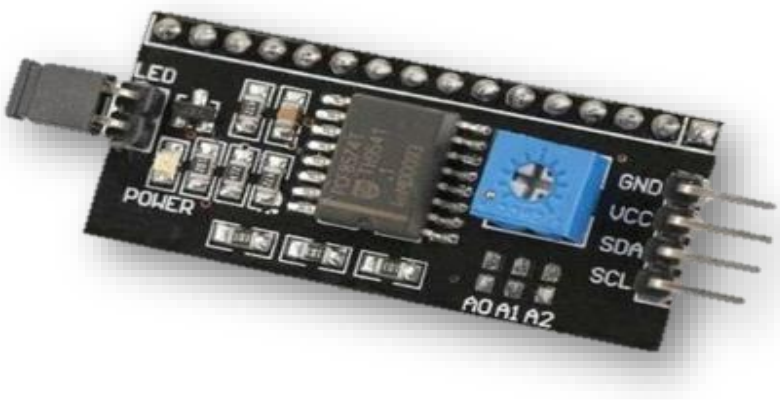



**Figure 3.11 : I2C LCD Interface Module**

**Key Technical Details of I²C Module**

**1. Communication**

- Works on the I²C protocol using two lines: **SDA (Serial Data)** and **SCL (Serial Clock)**.
- Multiple I²C devices can share the same bus with different addresses.

**2. Pin Description**

- **VCC:** 5V power input.
- **GND:** Ground.
- **SDA:** Connects to NodeMCU D2 (GPIO4).
- **SCL:** Connects to NodeMCU D1 (GPIO5).

**3. Device Addressing**

- The default I²C address is **0x27** or **0x3F** (depending on manufacturer).
- Address can be changed via solder jumpers on the module.

**4. Extra Features**

- Contrast adjustment using onboard potentiometer.
- Backlight control for better visibility.
- Compact design fits directly onto the back of the LCD module.

**Applications**

- Used to control 16x2, 20x4, and larger character LCDs in embedded projects.
- Reduces wiring effort and errors in circuit assembly.
- Suitable for microcontrollers with limited pins such as ESP8266 and Arduino Nano.

**Why This Module is Important for the Project**

The I²C module was chosen because it minimizes wiring complexity and allows the NodeMCU to manage both the LCD and other components like the PZEM004T sensor efficiently. By using just two pins, the system remains compact and reliable while still being able to display multiple real-time energy parameters to the user.

## 3.13 18650 Battery (Rechargeable Lithium Cell)

The 18650 battery is a rechargeable lithium-ion cell with a nominal voltage of 3.7V and high energy density. It is widely used in portable electronics and IoT projects due to its compact size, long lifespan, and reliability. In this project, it powers the NodeMCU and other modules, while a Type-C charging module ensures safe recharging and protection against overcharge or deep discharge.




**Figure 3. 12 : 18650 Battery (Rechargeable Lithium Cell)**

## 3.14 18650 Charging Module (Type-C)

In short, the 18650 Type-C charging module is a small power management board designed to safely charge and discharge **18650 lithium-ion batteries** using a modern **USB Type-C port**. It provides overcharge, over-discharge, and short-circuit protection, ensuring battery safety during continuous operation. The module typically uses a **TP4056 chip** for charging control and a DW01 protection IC to maintain battery health. In this project, the 18650 charging module can be used as a backup power supply for the ESP8266 NodeMCU and connected sensors. By integrating this module, the smart energy meter remains functional even during short power outages, ensuring uninterrupted data monitoring and transmission.




**Figure 3. 13 : 18650 Charging Module (Type-C)**

**Key Technical Details**

**1. Input & Output**

- **Input Voltage:** 5V DC via Type-C USB port.
- **Charging Current:** 1A (standard, some variants up to 2A).
- **Output Voltage:** 3.7V nominal (18650 cell), boosted to 5V using a step-up module if needed.

**2. Features**

- **Built-in protection:** Overcharge, over-discharge, overcurrent, and short-circuit.
- **LED Indicators:** Red for charging, Blue for full charge.
- **Compact design:** Easy integration with battery packs for portable IoT devices.

**3. Application in Project**

- Provides rechargeable battery backup for the NodeMCU.
- Ensures continuous energy monitoring during power failure.
- Makes the system portable for testing and demonstration.

**Why This Module is Used in the Project**

The 18650 Type-C charging module was chosen for its safety, convenience, and compatibility with rechargeable batteries. It allows the IoT-based smart energy meter to run independently of mains power when required, ensuring reliability and flexibility in real-world applications.

## 3.15 USB Output Module

the USB output module is a compact power supply board that delivers a regulated **5V DC output** through a USB port. It is widely used in embedded systems and IoT projects to power boards like the ESP8266 NodeMCU, sensors, and display modules. By providing stable voltage, it prevents malfunction or damage to sensitive electronic components.
In this project, the USB output module powers the NodeMCU, the PZEM004T energy meter sensor, and the 20x4 LCD with I²C interface. Its ability to deliver consistent 5V ensures that the smart energy meter can continuously monitor and transmit dat without interruption.



**Figure 3. 14 : USB Output Module**

**Key Technical Details**

**1. Input & Output**

- Input Voltage: Typically 100–240V AC (via an adapter/charger) or 12V DC (for DC-to-USB modules).
- Output Voltage: 5V regulated through the USB port.
- Output Current: Usually between 500mA to 2A, depending on the design.

**2. Features**

- Provides a stable and regulated 5V supply for microcontrollers.
- Includes protection against short circuits and overcurrent.
- Compact, plug-and-play design using a standard USB interface.

