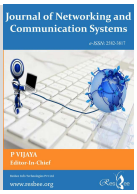
Vol.9 Issue 4 Oct 2026

Journal of Networking and Communication Systems

Received 2 December, Revised 16 October, Accepted 29 October



Design And Construction of an Iot-Based Energy Metering System

Ibrahim Muazzam Ibrahim
Department of Electrical Engineering, Bayero University Kano.
muazzamibrahim1234@gmail.com

Muhammad Marwan
Department of Electrical Engineering, Bayero University Kano
marwanmuhammad197@gmail.com

Ibrahim Muhammad Said
Department of Electrical Engineering, Bayero University Kano.
Ibrahimmsd18@gmail.com

Ezak Victory
Cross River University of Technology, Calabar.
ezakvictorypro@gmail.com

Abstract: This paper presents the design and implementation of an IoT-Based Energy Metering System, aimed at improving the monitoring, accuracy, and accessibility of electricity consumption data. Traditional metering methods often lack real-time monitoring, are prone to human errors, and limit user interaction with consumption data. Motivated by the growing demand for efficient energy management, this paper proposes a smart solution using modern IoT technology to automate data acquisition and visualization. The system integrates a PZEM-004T energy monitoring module with an ESP8266 NodeMCU microcontroller. It measures electrical parameters including voltage, current, power, energy, frequency, and power factor. These parameters are processed and displayed locally on a 20x4 LCD and remotely via the ThingSpeak IoT dashboard. The software was developed using Arduino IDE with relevant libraries for sensor interfacing and cloud communication. Testing with real loads confirmed the system's functionality and accuracy, showing reliable synchronization between the local display and cloud platform. Results demonstrated that users can effectively track and analyse their energy usage in real-time, both onsite and remotely. In addition, an accuracy test was performed on the system using standard measuring devices. In conclusion, the IoT-based Energy Metering System offers a low-cost, user-friendly, and efficient approach to energy monitoring. Future enhancements are recommended, such as integrating billing features, mobile alerts, and renewable energy compatibility to extend the system's utility in smart home and industrial applications.

Keywords: IoT, ESP8266, Energy Consumption, Current Transformer, Arduino, Thing Speak, PZEM-004T.

Nomenclature

Abbreviation	Description
IoT	Internet of Things
LCD	Liquid Crystal Display
AC	Alternating Current
ESP	Espressif (family name for ESP8266/ESP32 chips)
MCU	Microcontroller Unit
UART	Universal Asynchronous Receiver-Transmitter
IDE	Integrated Development Environment
Wi-Fi	Wireless Fidelity
API	Application Programming Interface
I2C	Inter-Integrated Circuit
STN	Super Twisted Nematic (LCD technology)
GSM	Global System for Mobile Communications
LoRaWAN	Long Range Wide Area Network
NB-IoT	Narrowband Internet of Things
SPI	Serial Peripheral Interface
OLED	Organic Light Emitting Diode
TFT	Thin Film Transistor
HMI	Human Machine Interface
TTL	Transistor-Transistor Logic
CPU	Central Processing Unit
RAM	Random Access Memory
SDK	Software Development Kit
SDA	Serial Data Line
SCL	Serial Clock Line

© Resbee Info Technologies Pvt Ltd

1. Introduction

1.1 Background

The most usable form of energy (electrical energy) is used widely throughout the world. With the evolution of modern technology, the usage of electricity is escalating gradually. According to their inspection, the electric bills are prepared, and most often these are prepared on the basis of assumptions, which could be inaccurate. Due to the absence of a regular monitoring system, sometimes consumers don't know about the usage of electrical energy in their homes. A remote monitoring of the energy meter and digital billing system is implemented through IoT. For the monitoring server, major programming languages have been introduced to relate the methodologies, store data in a database, and send the energy consumption to the IoT dashboard; finally, they will have a clear view of their energy consumption[1].

The manual procedure of reading electricity is the conventional method. Utility companies encounter difficulties in monitoring consumption trends accurately, billing and collecting money on time, handling massive amounts of human data entry, and reducing energy theft and tampering. In the interim, customers encounter difficulty reading meters by hand and restricted availability of consumption data[2].

The emergence of IOT and automation technologies offers a chance to change this procedure. This paper attempts to close the gap between outdated manual systems and contemporary automated solutions by utilizing Arduino and microcontrollers to provide a dependable, effective, and user-friendly electricity meter reading and billing system[3].

This paper aims to design and construct an "IoT-Based Energy Metering System". The system aims to provide a reliable, cost-effective, and user-friendly solution for measuring electricity consumption, processing data, and sharing the data to display via LCD and to an IoT dashboard to access the data in real-time.

1.2 Motivation and Significance

With the rising demand for electricity and the increasing need for efficient energy management, traditional metering systems often fall short in providing real-time monitoring and control. Many consumers lack access to detailed insights about their energy usage, leading to inefficient consumption and higher electricity bills. Moreover, manual meter readings are prone to errors and delays, making them unreliable for accurate billing.

The motivation behind this paper stems from the need to develop a smart, automated, and IoT-enabled energy metering system that provides real-time data access, enhances energy efficiency, and minimizes power wastage. By integrating modern technologies such as the Internet of Things (IoT), users can remotely monitor their energy consumption and take proactive measures to optimize their electricity usage.

This paper, when implemented, has a lot of significance, including:

- **Real-Time Energy Monitoring:** The system provides live updates on electricity consumption, enabling users to track their power usage anytime and anywhere through an IoT dashboard.
- **Cost-Effective and Efficient:** By analyzing real-time data, consumers can identify excessive power usage and adopt energy-saving strategies, leading to reduced electricity costs.
- **Minimizing Human Errors:** Unlike traditional manual meter reading methods, this system ensures accuracy and reliability in data collection and billing.
- **Remote Accessibility:** The IoT integration allows users to access their energy consumption data from any device with an internet connection, promoting convenience and smart energy management.
- **Contribution to Smart Grid Development:** This paper aligns with modern smart grid technologies, supporting efficient energy distribution, demand-side management, and sustainability [4].
- By implementing this IoT-Based Energy Metering System, consumers, utility companies, and policymakers can benefit from an innovative solution that promotes transparency, efficiency, and sustainability in energy consumption.

1.3 Problem Statement

Conventional electricity metering systems depend on manual readings and antiquated technology, frequently resulting in erroneous billing, delayed data processing, and ineffective energy consumption management. Consumers are generally oblivious to their real-time power usage, culminating in waste and elevated electricity expenses. Furthermore, manual meter readings are susceptible to human errors,

tampering, and fraud, rendering them unreliable for precise energy monitoring. In light of the swift advancement of smart technologies and IoT-based solutions, there exists an imperative for a real-time, automated, and remotely accessible energy metering system. This paper confronts these issues by designing and implementing an IoT-Based Energy Metering System, enabling users to monitor their electricity consumption in real-time through an LCD and an IoT dashboard.