**3. Application in Project**

- Supplies 5V power to the ESP8266 NodeMCU via its Micro-USB port.
- Provides current for the PZEM004T sensor and the LCD module.
- Ensures smooth and continuous real-time data monitoring and cloud transmission.

**Why This Module is Used in the Project**

The USB output module was chosen because it offers a **simple, reliable, and safe way** to provide the required 5V supply for the IoT-based smart energy meter. It reduces circuit complexity, makes the system easier to assemble, and ensures stable performance for long-term operation

## 3.17 Designed Flowchart

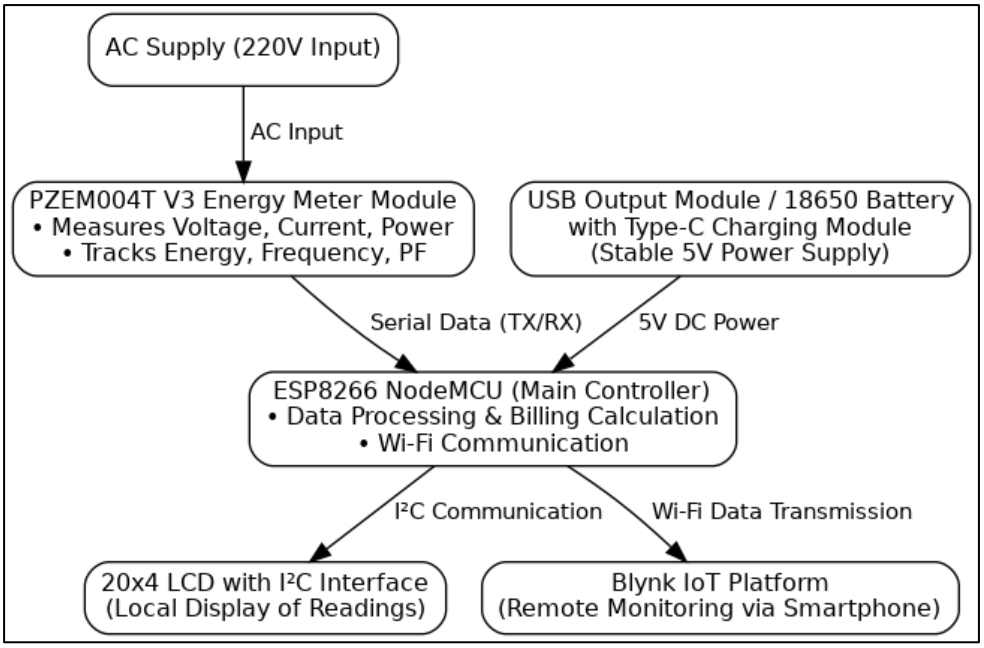



**Figure 3.15 : Designed Flowchart**

## 3.16 Circuit Diagram

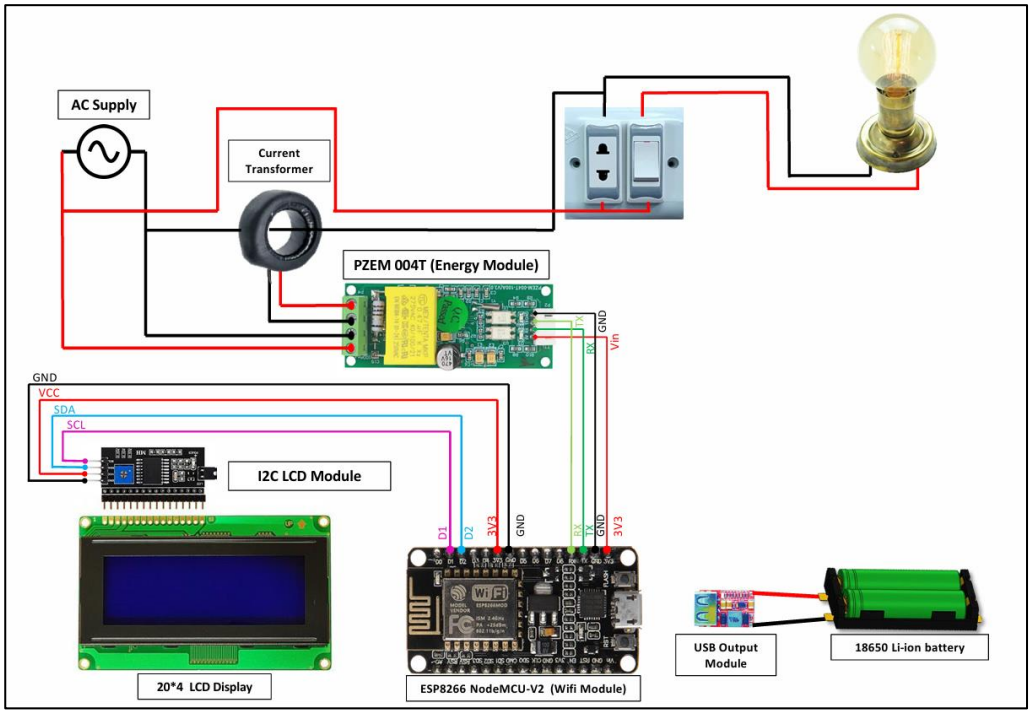



**Figure 3.16 : Circuit Diagram**

## 3.18 Blynk App Integration (Mobile UI)

The **Blynk mobile application** plays a crucial role in this project by providing a user-friendly interface for remote monitoring and control of the IoT-based smart energy meter. Blynk is an IoT platform that allows microcontrollers like the ESP8266 NodeMCU to connect with smartphones through the internet, enabling real-time visualization and interaction with system data. The app is available on both Android and iOS, making it widely accessible.

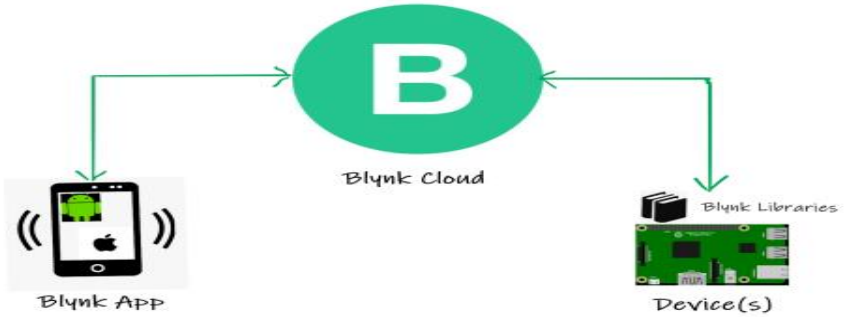



**Figure 3. 17 : Blynk App Integration (Mobile UI)**

In this project, the ESP8266 NodeMCU collects electrical parameters such as voltage, current, power, energy consumption, frequency, power factor, daily bill, and monthly bill from the PZEM004T module. These values are transmitted via **Wi-Fi** to the Blynk server, which then updates the mobile application in real time. Users can view these readings anywhere through their smartphone, ensuring convenience and continuous monitoring of energy usage.

**Features of Blynk Integration in This Project**

1. **Real-Time Data Display**
    - The measured parameters are displayed using **Value Display widgets** mapped to virtual pins (V1–V8).
    - Voltage, current, power, energy, frequency, and billing information are continuously updated.




**Figure 3. 18 : Real-Time Data Display**

2. **Virtual Pins Usage**
    - V1: Voltage
    - V2: Current
    - V3: Power
    - V4: Energy
    - V5: Frequency
    - V6: Power Factor
    - V7: Today's Bill
    - V8: Monthly Bill

| ID | Name | Pin | Color |
|---|---|---|---|
| 1 | Voltage | V1 | |
| 2 | Current | V2 | |
| 3 | Power | V3 | |
| 4 | Energy | V4 | |
| 5 | Frequency | V5 | |
| 6 | PowerFactor | V6 | |
| 7 | Today Bill | V7 | |
| 8 | Monthly Bill | V8 | |

**Figure 3.19 : Virtual Pins Usage**

3. **User Interface (UI) Design**
    - Simple and interactive dashboard created using drag-and-drop widgets.
    - Widgets such as Value Display, Gauge, and Graph provide clear visualization of data.
    - Labels and formatting make it easy for users to understand energy consumption trends.




**Figure 3. 20: User Interface (UI) Design**

**4.Cloud Connectivity**

- The NodeMCU communicates with the **Blynk Cloud Server** through Wi-Fi.
- Data is updated in real time and synchronized across multiple devices.

4. **Advantages**
   - Enables **remote monitoring** of energy usage.
   - Provides **data logging** for billing analysis.
   - Eliminates the need for manual meter readings.
   - Reduces wiring and hardware requirements by shifting visualization to the mobile app.

**Why Blynk App is Used in the Project**

The Blynk app was selected because it simplifies IoT integration with minimal coding requirements and provides an intuitive interface for end-users. Unlike traditional systems where monitoring is restricted to local displays, Blynk allows **global access** to energy data in real time. This makes the project more practical and aligned with modern smart metering systems, ensuring accessibility, transparency, and user convenience.

## 3.19 Wi-Fi Network Configuration (IoT Setup)

Wi-Fi network configuration is the core technology that enables IoT devices to connect to the internet and share data. In this project, the ESP8266 NodeMCU connects to a local Wi-Fi router using stored SSID and password. Once connected, it communicates with the Blynk cloud server through standard TCP/IP protocols.

## 3.20 Arduino IDE Software

The **Arduino IDE (Integrated Development Environment)** is an open-source software platform used to write, compile, and upload code to microcontrollers such as the ESP8266 NodeMCU. It supports a simplified version of C/C++ programming, making it user-friendly for developing IoT-based projects.
In this project, the Arduino IDE is used to program the NodeMCU for acquiring energy data from the **PZEM-004T sensor**, processing it, and transmitting it via Wi-Fi to the **Blynk cloud platform**. The IDE allows the addition of external libraries (e.g., ESP8266WiFi, Blynk, and PZEM004T libraries), which extend its functionality.