By integrating IoT technology, this system enhances user engagement, promotes efficient energy usage, and contributes to the development of smart energy grids, thereby reducing energy waste and optimizing power distribution.

1.4 Aim and Objectives

This paper aims to design and construct an IoT-based energy metering system. The aim is to be achieved through the following objectives:

- To design a system that can accurately measure electrical energy and provide real-time consumption information.
- To create a program using Arduino IDE software to run the system.
- To create an IoT dashboard and link it to the system
- To test the functionality of the system.

1.5 Methodology Materials/Components

- ESP 8266 NodeMCU
- PZEM 004T Module
- LCD 20x4
- I2C Interface Module
- IOT Dashboard (ThingSpeak)
- Arduino IDE software
- Power Supply (5V)
- Connecting wires

The PZEM004T Module comes with a Current Transformer (CT), which is usually used to measure current flowing in a given circuit, and a voltage sensor, which measures the voltage of a source connected across it. This module will be used to measure the current, voltage, power, and energy consumed by the system (load). The PZEM004T Module is to be interfaced with the Arduino board. A program will be written in Arduino IDE to receive the data and calculate the energy consumed. Then, it will send parameters to the LCD and IOT dashboard.

1.6 Scope

This system will measure current, voltage, and power, calculate energy, and display the parameters via LCD and send the data to an IoT dashboard for real-time tracking. Also, the system will only work when there is a power supply and an internet connection.

1.7 Paper Organisation

The remainder of the paper is organized as follows

- Section 2 presents the theoretical background which contains the literature review and the review of all the components used, their operation and application.
- Section 3 presents the detailed design and implementation, which gives the details of how each unit is designed and integrated.
- Section 4 is construction and testing, which comprises the construction work and testing of the device.
- Section 5 presents the conclusion and recommendations.

2. Literature Review and Components Identification

2.1 Review from previous work

The research in [5] presents a smart energy metering system that uses IoT technology to enable real-time monitoring, billing automation, and remote load control. The system integrates a microcontroller, energy metering module, Wi-Fi unit (ESP8266), LCD, and a relay module to track electricity consumption and control appliances remotely. Data is sent to a cloud-based platform, allowing users to view their energy usage online and receive alerts. This setup eliminates manual meter reading, enhances billing accuracy, and encourages energy-efficient habits. Though effective, the system faces challenges such as reliance on stable internet, higher initial cost, and data security concerns. This study provides a solid foundation for developing more advanced, user-friendly energy metering solutions using IoT technology.

The paper [6] provides an insightful review of the current advancements in IoT-based smart energy metering systems. It begins by highlighting the transition from traditional meters to smart meters, which use sensors and communication modules to enable real-time monitoring and remote data access. The authors examine key communication technologies such as Wi-Fi, ZigBee, LoRaWAN, and NB-IoT, noting that long-range, low-power protocols like LoRaWAN are ideal for smart metering. The paper introduces a three-layer IoT architecture—perception, network, and application layers—that forms the basis of most energy metering systems. Additionally, the study explores how cloud and edge computing platforms are used to store and analyse energy data, with AI and machine learning providing predictive capabilities. Applications like smart homes, energy theft detection, and demand forecasting are also discussed. Despite its benefits, the paper acknowledges challenges such as data security and interoperability. Solutions like blockchain and encryption are proposed to address these issues. In conclusion, this paper offers a well-rounded overview of technologies and strategies driving the future of smart energy metering, serving as a valuable resource for anyone involved in IoT-based energy management.

The research in [7] describes the design and implementation of a basic energy meter using Arduino Uno. It measures voltage, current, power, and energy consumption, and displays data on an LCD. The components used include: Arduino Uno, Current Sensor (ACS712), Voltage Sensor (ZMPT101B), LCD Display (16x2), Resistors, Breadboard, Wires, etc. This hardware setup is ideal for a low-cost energy monitoring prototype. Voltage and current sensors measure analog values from the mains; the Arduino reads these analog values and calculates instantaneous power and Energy consumed (by integrating power over time). Values are updated on the LCD. The code handles sampling, filtering (simple averaging), and multiplication to compute power and energy. The meter was tested with various loads. It successfully calculated and displayed energy consumed. Good accuracy was observed for domestic-level appliances.

Researchers in [8] designed and implemented a system that automates energy meter reading and billing using GSM communication, eliminating manual errors and reducing labor costs. They used components like a PIC Microcontroller, an energy meter with optical isolation, a GSM Module, an LCD Screen, and a power supply unit. The microcontroller-based central unit controls meter reading and GSM communication; the GSM Module sends readings via SMS to a central server or billing center; the energy meter interface reads energy usage (in kWh) from a digital energy meter; the LCD Display shows real-time readings and status messages. The energy meter records power consumption; the microcontroller reads the meter pulse count. At set intervals, the microcontroller sends consumption data via GSM to the utility provider; the billing system receives the data and processes it for invoice generation. The system demonstrates a reliable and cost-effective solution to modernize energy metering and billing using GSM technology, particularly useful in developing regions and remote locations.

In [4], the detailed design and implementation of an advanced smart energy metering system that leverages both IoT and cloud computing to enhance smart grid performance was presented. Their key objectives were to develop an IoT-based smart energy meter with real-time monitoring, enable remote data logging, control, and analysis via a cloud platform, and improve energy usage efficiency and management in smart grids. The system integrates a microcontroller (ESP32) for central control and communication, Current and voltage sensors (ACS712, ZMPT101B) for power measurement, a built-in Wi-Fi module built into the ESP32 for internet connectivity, a cloud platform (ThingSpeak) for data storage, visualization, and analysis,

Mobile/PC interface for user interaction with the data. The prototype was tested in a residential environment; it successfully monitored and controlled energy usage remotely, and power measurements were accurate with minimal deviation from standard meters, demonstrating low power consumption and cost-effectiveness.

The paper [9] presents an IoT-based solution for automating the reading of electrical energy consumption. The core aim of the system is to eliminate the need for manual meter reading by human operators, thereby improving efficiency and reducing costs. The methodology employed involves using a Node-MCU as the central controller to acquire energy consumption data from a traditional energy meter. The energy meter's LED terminals are connected to an optocoupler, which senses the pulses generated by the meter, with these pulses corresponding to the electricity used by the consumer. The Opto-Coupler then converts these pulses into electrical signals that are fed into the Node-MCU. The Node-MCU is programmed to count these pulses, with a specific number of pulses (3200 in this case) representing 1 kWh of energy consumption. The system's hardware components consist of the following: An energy meter to measure the electrical energy, a Node-MCU, which is a microcontroller with built-in Wi-Fi, used for data processing and communication, an optocoupler to sense pulses from the energy meter, a power bank module to provide backup power to the system, and an LCD to display the consumed units locally. For data management and communication, Firebase, a mobile and web application development platform, is utilized. The Node-MCU transmits the calculated energy consumption data to the Firebase server via its built-in ESP8266 Wi-Fi module. This data is then made accessible to users through an IoT mobile application developed using MIT App Inventor, a visual programming tool. Additionally, the system incorporates the Blynk app to enable users to remotely control the power supply to their appliances, offering a degree of load management. The results of the implementation demonstrate the system's ability to automate energy consumption monitoring and provide users with real-time access to their energy usage data. The system displays the consumed units and their corresponding cost on both an LCD and the mobile application. This facilitates better energy management and empowers consumers to make informed decisions about their energy consumption.