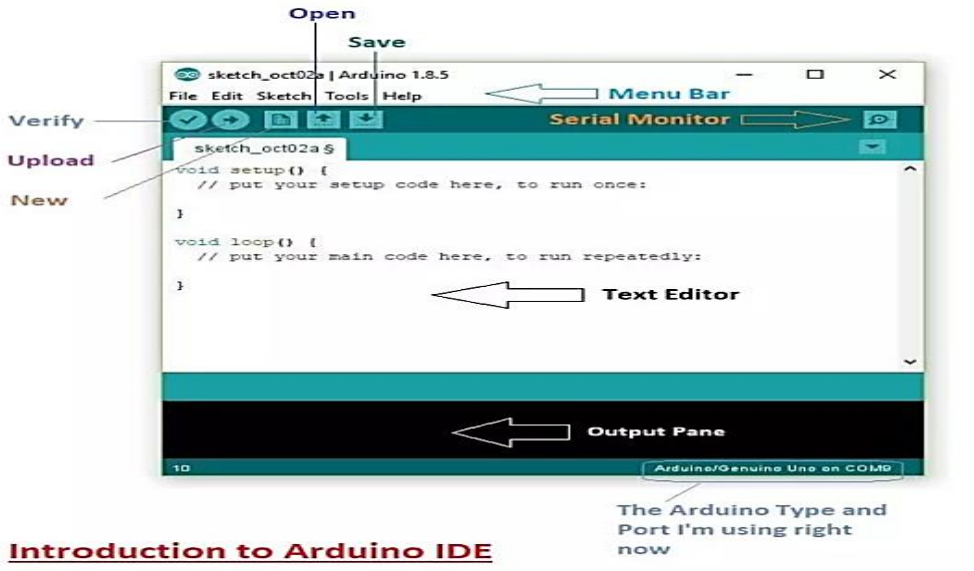



**Figure 3. 21: Arduino IDE Software program**

**Key features include:**

- Code editor with syntax highlighting.
- Built-in compiler for error checking.
- USB/serial communication for uploading programs.
- Cross-platform support (Windows, macOS, Linux).
- Community-driven library and board manager for hardware integration.

The Arduino IDE is essential for this project as it provides a **simple yet powerful environment** for developing and testing the IoT-based smart energy meter.

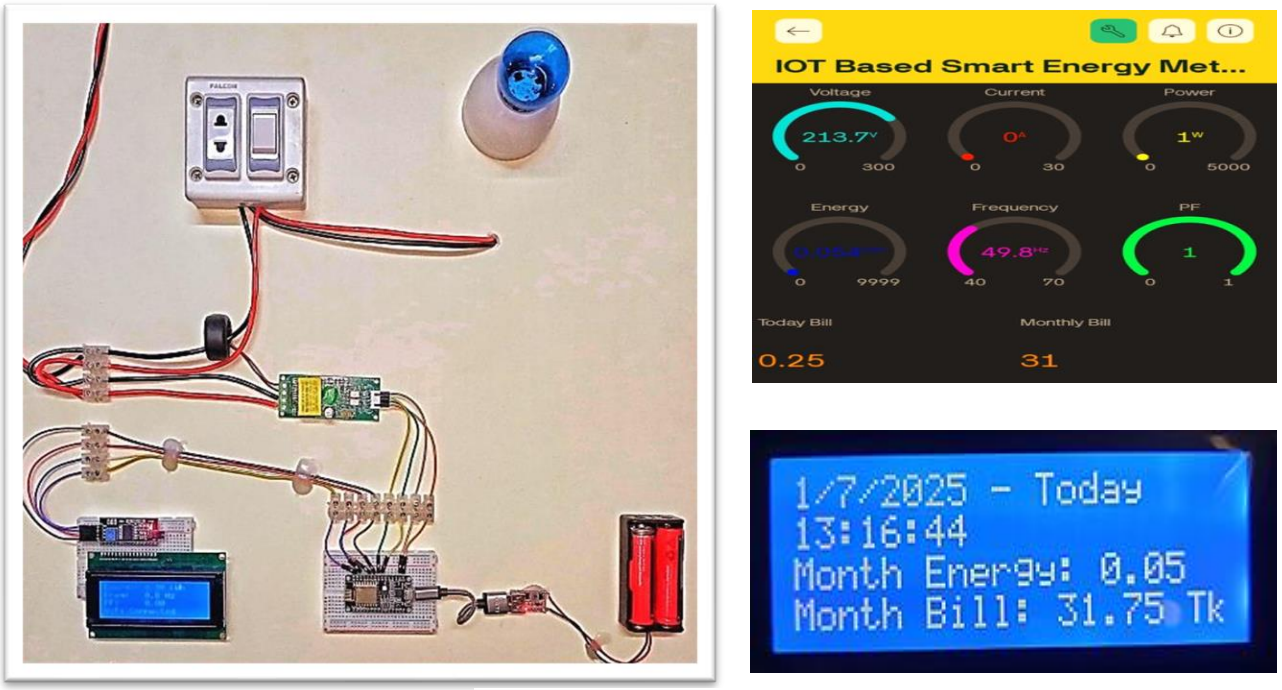
# 4. RESULT AND DISCUSSION

## 4.1 Introduction

In this chapter, we present the tangible results and observations obtained from the operation of our IoT-Based Smart Energy Meter. The performance of the integrated system is discussed, focusing on how effectively it measured, monitored, and transmitted real-time energy data. We also analyze the functionality of both hardware and software components and evaluate the accuracy, reliability, and responsiveness of the system under different conditions.

## 4.2 Result

In the Results section, we present the outcomes of our project's implementation, showcasing the successful integration of an **IoT-Based Smart Energy Meter** using the ESP8266 NodeMCU, PZEM-004T sensor module, and Blynk platform. The system was able to accurately measure key electrical parameters such as voltage, current, power, and energy consumption, and transmit this data in real time to the user's smartphone application. The results confirm that the hardware and software components worked together effectively to provide reliable and continuous monitoring of electricity usage.




***F*igure 4.1 : Result of the Project**

## 4.3 System Functionality

- **Real-Time Energy Monitoring:** The system successfully measures voltage, current, power, and energy consumption using the PZEM-004T sensor and continuously updates the data.
- **IoT Connectivity:** The ESP8266 NodeMCU transmits the measured data to the Blynk cloud platform through Wi-Fi, allowing users to access readings remotely via a smartphone application.
- **Data Logging and Visualization:** The Blynk interface provides real-time graphs, logs, and dashboards that help users visualize their energy usage patterns.
- **Offline Functionality:** When Wi-Fi is unavailable, the system continues monitoring locally, ensuring uninterrupted data collection until reconnection.
- **User-Friendly Interface:** The mobile app offers simple access to consumption data, enabling better decision-making for energy efficiency and cost management.