The paper [10] details the development of a smart energy meter designed to provide consumers with real-time energy consumption data. The primary goal of the paper was to enhance productivity, improve the precision of energy readings, and expedite the process of determining energy consumption in residential settings. The key hardware components include: PZEM-016 AC Energy Meter, RS-485 UART Serial Converter, NodeMCU ESP8266, Arduino Uno R3, LCD Arduino Keypad Module Shield Board. The software components used are: Arduino IDE, used for programming both the Arduino Uno and the NodeMCU ESP8266; and the Blynk IoT Application, a platform for real-time monitoring of the energy consumption data. The system operates by having the PZEM-016 AC Energy Meter measure the electrical parameters, and then the RS-485 UART Serial Converter translates this data for the NodeMCU ESP8266 and Arduino Uno R3. The NodeMCU then sends the data to the Blynk IoT Application for remote monitoring, while the Arduino Uno displays the data on the LCD module for local viewing. The results of testing the smart energy meter in an actual household demonstrated that the energy consumption data displayed on the Blynk IoT App and the LCD closely matched the readings of the existing household energy meter. The device also accurately measured the energy consumption of individual appliances.

The research in [11] addresses the issue of traditional, labor-intensive electricity usage recording methods by proposing an IoT-based kWh meter. The core objective is to develop a system that enables electricity companies to monitor customer electricity usage in real-time and remotely, thereby reducing operational costs. The methodology employed in this research utilizes the waterfall method for system development, encompassing analysis, design, coding, and testing phases. The system's design incorporates a block diagram illustrating the flow of electrical power measurement and data transmission. The components used in the system are: PZEM-004T, ESP32 microcontroller, OLED display, and relay module. The system functions by measuring the electrical load using the PZEM-004T sensor, with the ESP32 microcontroller reading and processing this data. The ESP32 then displays the data on an OLED screen and transmits it to a Firebase Realtime Database. A mobile application allows users to access this data from the cloud. The relay module, controlled by the ESP32, enables remote control of the electrical load. The results of the research demonstrate the successful development of an IoT-based kWh meter. The system was tested for sensor functionality, display accuracy, database connectivity, and load measurement. The measured electrical parameters were compared with those from a commercial Digital Energy Meter to assess accuracy. The results showed a small percentage of error in voltage measurements (0.35% and 1.45% in different tests), indicating that the developed system provides reasonably accurate measurements of electrical power.

The researchers in [12] explore the use of two measurement modules, PZEM-004T and ADE7753, with the Arduino Uno WiFi Rev2 microcontroller, for real-time monitoring of electrical grid frequency. The research aims to provide a comparative evaluation of these modules for this specific application. The methodology involves using both the PZEM-004T and the ADE7753 modules to measure the frequency of

an AC power signal. The Arduino Uno WiFi Rev2 is used as the microcontroller to process and transmit the data. The system is designed to display the measured frequency on various display devices (LCD, OLED, TFT, or HMI) and also make it accessible via a web interface. The key hardware components include:

- PZEM-004T module: A cost-effective module for measuring electrical parameters, including frequency.
- ADE7753 module: A more sophisticated integrated circuit designed for high-accuracy power and energy measurement, also capable of frequency measurement.
- Arduino Uno WiFi Rev2: A microcontroller used to acquire data from the measurement modules, process it, and facilitate data display and transmission.
- Display devices: LCD, OLED, TFT, and HMI displays for local visualization of the measured frequency.

The PZEM-004T module is connected to the Arduino using a TTL adapter and communicates via a serial interface (UART). The ADE7753 module communicates with the Arduino via an SPI interface. Both modules send the measured frequency data to the Arduino, which then displays it locally and/or transmits it for remote access through a web platform. The results of the comparative evaluation show that both modules can be used for electrical grid frequency monitoring. The PZEM-004T is found to be a simpler and more cost-effective solution suitable for basic monitoring, particularly in home automation and IoT applications. The ADE7753, on the other hand, offers higher accuracy and advanced features, making it more suitable for industrial applications requiring precise measurements and power quality analysis.

The research in [13] presents a low-cost IoT system designed to monitor electrical variables in low-voltage networks. The system aims to support the estimation of energy consumption and recognize consumption patterns in homes or small industries. The methodology involves an IoT architecture with a modular design. The system includes an energy meter for measuring electrical variables and a LoRa gateway for wireless communication. The energy meter consists of: Sensing unit: YHDC SCT016S current transducers to measure voltage and current; Sensing signal conditioning: An Atmel M90E32AS chip for calculating electrical variables; Control unit: An ESP32 VROOM microcontroller to obtain readings from the meter chip; Data conditioning and pre-processing: Heltec ESP32 LoRa development boards for communication. The LoRa gateway communicates with the cloud via Wi-Fi. The system measures and transmits variables like current, voltage, active/reactive energy, power factor, etc., in three-phase circuits. A calibration process was performed in an accredited laboratory. The system was tested in a small factory, and the data allowed for the identification of consumption patterns.

In [14], a novel High-Frequency Electric Energy Metering System designed to examine non-conventional features relevant to real-time energy disaggregation and appliance control in households was presented. The system integrates a custom data acquisition and control board with an Arduino Due and the M90E36A Demo Board, enabling flexible electrical energy measurements. The methodology involves using the M90E36A IC for energy measurement, with the system capable of synchronously measuring up to 4 current channels. The M90E36A IC's internal Digital Signal Processor (DSP) can obtain and process samples at 3 Hz in the time domain and 2 Hz in the frequency domain. Additionally, the system can operate in the M90E36A IC's direct access memory mode to obtain 8 kHz voltage and current signals. Integration with a Raspberry Pi allows for the incorporation of a custom signal processor.

The key hardware components are: A custom-designed data acquisition and control board, Arduino Due, M90E36A Demo Board, and Raspberry Pi (for custom signal processing). The system's capabilities are demonstrated through an application example that obtains the variation of residual harmonic components of a household appliance. The paper focuses on the design and capabilities of the hardware. It primarily addresses the limitations of existing commercial energy meters that lack access to firmware and raw data.