## 4.4 Performance Metrics

- **Response Time:** The system records and updates real-time voltage, current, power, and energy values with minimal delay (typically <1 second) to ensure accurate monitoring.
- **Data Transmission Speed:** The ESP8266 NodeMCU transmits readings to the Blynk server almost instantly, allowing users to access updated values on their mobile app in real time.
- **Accuracy:** The PZEM-004T sensor provides high precision in energy measurements, with error rates typically within ±1%.
- **Offline Data Handling:** In the event of Wi-Fi disconnection, the system continues monitoring locally and automatically syncs the data once connectivity is restored.

## 4.5 Hardware Integration

- **Sensor Accuracy:** The PZEM-004T V3 energy meter module provides precise voltage, current, power, and energy readings with minimal error, ensuring reliable monitoring.
- **Microcontroller Control:** The ESP8266 NodeMCU effectively processes sensor data, manages Wi-Fi connectivity, and communicates with the Blynk platform for remote monitoring.
- **Power Management:** The 18650 rechargeable battery and Type-C charging module ensure stable power supply to the system, allowing continuous operation during interruptions.
- **Display Functionality:** The 20x4 LCD module accurately displays real-time energy parameters locally, ensuring monitoring even without internet access.

## 4.6 Software Logic

**Algorithm Effectiveness:** The Arduino code implemented in the system ensures accurate real-time acquisition of electrical parameters such as voltage, current, power, and energy. The logic also calculates slab-wise billing according to the Bangladeshi tariff structure, including VAT and fixed charges, which aligns with practical billing scenarios. Data is periodically updated on both the LCD and Blynk IoT platform, ensuring synchronization and transparency. The time management algorithm, using NTP synchronization with a fallback on millis(), guarantees correct date and time even during temporary Wi-Fi outages, making the system highly reliable.

**Error Handling:** The software is designed to handle common edge cases gracefully. For instance, if Wi-Fi connectivity is lost, the system switches to offline mode and continues energy and billing calculations without interruption. Once the connection is restored, data synchronization resumes automatically. Similarly, sensor communication is monitored, and in case of abnormal readings (e.g., zero values or sensor errors), the display updates prevent misleading data. These mechanisms ensure the system remains operational and consistent even in adverse conditions.

## 4.7 User Interface

**LCD Display:** The system uses a 20x4 I2C LCD to display real-time parameters, including voltage, current, power, energy, and calculated billing. The display is organized into four phases, cycling every 5 seconds to present detailed system data without user intervention.

**Blynk Mobile Dashboard:** The IoT interface is implemented using the Blynk platform, enabling users to monitor electrical parameters and billing information remotely through a smartphone. Data such as voltage, current, power, energy usage, and daily/monthly bills are updated every 5 seconds, ensuring real-time synchronization.

## 4.8 Result Summary

The IoT-Based Smart Energy Meter successfully monitored voltage, current, power, and energy in real time, while accurately calculating slab-wise billing with VAT and fixed charges. Data was updated every 5 seconds on both the LCD and Blynk app, ensuring reliable user access. The system maintained functionality during Wi-Fi outages by using offline time tracking. Overall, the solution proved stable, cost-effective, and suitable for residential energy management. Future upgrades could include Google Sheets integration and AI-based usage predictions.

# 5. CHALLENGES, IMPROVEMENTS, LIMITATION & FUTURE SCOPE

## 5.1 Challenges and Improvements

- **Challenges Faced:** Wi-Fi instability caused occasional delays in data updates. Sensor calibration required precision to ensure accurate readings. LCD visibility and handling sensor errors were also challenging.
- **Improvements Suggested:** Add Google Sheets for data logging, include overcurrent alerts, improve UI, and upgrade to ESP32 for better performance. AI-based energy prediction and offline storage can enhance functionality.

## 5.2 Limitations of the Work

**Hardware Limitations:**

1. **Sensor Dependency:**
   The system relies on the PZEM004T module for accurate measurements. Any sensor malfunction or wiring issue can lead to incorrect readings.
2. **Voltage and Current Range:**
   The PZEM004T sensor supports limited voltage and current ratings (up to 260V AC and 100A with CT). Higher loads require additional sensors and protection circuits.
3. **Wi-Fi Dependency:**
   The NodeMCU ESP8266 operates on a 2.4 GHz Wi-Fi network. In areas with weak signals, remote monitoring through Blynk becomes inconsistent.
4. **Power Supply Reliability:**
   The system uses a 5V regulated power source. Voltage fluctuations or unstable power can affect performance and damage components.

**Software Limitations:**

1. **Processing Constraints:**
   The ESP8266 has limited memory and processing capability, restricting complex features like AI-based prediction or heavy data analytics.
2. **Offline Data Storage:**
   The current design lacks local storage for historical data when Wi-Fi is unavailable.
3. **Limited Error Handling:**
   Although basic error detection exists, advanced fault-tolerant algorithms are not implemented.

**System-Level Limitations:**

1. **Single-Phase Only:**
   The system is designed for single-phase loads and cannot monitor three-phase connections without major modifications.
2. **Security Risks:**
   Data is transmitted using standard Blynk protocols without advanced encryption, making it vulnerable to potential cyber threats.

## 5.3 Lack of Advanced Communication Protocols

3. The ESP8266 NodeMCU used in this project supports only basic communication methods such as UART, I²C, and Wi-Fi. However, it does not natively support advanced protocols like Modbus, ZigBee, or MQTT in hardware, which are often required for integration with industrial smart grids and large-scale IoT platforms. This limitation reduces the system's scalability and restricts its direct compatibility with advanced energy management infrastructures.