The paper [1] discusses an automatic meter reading system that uses GSM (Global System for Mobile Communications) to remotely monitor energy meters and provide digital billing. The system aims to address inaccuracies in traditional billing methods and provide consumers with a clearer view of their energy consumption. The methodology involves using GSM technology to transmit energy consumption data from the meter to a central server. Programming languages are used to manage the data, store it in a database, and send monthly bills to consumers via SMS.

Components: GSM technology for wireless communication, a central server for data storage and processing, and a microcontroller.

The system enables remote monitoring of energy consumption, digital billing is provided to consumers via SMS, and consumers gain a clearer view of their energy consumption. The paper emphasizes the importance of accurate, reliable, and cost-effective communication systems for AMR. The system's reliance on GSM technology implies a limitation in areas with poor GSM network coverage.

The research in [15] discusses an IoT-based electricity energy meter designed to provide real-time monitoring of electrical energy parameters. The system aims to replace traditional electricity meters with

an internet-based system that collects and displays information on energy consumption. The methodology involves using IoT technology to measure electrical energy parameters (voltage, current, power) and transmit this data to a database. The system allows users to observe electricity consumption on a monthly basis through a mobile application and webpage. The system is designed to connect to the electricity service provider to automate electricity bill generation. Wi-Fi connectivity, IoT technology, a mobile application, and a webpage were used to develop the system. After the test was carried out on the system, it was able to perform real-time measurement of voltage, current, and power, Monthly observation of electricity consumption by users, and automated generation of electricity bills.

2.2 Theoretical Background

2.2.1 Microcontroller

A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory, and input/output (I/O) peripherals on a single chip.

Microcontrollers, sometimes referred to as embedded controllers or microcontroller units (MCUs), are found in many different goods, including robots, home appliances, medical devices, mobile wireless transceivers, office equipment, cars, and vending machines. In essence, they are tiny, extremely basic personal computers (PCs) that lack a sophisticated front-end operating system and are made to function as tiny versions of larger components (OS).

2.2.1.1 NodeMCU

The NodeMCU (Node MicroController Unit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. Having a clock speed of 80 MHz, the NodeMCU is available in various package styles. Common to all the designs is the base ESP8266 core. Designs based on the architecture have maintained the standard 30-pin layout.

2.2.1.2 ESP8266

The ESP8266 shown is fig. 1 is a low-cost Wi-Fi microcontroller with built-in TCP/IP networking software and microcontroller capability. The ESP8266, designed and manufactured by Espressif Systems, contains the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds. The pins of the ESP8266 are explained as follows:

- **Power Pins:** There are four power pins. VIN pin and three 3.3V pins.
  - VIN can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on VIN is regulated through the onboard regulator on the NodeMCU module – you can also supply 5V regulated to the VIN pin.
  - 3.3V pins are the output of the onboard voltage regulator and can be used to supply power to external components.
- **GND:** are the ground pins of NodeMCU/ESP8266
- **I2C Pins:** are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that the I2C clock frequency should be higher than the slowest clock frequency of the slave device.
- **GPIO Pins:** NodeMCU/ESP8266 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light, and Button programmatically. Each digital-enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.
- **ADC Channel:** The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using the ADC. Test the power supply voltage of the VDD3P3 pin and test the input voltage of the TOUT pin. However, they cannot be implemented at the same time.
- **UART Pins:** NodeMCU/ESP8266 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin) features only a data transmit signal, so it is usually used for printing logs.
- **SPI Pins:** NodeMCU/ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:
  - 4 timing modes of the SPI format transfer

- Up to 80 MHz and divided clocks of 80 MHz
- Up to 64-Byte FIFO

- **SDIO Pins:** NodeMCU/ESP8266 features Secure Digital Input/Output Interface (SDIO), which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.
- **PWM Pins:** The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 µs to 10000 µs (100 Hz and 1 kHz).
- **Control Pins:** are used to control the NodeMCU/ESP8266. These pins include the Chip Enable pin (EN), Reset pin (RST), and WAKE pin.
  - **EN:** The ESP8266 chip is enabled when the EN pin is pulled HIGH. When pulled LOW, the chip works at minimum power.
  - **RST:** The RST pin is used to reset the ESP8266 chip.
  - **WAKE:** The WAKE pin is used to wake the chip from deep-sleep.

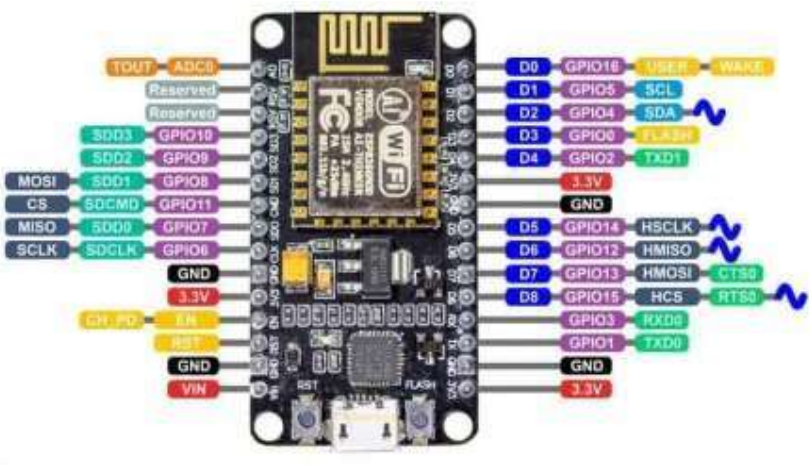


Fig. 1. *NodeMCU/ESP8266 Pinout*

2.2.2 PZEM-004T Module

The PZEM-004T is a popular energy monitoring module that's widely used in IoT and embedded system projects. The PZEM-004T module is connected to the NodeMCU using a TTL interface and communicates via a serial interface (UART) [12]. The module measures the voltage and current of a single-phase AC circuit using an integrated voltage sensor and a non-invasive current transformer (CT). The module sends the measured data to the NodeMCU using a serial communication protocol (RS485 or UART). The fig. 2 below shows a typical picture of the PZEM-004T Module.



Fig. 2. *PZEM-004T Module.*

Each of the parameters is obtained as follows:

- **Voltage:** The PZEM-004T connects directly to the AC supply line. It uses a voltage sensor to measure line voltage. The voltage is measured in real-time and updated continuously.
- **Current:** A CT (Current Transformer) clamp is connected to one of the AC wires (usually the live wire). The CT senses magnetic flux generated by current flow and produces a small signal. The PZEM module converts this signal into a measurable current value.
- **Power (Active Power):** Once voltage and current are known, the module uses the formula:

$$Power\ (W)\ =\ V \times I \times Power\ Factor \tag{1}$$

The power factor is dynamically calculated and used to derive real power (not apparent power).