## 5.4 Safety Concerns

Switching between different power sources (USB module and 18650 battery backup) must be done carefully to prevent short circuits, voltage spikes, or reverse current flow. Proper isolation and protection circuits are necessary to ensure safe and reliable operation of the smart energy meter.

## 5.5 Scalability

## 5.6 Cost and Maintenance Limitations

- **Cost of Components:** High-quality sensors (PZEM004T), LCD displays, and power management modules can increase the overall project cost.
- **Maintenance Requirements:** Components such as the 18650 battery and connectors may require periodic maintenance or replacement to ensure reliability.
- **Dependency on External Libraries:** The ESP8266 NodeMCU relies heavily on external Arduino libraries for Wi-Fi, LCD, and sensor communication. Poorly maintained or incompatible libraries can cause software issues.

## 5.7 User-Related Limitations

- **Technical Expertise:** Designing, assembling, and programming the IoT-based smart energy meter requires knowledge of electronics, coding, and IoT protocols, which may be difficult for beginners.
- **User Interface:** While the Blynk app provides a mobile interface, additional effort may be needed to make the system more user-friendly, such as enhancing the LCD display or customizing the dashboard.

## 5.8 Regulatory and Compliance Issues

The IoT-based smart energy meter developed in this project is primarily designed for academic and small-scale residential applications. However, if such a system is to be deployed in real-world grid-connected environments, it must comply with national and international regulations on electrical safety, electromagnetic compatibility (EMC), and power quality standards. Failure to meet these requirements can pose safety hazards and prevent the system from being officially adopted by utility companies.

## 5.9 Certification

The system may require certification for use in certain applications, which can be time-consuming and costly.

## 5.10 Future Scope

The future scope of IoT-Based Smart Energy Metering is vast and promising, driven by rapid advancements in IoT, AI, renewable energy integration, and smart grid technologies. Below are some key areas of development and opportunities.

**1. Integration with Renewable Energy Sources**

- Smart meters can be integrated with solar panels, wind turbines, and biogas systems to track both consumption and generation in real-time.
- Net metering support will allow users to sell excess energy back to the grid, promoting decentralized energy ecosystems.

**2. Smart Grid and IoT Integration**

- **Real-Time Monitoring:** Continuous monitoring of voltage, current, power, and energy usage will enable dynamic load management and fault detection.
- **Remote Control:** With IoT and mobile apps (e.g., Blynk), consumers and utilities can monitor and control energy usage remotely.
- **Data Analytics:** Usage data can be analyzed to identify patterns, optimize billing, and forecast future energy needs.

**3. Energy Storage and Management**

- **Battery Management Systems (BMS):** Smart meters can work with batteries to optimize charging/discharging cycles, ensuring efficient energy storage.
- **Peak Load Shaving:** The system can reduce grid dependency by switching to stored energy during peak demand hours.

**4. Microgrid Development**

- IoT-based smart meters can become essential components of microgrids, enabling localized energy management, reliability, and resilience in rural or remote areas.

**5. Enhanced Automation and AI Integration**

- **Artificial Intelligence (AI):** Machine learning algorithms can predict consumption patterns, detect anomalies, and recommend energy-saving actions.
- **Automation:** Integration with home automation systems (e.g., smart appliances) will allow intelligent control of devices based on energy availability and cost.

**6. Improved User-Centric Features**

- Multi-language support, voice assistant integration, and gamification (e.g., rewards for saving energy) can increase user engagement.
- Accessibility features for elderly and rural populations can ensure inclusivity.

# 6. Conclusion

The “IoT-Based Smart Energy Meter” project successfully demonstrates an efficient and reliable system for monitoring and managing electrical energy consumption in real time. By utilizing the ESP8266 microcontroller, PZEM004T sensor, LCD display, and Blynk IoT platform, the system provides accurate measurement of voltage, current, power, and energy usage, along with slab-wise billing based on national tariff regulations.

Through IoT integration, users can remotely access their consumption data via smartphones, increasing awareness and enabling informed energy-saving decisions. The system also ensures billing transparency by calculating costs with VAT and fixed charges, while offering both local and remote display of data.

This project highlights the potential of IoT-based smart metering systems in enhancing energy efficiency, supporting renewable energy integration, and contributing to the development of smart grids. Despite challenges such as initial cost, network dependency, and regulatory barriers, the work provides a practical and innovative solution to the limitations of traditional energy meters.

Overall, this project not only achieves its stated objectives but also paves the way for future advancements in smart grid technologies, AI-driven analytics, and sustainable energy management. It serves as a strong foundation for building smarter, greener, and more consumer-centric energy ecosystems.

## 6.1 References


[1] International Energy Agency. (2023). *Global Energy Outlook*. Retrieved from https://www.iea.org

[2] Espressif Systems. (n.d.). *ESP8266 Technical Reference Manual*. Retrieved from https://www.espressif.com

[3] Blynk Inc. (2024). *Blynk IoT Platform Documentation*. Retrieved from https://docs.blynk.io

[4] NTP.org. (2023). *Network Time Protocol Documentation*. Retrieved from https://www.ntp.org

[5] Tech Explorations. (2022). *PZEM004T Energy Meter Usage Guide*. Retrieved from https://techexplorations.com

[6] Arduino. (n.d.). *SoftwareSerial Library Documentation*. Retrieved from https://www.arduino.cc

[7] OpenAI. (2025). *Assistance on Research and Writing via GPT-4.5*.

[8] Bangladesh Energy Regulatory Commission (BERC). (2023). *Electricity Tariff Guidelines*. Retrieved from http://www.berc.org.bd

[9] Ahmed, S., & Rahman, M. (2021). *IoT-Based Energy Monitoring and Billing Systems: A Review*. International Journal of Smart Grid and Clean Energy, 10(2), 135-144.