- **Power Factor:** The power factor is calculated by comparing the phase difference between the voltage and current waveforms. A pure resistive load (like a heater) will have a PF ≈ 1.0, while inductive loads (like a fridge or fan) may have PF < 1.0.
- **Energy (kWh):** The energy consumed is the integration of power over time, typically:

$$Energy\ (kWh)\ =\ \sum Power\ (kW) \times Time\ (h) \tag{2}$$

- **Frequency:** The module detects zero-crossings of the AC signal to measure frequency. Frequency is the Number of cycles per second, typically 50 Hz in most countries like Nigeria.

Table 1 gives the design specifications of PZEM-004T Module.

Table 1. *Design Specifications of PZEM-004T Module*

Specification	Details
Power Supply	+5V DC
Voltage Measurement	80V AC to 260V AC
Current Measurement	0 – 100A AC
Power Measurement	0 – 22kW
Energy Measurement	0 – 9999 kWh
Frequency Range	45 – 65 Hz
Power Factor Range	0.00 – 1.00
Accuracy	± 1% for voltage, current, and power measurements
Baud Rate	UART (TTL Serial) @ 9600 bps (8N1)
Communication Protocol	TTL serial data communication interface for parameter reading and setting
Dimensions	Board Size: 7.3 x 3 cm Current Transformer Size: 3.1 x 2.9 x 4.7 cm Wire Length: 20 cm
Operating Temperature	-10°C to +60°C
Isolation	Opto-isolated UART
LED Indicator	On-board LED flashes to indicate UART activity or power status

2.2.3 Liquid Crystal Display (LCD)

A Liquid Crystal Display (LCD) is a flat-panel display technology that uses liquid crystals to modulate

light, creating images by controlling the alignment of molecules in response to an electric field. LCDs are the most commonly used display device and are used in a wide range of consumer and commercial equipment due to their low operating power, great image quality, and long working life. This includes anything from small monochrome displays like those seen in calculators and watches to huge, high-resolution colour displays like those found in phones, televisions, tablets, and computers. A 20x4 LCD can display 16 characters on each of its 4 lines. Table 2 gives the design specifications of LCD.

Table 2. *Design Specifications of Liquid Crystal Display (LCD)*

Specification	Details
Display Format	20 characters × 4 lines (80 characters total)
Controller IC	HD44780 or compatible
Interface Type	Parallel (4-bit or 8-bit mode), optionally via I2C using a PCF8574 module.
Display Technology	STN (Super Twisted Nematic), positive reflective
Character Size	5 × 8 pixel matrix per character
Character Pitch	6.0 mm (W) × 8.55 mm (H)
Module Size	98.0 mm (W) × 60.0 mm (H) × 12.0 mm (D)
Viewing Area	6 mm (W) × 25 mm (H)
Operating Voltage	+5V DC
Logic Voltage	2.7V to 5.5V
Backlight	Usually LED (white or yellow-green) with current-limiting resistor
Backlight Voltage	+5V DC (via transistor or direct pin with resistor)
Power Consumption	~1 to 2 mA (without backlight); ~20 mA with backlight
Operating Temperature	-20°C to +70°C
Interface Pins	16 pins (if direct connection); 4 with I2C module
Communication Speed (I2C)	Typically 100 kHz (standard mode)

2.2.3.1 I2C Module

The LCD cannot be interfaced with the microcontroller directly. However, an I2C module can be used to interface the LCD and the microcontroller using the SDA and SCL pins of the I2C. SDA is the serial data line used for transmitting and receiving data, while SCL is the serial clock line used to synchronize the data transfer. The 16 pins of the I2C are connected to the 16 pins of the LCD. Fig. 3 shows the pinout diagram of I2C module.



Fig. 3. *Pinout diagram of I2C module*

The table 3 shows the design specification of the I2C module.

Table 3. *I2C Design specifications*

Specification	Details
Full Name	Inter-Integrated Circuit (I²C or I2C)
Communication Type	Serial, synchronous, master-slave
Number of Wires	2 (SDA – Serial Data, SCL – Serial Clock)
Maximum Bus Devices	Up to 128 devices (7-bit addressing); 1024 with 10-bit addressing
Speed Modes	- Standard: 100 kbps \n- Fast: 400 kbps \n- Fast Plus: 1 Mbps \n-High Speed: up to 3.4 Mbps
Voltage Levels	Operates typically at 3.3V or 5V (depends on device compatibility)
Pull-Up Resistors	Required on SDA and SCL lines (typically 4.7kΩ – 10kΩ)
Bus Arbitration	Multi-master capable; uses open-drain lines and clock stretching.
Data Frame	8-bit data + 1-bit ACK/NACK
Addressing	7-bit or 10-bit address for each slave device
Power Consumption	Low, suitable for embedded and battery-powered systems

2.2.4 IoT Dashboard

Thing Speak is a cloud-based platform specifically designed for IoT applications. It receives and visualizes data sent from the ESP8266 over the internet. Users can log in to their dashboard and view live charts of their energy consumption. It supports data logging, graph plotting, and alerts, which makes it very practical for monitoring trends over time. It is accessible via web browsers or mobile apps, enabling remote access to data from anywhere in the world. This enhances the system's usability in smart home, energy-saving, and demand-response scenarios.

2.2.4.1 Key Features and Functionalities:

- Data Collection and Storage: Thing Speak allows devices to send data to the cloud, where it's stored in channels. Each channel can hold up to 8 fields of data, enabling the storage of various sensor readings or parameters.
- Data Visualization: The platform provides built-in tools for visualizing data through charts, graphs, and gauges. Users can create custom dashboards to display real-time and historical data.
- Data Analysis: Thing Speak offers basic data processing and analysis capabilities, including mathematical operations, aggregations, and time-based calculations. It also integrates with MATLAB for more advanced analytical tasks.
- Triggers and Alerts: Users can set up triggers to perform actions based on specific data conditions. For example, an alert can be sent via email or SMS if a sensor reading exceeds a certain threshold.
- API Integration: Thing Speak provides APIs that allow devices and applications to interact with the platform programmatically. This enables seamless data transfer and integration with other systems.

2.2.4.2 Applications of ThingSpeak IoT dashboard

Thing Speak is used in a wide range of IoT applications, including:

- Environmental Monitoring: Collecting and visualizing data from weather stations, air quality sensors, and water level sensors.
- Smart Agriculture: Monitoring soil moisture, temperature, and humidity in agricultural fields.
- Home Automation: Tracking energy consumption, controlling appliances, and monitoring security systems.
- Industrial IoT: Monitoring equipment performance, predicting maintenance needs, and optimizing production processes.