[10] Khan, M. T., & Hossain, M. S. (2022). *Integration of Smart Metering with Renewable Energy Sources in Bangladesh*. Journal of Electrical Engineering and Automation, 4(3), 51-60.

[11] IEEE Standards Association. (2020). *IEEE 2030.5-2020 - Smart Energy Profile Application Protocol*. Retrieved from https://standards.ieee.org

[12] International Electrotechnical Commission (IEC). (2019). *IEC 61850: Communication Networks and Systems for Power Utility Automation*. Retrieved from https://www.iec.ch

[13] Chowdhury, R., & Islam, T. (2020). *Challenges in Deploying IoT in Rural Energy Monitoring Systems in South Asia*. Proceedings of the IEEE Conference on Emerging Technologies and Applications, 89-94.

[14] Mahmud, K., Town, G. E., & Morsalin, S. (2020). *Smart Grid Demand Response: A Perspective on Bangladesh Energy Scenario*. Renewable and Sustainable Energy Reviews, 122, 109713.

[15] Hasan, M. M., & Alim, M. A. (2021). *Design and Implementation of a Smart Energy Meter Based on IoT and Arduino Platform*. Journal of Advanced Research in Embedded System, 5(2), 32–41.

[16] Alam, F., Mehmood, R., & Katib, I. (2022). *Smart Grid Enabled Smart Cities: Opportunities and Challenges*. Sustainable Cities and Society, 74, 103144.

[17] World Bank. (2023). *Electricity Access in South Asia: Progress and Policies*. Retrieved from https://www.worldbank.org

[18] Islam, M. S., & Sarker, A. (2020). *A Review on Smart Metering Infrastructure for Bangladesh*. Bangladesh Journal of Engineering and Technology, 20(1), 67–75.