3. Design and Analysis

3.1 Hardware Components and Design

The IoT-Based Energy Metering System is designed to provide real-time monitoring and analysis of electricity consumption. The system is made up of several essential functional elements that work together to collect, process, transmit, and display energy usage information. These units are the energy sensing module, the data processing and communication module, and the data visualization and storage module. The energy sensing module measures electrical parameters accurately, the data processing and communication module handles data acquisition, processing, and wireless transmission, and the data visualization and storage module provides a user-friendly interface for accessing and analyzing energy consumption data. The combination of these components allows the system to provide comprehensive energy monitoring capabilities. Fig. 4 shows the circuit diagram and Fig. 5 shows the block diagram of the system.

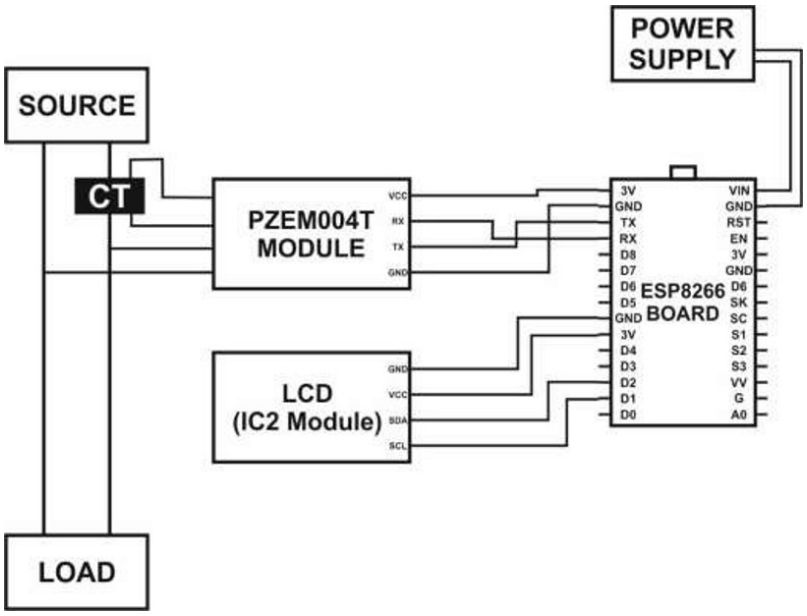


Fig. 4. *Circuit diagram of the system*

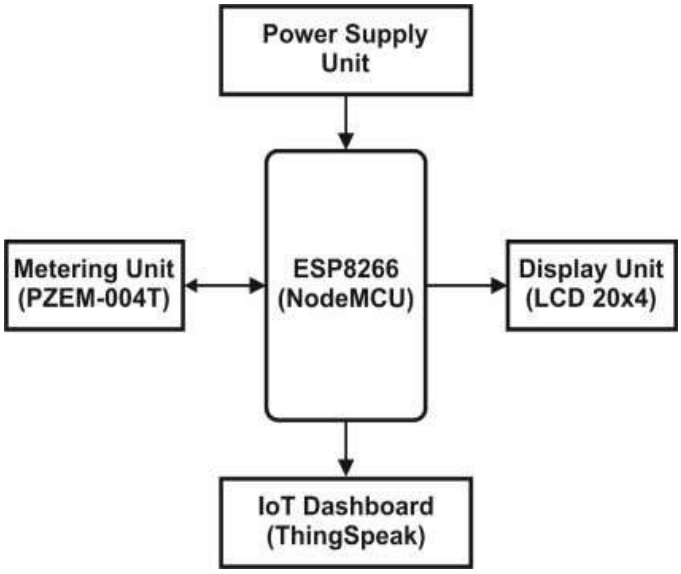


Fig. 5. *Block Diagram of the system*

3.1.1 Power Supply Unit

This unit is responsible for supplying a stable 5V DC voltage required to operate all components of the system, particularly the ESP8266 and the other modules connected to it. It will be tapped from the source via a 5V voltage regulator adapter, and then a cable will be used to power the ESP8266. Voltage regulation is crucial to avoid fluctuations that can cause data errors or hardware malfunctions. The power supply ensures the system remains portable and usable.

3.1.2 Control Unit

This is the central processing unit of the system. It is a powerful microcontroller with built-in Wi-Fi capability, making it ideal for IoT applications. In this paper, the ESP8266 performs the following functions:

- Data Collection: It communicates with the PZEM-004T energy meter via a serial interface (e.g., UART) to fetch real-time electrical parameters like voltage, current, power, and energy consumed. The interfacing is done via the RX and TX pins (transmitting and receiving pins, respectively) of the module.

- Data Processing: The raw data received from the sensor is processed, formatted, and prepared for both local and remote display.
- Display Control: It sends processed data to the LCD 20x4 through I2C for visualization. The ESP8266 is interfaced with the display unit via the pins of the I2C module. Pin D1 (GPIO5) and D2 (GPIO4) of the microcontroller are to be connected to the SCL and SDA pins of the I2C module, respectively, to enable serial data transfer to the LCD.

- IoT Communication: Using its built-in Wi-Fi, it connects to a specific network and pushes the energy data to the ThingSpeak cloud platform at regular intervals for remote monitoring (an interval of 20 seconds was used).

3.1.3 Metering Unit

The PZEM-004T module measures the parameters as follows:

- Voltage: The PZEM-004T connects directly to the AC supply line. It uses a voltage sensor to measure line voltage. The voltage is measured in real time and updated continuously.
- Current: A CT (Current Transformer) clamp is connected to one of the AC wires (usually the live wire). The CT senses magnetic flux generated by current flow and produces a small signal. The PZEM module converts this signal into a measurable current value.
- Power (Active Power): Once voltage and current are known, the module uses the formula:

$$P = V \times I \times PF$$

Where:

- P is the active power of the system in watts
- V is the voltage obtained by the voltage sensor in volts
- I is the current measured by the current transformer in Amperes
- PF is the power factor of the system

NB: The power factor is dynamically calculated and used to derive real power (not apparent power).

- Power Factor: The power factor is calculated by comparing the phase difference between the voltage and current waveforms.
- Energy (kWh): The energy consumed is the integration of power over time, typically:

$$E = \sum P \times T$$

Where:

- E is the energy consumption of the load in kWh
- $\sum P$ is the summation (integration) of the active power consumed by the load in kW
- T is the time of consumption in Hours

- Frequency: The module detects zero-crossings of the AC signal to measure frequency.

3.1.4 Display Unit

This Liquid Crystal Display (20x4) shows data in a human-readable format on-site. It is interfaced with an I2C adapter, which reduces the number of GPIO pins used on the NodeMCU.
i.e., VCC, GND, SDA, SCL pins. The parameters will be displayed via the LCD, meaning the energy consumption can be monitored locally without relying solely on the cloud dashboard.

3.1.5 IoT Dashboard

The IoT dashboard to be used in this project is the ThingSpeak dashboard. An account is created using a university email (mii1800022.ele@buk.edu.ng) via MathWorks. After the successful creation of an account, a new channel is opened, which will be used to send the data. The channel number and the API key of the channel are used to connect the WiFi module to the IoT dashboard. A chart and numeric display widget were added to show the parameters for visualization.

3.2 Software Design and Analysis

This section describes the control software written to drive the operations of the IoT-Based Energy Metering System. The program was developed using the Arduino IDE, with the C++ programming

language and several external libraries used to facilitate communication with the system's components. The program handles the acquisition, display, and transmission of energy metering data.

Step 1: Inclusion of Libraries and Module Definitions

The first step in the software development process involved including the necessary libraries to support communication with hardware peripherals. These libraries include:

- ESP8266WiFi.h and ThingSpeak.h for Wi-Fi and IoT communication,
- PZEM004Tv30.h and SoftwareSerial.h for data acquisition from the PZEM-004T module,
- Wire.h and LiquidCrystal_I2C.h for interfacing with the 20x4 LCD via I2C.

The LCD was initialized with I2C address 0x27 using the object lcd(0x27, 16, 4), while the PZEM-004T module was defined using the ESP8266 GPIO pins D1 and D2 (SCL and SDA, respectively).

Step 2: Wi-Fi Connection and Module Initialization

In the setup() function, the system establishes a connection to the specified Wi-Fi network using WiFi.begin()and continuously checks for a successful connection using a while loop. The ThingSpeak platform was initialized to allow for the transmission of energy data to the cloud for real-time monitoring. Additionally, the LCD was initialized using lcd.init() and backlight enabled with lcd.backlight(), followed by an initial welcome display message.

Step 3: Acquisition, Display, and Transmission of Data.

In the loop() function, energy parameters including voltage, current, power, energy, frequency, and power factor are acquired from the PZEM-004T module using the library functions (pzem.voltage(), pzem.current(), etc.).

The acquired data is then displayed on the LCD in four rows (one for each key metric: voltage, current, power, and energy), printed to the serial monitor for debugging and verification, and transmitted to ThingSpeak via ThingSpeak.setField() and ThingSpeak.writeFields() for remote IoT monitoring.

To comply with ThingSpeak's API rate limits, a delay of 20 seconds (delay(20000)) is used between each transmission loop.

4. Construction and Testing

4.1 Construction of the System

Construction Steps: The construction of the IoT-Based Energy Metering System was carried out in the following sequential steps:

- **Step 1:** Component Arrangement

All necessary components—NodeMCU ESP8266, PZEM-004T module, 20x4 LCD with I2C, and CT sensor—were gathered and tested individually to ensure they were functional.

- **Step 2:** Circuit Design

The NodeMCU served as the central controller, with UART communication lines connected to the PZEM-004T, and I2C lines to the LCD. A 5V regulated power supply was connected to power the board.

- **Step 3:** Prototyping on Breadboard

The circuit was initially constructed on a breadboard to test the integration between components. The CT clamp was installed on a test AC line to measure current flow.

- **Step 4:** Code Uploading

The control program, written in Arduino IDE, was uploaded to the NodeMCU. Serial Monitor and LCD output were used to debug and verify communication between modules.

- **Step 5:** Integration with ThingSpeak

Wi-Fi credentials and API keys were configured to establish communication with the ThingSpeak IoT platform. Real-time data upload was tested and confirmed through the online dashboard.

- **Step 6:** Final Assembly

After successful testing, components were securely mounted inside a project enclosure (Adaptable Box). Wires were soldered and connections neatly arranged to reduce noise and enhance stability. The fig. 6 shows the picture of the system after final assembly.

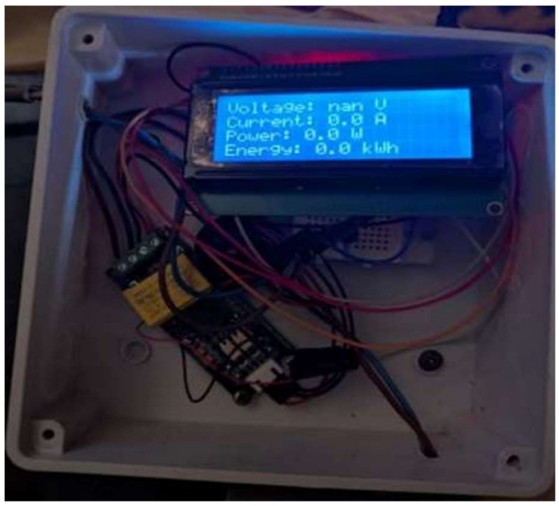


Fig. 6. *Picture of the system after final assembly*

4.2 Testing and Results

- **Testing Procedure:** The system was connected to a standard AC load, upon powering up, the LCD displayed real-time voltage, current, power, and energy values. The ThingSpeak dashboard was monitored simultaneously to ensure values were uploaded every 20 seconds. Loads were connected to the system to test the response of the system. A clamp multimeter were used to verify the voltage and current measure by the system.
- **Observed Results:** The table 4 shows the result obtained from the LCD display as well as the ThingSpeak dashboard when a 5W bulb and a Laptop were connected to the system.

Table 4. *Observed Results*

Parameter	LCD Display	ThingSpeak Dashboard
Voltage	230.4 V	230.40 V
Current	0.2 A	0.17 A
Power	21.50 W	21.5 W
Energy	0.4 kWh	0.373 kWh
Frequency	-	50.1 Hz
Power Factor	-	0.420

The system functioned reliably under various load conditions. There was close correlation between LCD readings and ThingSpeak data, with minor tolerances well within acceptable error margins. This confirms the accuracy, stability, and IoT capability of the design.

5. Conclusion

The IoT-Based Energy Metering System was successfully implemented and tested. It allows users to monitor power consumption both locally (on the LCD) and remotely (via IoT dashboard). The system is useful for tracking energy usage, reducing wastage, and making informed decisions about power consumption.

The paper met its objectives and can be a valuable tool for smart homes, small businesses, or educational use.

5.1 Recommendation for future work

- **Add Mobile Alerts:** Include SMS or app notifications when energy usage crosses a set limit.
- **Add Billing Functionality:** Automatically calculate and display estimated electricity bills.
- **Support Renewable Energy:** Modify the system to monitor solar power usage or hybrid systems.

Compliance with Ethical Standards

Conflicts of interest: Authors declared that they have no conflict of interest.

Human participants: The conducted research follows the ethical standards and the authors ensured that they have not conducted any studies with human participants or animals.

References

[1] S. Lakshmanan, S. Kiruthika Devi, S. M. Nevetha, and V. Krishnakumaran, "Automatic Meter Reading Using GSM Via SMS," *SSRN Electron. J.*, 2018, doi: 10.2139/ssrn.3141117.

[2] F. Potortì, D. La Rosa, and F. Palumbo, "HardwareX Enerduino-pro : Smart meter led probe using Arduino," vol. 15, no. April, 2023.