## 6.2 Appendix

```
//========= BLYNK =============
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPL6QMbGnoAi"
#define BLYNK_TEMPLATE_NAME "IOT Based Smart Energy Meter"
#define BLYNK_AUTH_TOKEN "zcWHAHN29cQ4ErRzqLJgnkuY-FM8Ci5t"

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <LiquidCrystal_I2C.h>
#include <PZEM004Tv30.h>
#include <SoftwareSerial.h>

// ========== WiFi ==========
char ssid[] = "realme C25s";
char pass[] = "22334455";

// ========== LCD ==========
LiquidCrystal_I2C lcd(0x27, 20, 4);

// ========== PZEM ==========
SoftwareSerial pzemSerial(D5, D6);
PZEM004Tv30 pzem(pzemSerial);

// ========== Time ==========
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", 21600, 60000);  // UTC+6
(Bangladesh)

// ========== Variables ==========
unsigned long previousMillis = 0;
const unsigned long interval = 5000;

bool wifiConnected = false;
unsigned long wifiLostEpoch = 0;
unsigned long wifiLostMillis = 0;
unsigned long displayCycleStart = 0;
int displayPhase = 0;

BlynkTimer timer;

// ========== Slab-wise Bill Calculation ==========
float calculateBill(float unit, bool includeChargeAndVat = true) {
  float bill = 0;
  if (unit <= 50) bill = unit * 4.63;
  else if (unit <= 75) bill = 50 * 4.63 + (unit - 50) * 5.76;
```

```
  else if (unit <= 200) bill = 50 * 4.63 + 25 * 5.76 + (unit - 75) *
7.20;
  else if (unit <= 300) bill = 50 * 4.63 + 25 * 5.76 + 125 * 7.20 +
(unit - 200) * 7.21;
  else if (unit <= 400) bill = 50 * 4.63 + 25 * 5.76 + 125 * 7.20 + 100
* 7.21 + (unit - 300) * 8.26;
  else if (unit <= 600) bill = 50 * 4.63 + 25 * 5.76 + 125 * 7.20 + 100
* 7.21 + 100 * 8.26 + (unit - 400) * 12.67;
  else bill = 50 * 4.63 + 25 * 5.76 + 125 * 7.20 + 100 * 7.21 + 100 *
8.26 + 200 * 12.67 + (unit - 600) * 14.61;

  if (includeChargeAndVat) {
    bill += 30;           // Meter charge যোগ হবে শুধু এখানে
    bill += bill * 0.05;  // ৫% VAT যোগ হবে শুধু এখানে
  }
  return bill;
}

// ========== Setup ==========
void setup() {
  Serial.begin(115200);
  pzemSerial.begin(9600);

  lcd.init(); lcd.backlight(); lcd.clear();
  lcd.print("Connecting WiFi...");
  WiFi.begin(ssid, pass);

  int retry = 0;
  while (WiFi.status() != WL_CONNECTED && retry < 20) {
    delay(500); lcd.print("."); retry++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    wifiConnected = true;
    lcd.clear(); lcd.print("WiFi Connected"); delay(1500);
    timeClient.begin(); timeClient.update();
    wifiLostEpoch = timeClient.getEpochTime();
    wifiLostMillis = millis();
  } else {
    wifiConnected = false;
    lcd.clear(); lcd.print("WiFi Not Connected"); delay(1500);
    wifiLostEpoch = 0;
    wifiLostMillis = millis();
  }

  lcd.clear();
  displayCycleStart = millis();
  displayPhase = 0;
```

```
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  timer.setInterval(5000L, sendToBlynk);
}

// ========== Loop ==========
void loop() {
  Blynk.run();
  timer.run();

  wl_status_t currentStatus = WiFi.status();
  if (currentStatus == WL_CONNECTED && !wifiConnected) {
    wifiConnected = true;
    lcd.clear(); lcd.print("WiFi Reconnected"); delay(1000);
lcd.clear();
    timeClient.begin(); timeClient.update();
    wifiLostEpoch = timeClient.getEpochTime();
    wifiLostMillis = millis();
  } else if (currentStatus != WL_CONNECTED && wifiConnected) {
    wifiConnected = false;
    lcd.clear(); lcd.print("WiFi Lost"); delay(1000); lcd.clear();
    wifiLostMillis = millis();
  }

  if (wifiConnected) timeClient.update();
  unsigned long epoch = getCurrentEpoch();

  float voltage = pzem.voltage();
  float current = pzem.current();
  float power = pzem.power();
  float energy = pzem.energy();
  float freq = pzem.frequency();
  float pf = pzem.pf();

  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    displayPhase = ((currentMillis - displayCycleStart) / interval) %
4;

    switch (displayPhase) {
      case 0: displayVoltageCurrentPower(voltage, current, power);
break;
      case 1: displayEnergyFreqPFWiFi(energy, freq, pf, wifiConnected);
break;
      case 2: displayTodayInfo(epoch, energy); break;
      case 3: displayMonthInfo(epoch, energy); break;
    }
```

```
  }
}

// ========== Blynk ==========
void sendToBlynk() {
  float v = pzem.voltage();
  float c = pzem.current();
  float p = pzem.power();
  float e = pzem.energy();
  float f = pzem.frequency();
  float pf = pzem.pf();

  float todayBill = calculateBill(e, false);  // Today bill without
charge & VAT
  float monthBill = calculateBill(e, true);   // Month bill with charge
& VAT

  Blynk.virtualWrite(V1, v);
  Blynk.virtualWrite(V2, c);
  Blynk.virtualWrite(V3, p);
  Blynk.virtualWrite(V4, e);
  Blynk.virtualWrite(V5, f);
  Blynk.virtualWrite(V6, pf);
  Blynk.virtualWrite(V7, todayBill);
  Blynk.virtualWrite(V8, monthBill);
}

// ========== Display Functions ==========
void displayVoltageCurrentPower(float v, float c, float p) {
  lcd.clear();
  lcd.setCursor(0, 0); lcd.printf("Voltage: %.1f V", v);
  lcd.setCursor(0, 1); lcd.printf("Current: %d mA", int(c * 1000));
  lcd.setCursor(0, 2); lcd.printf("Current: %.3f A", c);
  lcd.setCursor(0, 3); lcd.printf("Power:   %d W", int(p));
}

void displayEnergyFreqPFWiFi(float e, float f, float pf, bool wifi) {
  lcd.clear();
  lcd.setCursor(0, 0); lcd.printf("Energy: %.2f kWh", e);
  lcd.setCursor(0, 1); lcd.printf("Freq:   %.1f Hz", f);
  lcd.setCursor(0, 2); lcd.printf("PF:     %.2f", pf);
  lcd.setCursor(0, 3); lcd.print(wifi ? "WiFi Connected" : "WiFi Not
Connected");
}

void displayTodayInfo(unsigned long epoch, float e) {
  lcd.clear();
```

```
  lcd.setCursor(0, 0); lcd.print("Today ");
lcd.print(getFormattedDate(epoch));
  lcd.setCursor(0, 1); lcd.print("Time: ");
lcd.print(getFormattedTime(epoch));
  lcd.setCursor(0, 2); lcd.printf("Used Energy: %.2f", e);
  lcd.setCursor(0, 3); lcd.printf("Today's Bill: %.2f Tk",
calculateBill(e, false));  // no charge/VAT
}

void displayMonthInfo(unsigned long epoch, float e) {
  lcd.clear();
  lcd.setCursor(0, 0); lcd.printf("1/7/%d - Today",
yearFromEpoch(epoch));
  lcd.setCursor(0, 1); lcd.print(getFormattedTime(epoch));
  lcd.setCursor(0, 2); lcd.printf("Month Energy: %.2f", e);
  lcd.setCursor(0, 3); lcd.printf("Month Bill: %.2f Tk",
calculateBill(e, true));   // charge/VAT added
}

// ========== Time Helper Functions ==========
unsigned long getCurrentEpoch() {
  return wifiConnected ? timeClient.getEpochTime() : wifiLostEpoch +
(millis() - wifiLostMillis) / 1000;
}

int yearFromEpoch(unsigned long epoch) {
  return 1970 + epoch / 31556926;
}

String getFormattedTime(unsigned long epoch) {
  unsigned long t = epoch % 86400;
  int h = t / 3600, m = (t % 3600) / 60, s = t % 60;
  char buf[9]; sprintf(buf, "%02d:%02d:%02d", h, m, s);
  return String(buf);
}

bool isLeapYear(int y) {
  return ((y % 4 == 0 && y % 100 != 0) || (y % 400 == 0));
}

void epochToDate(unsigned long epoch, int &y, int &m, int &d) {
  const int daysInMonth[] = {31,28,31,30,31,30,31,31,30,31,30,31};
  unsigned long days = epoch / 86400UL;
  y = 1970;
  while (true) {
    int dy = isLeapYear(y) ? 366 : 365;
    if (days >= dy) { days -= dy; y++; }
    else break;
```

```
  }
  m = 0;
  while (true) {
    int dm = daysInMonth[m];
    if (m == 1 && isLeapYear(y)) dm++;
    if (days >= dm) { days -= dm; m++; }
    else break;
  }
  d = days + 1;
}

String getFormattedDate(unsigned long epoch) {
  int y, m, d;
  epochToDate(epoch, y, m, d);
  char buf[11]; sprintf(buf, "%02d/%02d/%04d", d, m + 1, y);
  return String(buf);
}
```