[3] J. Rahul, S. Sharma, M. Saxena, R. Chapola, and R. K. Yadav, "GSM BASED SMART ENERGY METER USING ARDUINO UNO," no. 06, pp. 2088–2092, 2022.

[4] M. Güçyetmez and H. S. Farhan, "Enhancing smart grids with a new IOT and cloud-based smart meter to predict the energy consumption with time series," *Alexandria Eng. J.*, vol. 79, no. June, pp. 44–55, 2023, doi: 10.1016/j.aej.2023.07.071.

[5] S. Prabhakaran, R. Sindhupriya, and V. Subhashini, "Smart Energy Meter Auditing With Power Demand Controller Using Internet of Things," *Irjmets.Com*, pp. 2582–5208, [Online]. Available: https://www.irjmets.com/uploadedfiles/paper/volume3/issue_2_february_2021/6388/1 628083269.pdf

[6] M. E. Véliz, G. E. Real, and A. D. Otero, "HardwareX Flexible and configurable embedded electrical energy measurement system to acquire and process high-frequency features," *HardwareX*, vol. 19, no. May, p. e00539, 2024, doi: 10.1016/j.ohx.2024.e00539.

[7] A. S. D. Al-arga, "Designing an Energy Meter Using Arduino Uno," vol. 1, no. 1, pp. 30–37, 2021.

[8] K. Chaudhary, D. Kumar, R. Singh, and M. K. Singh, "GSM Based Energy Meter Reading and Billing System," *Int. J. Adv. Res. Innov.*, vol. 2, no. 1, pp. 101–104, 2014, doi: 10.51976/ijari.211419.

[9] R. Kumar, R. Bhat, and V. Nagar, "AUTOMATIC WIRELESS ELECTRICITY," vol. 11, no. 6, pp. 96–102, 2023.

[10] A. P. Nebrida, A. Based, and S. Energy, "Arduino Based Smart Energy Meter," no. February, pp. 1–7, 2024.

[11] D. Innovation, "Machine Learning in Business Analytics: Advancing Statistical Methods for Data- Driven Innovation," pp. 104–111, 2023, doi: 10.32996/jcsts.

[12] M. Kaci, S. Merabti, and H. A. Said, "Precise and reliable monitoring of electrical grid frequency using PZEM-004T and ADE7753 modules with Arduino Monitoramento preciso e confiável da frequência da rede elétrica usando os módulos PZEM-004T e ADE7753 com Arduino Monitorización precisa y fiabl," no. December, 2024, doi: 10.54021/seesv5n2-733.

[13] N. E. Guevara, Y. H. Bolaños, J. P. Diago, and J. M. Segura, "Development of a low-cost IoT system based on LoRaWAN for monitoring variables related to electrical energy consumption in low voltage networks," *HardwareX*, vol. 12, p. e00330, 2022, doi: 10.1016/j.ohx.2022.e00330.

[14] M. E. Véliz, G. E. Real, and A. D. Otero, "Flexible and configurable embedded electrical energy measurement system to acquire and process high-frequency features," *HardwareX*, vol. 19, no. May, p. e00539, 2024, doi: 10.1016/j.ohx.2024.e00539.

[15] T. K. Gannavaram V, "IoT Based Electricity Energy Meter," *SSRN Electron. J.*, 2021, doi: 10.2139/ssrn.3875420.

APPENDIX

```
//Muazzam Ibrahim code
// Libraries
#include <Arduino.h> #include <ESP8266WiFi.h> #include <ThingSpeak.h> #include <PZEM004Tv30.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h> #include <Wire.h>
// LCD setup
LiquidCrystal_I2C lcd(0x27, 16, 4);
int SCL_pin = 14; //Pin D5 (Connect the SCL pin of the lcd to pin D5) int SDA_pin = 12; //Pin D6 (Connect
the SDA pin of the lcd to pin D6)
// ThingSpeak Channel ID and API Key unsigned long myChannelNumber = 123456;
const char *myWriteAPIKey = "V65BMEDZS0BYOLBO";
// WiFi network info
const char *ssid = "MUAZZAM";   // wifi username
const char *wifiPassword = "12345678";  // wifi password
// PZEM-004T pins and object PZEM004Tv30 pzem(14, 12);  // RX, TX
// Meter values
float Voltage, Current, Power, Energy, Frequency, PowerFactor;
// WiFi client WiFiClient client; void setup() { Serial.begin(115200);
// Connect to WiFi WiFi.begin(ssid, wifiPassword);
Serial.print("Connecting to WiFi");
```

```
while (WiFi.status() != WL_CONNECTED) { delay(500);
Serial.print(".");
}
Serial.println("\nWiFi connected");
// Initialize ThingSpeak ThingSpeak.begin(client);
// Initialize the LCD lcd.init(); lcd.backlight(); lcd.setCursor(0, 0);
lcd.print("Muazzam Energy Meter"); delay(2000);
lcd.clear();
}
void loop() {
// Get meter values Voltage = pzem.voltage(); Current = pzem.current(); Power = pzem.power(); Energy = pzem.energy();
Frequency = pzem.frequency(); PowerFactor = pzem.pf();
// Display values in Serial Monitor Serial.println("Measurement Results:"); Serial.print("Voltage: "); Serial.print(Voltage, 3);
Serial.println(" V"); Serial.print("Current: ");
Serial.print(Current, 3); Serial.println(" A"); Serial.print("Power: "); Serial.print(Power, 3); Serial.println(" W"); Serial.print("Energy: "); Serial.print(Energy, 3); Serial.println(" kWh"); Serial.print("Frequency: "); Serial.print(Frequency, 2); Serial.println(" Hz"); Serial.print("Power Factor: "); Serial.println(PowerFactor, 3); Serial.println("");
// Display values on the LCD lcd.clear();
lcd.setCursor(0, 0); lcd.print("Voltage: "); lcd.print(Voltage, 1); lcd.print(" V"); lcd.setCursor(0, 1); lcd.print("Current: "); lcd.print(Current, 1); lcd.print(" A"); lcd.setCursor(0, 2); lcd.print("Power: "); lcd.print(Power, 1); lcd.print(" W"); lcd.setCursor(0, 3); lcd.print("Energy: ");
lcd.print(Energy, 1); lcd.print(" kWh");
// Send data to ThingSpeak ThingSpeak.setField(1, Voltage); ThingSpeak.setField(2, Current); ThingSpeak.setField(3, Power); ThingSpeak.setField(4, Energy); ThingSpeak.setField(5, Frequency); ThingSpeak.setField(6, PowerFactor);
// Write data to ThingSpeak
int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey); if (x == 200) {
Serial.println("Data sent to ThingSpeak successfully");
} else {
Serial.print("Failed to send data. HTTP error code: "); Serial.println(x);
}
// Wait 20 seconds to comply with ThingSpeak rate limit delay(20000);
}
```