World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(Research Article)

An IoT-Enabled High-Frequency Smart Energy Meter with Web Based Data Management and Control

Ammar Abdussamad Umar *, Aliyu Nura Salisu, Lucky Ajidoku, Mohammad-Jamiu Babatunde Balogun and Zainal Abidina Abdulhadi

Department of Electrical Engineering, Bayero University, Kano, Kano State, Nigeria.

World Journal of Advanced Engineering Technology and Sciences, 2026, 18(01), 304-320

Publication history: Received on 11 December 2025; revised on 18 January 2026; accepted on 21 January 2026

Article DOI: https://doi.org/10.30574/wjaets.2026.18.1.0041

Abstract

The primary feature of a smart grid is the automation of processes within the entire power system. The control is not possible without an accurate data showing full demand on the grid. This paper presents the design of a high frequency smart energy meter that captures accurate energy consumption in real time utilizing the ESP32 microcontroller, MQTT communication protocol, PHP web application, and SQL database. The design enables automatic energy measurement and transmission of the measured data in real-time to a web server using the MQTT protocol, where the data is securely stored in an SQL database. The system also includes an integrated online payment platform, allowing customers to prepay their energy bills. The energy consumption data from the system can then be used in different ways such as forecasting load expansion overtime, knowing system expansion over time, and system automation.

Keywords: ESP32 Microcontroller; Smart Grid; MQTT Communication Protocol; Web Management System; SQL Database; PHP

1. Introduction

Smart grid is an advanced power network that uses digital communication technologies and automation to manage electricity generation, transmission, and distribution more efficiently and reliably. It integrates renewable energy sources, enables real-time monitoring and control, and facilitates two-way communication between utilities and consumers, allowing for better management of supply and demand, improved energy efficiency, and enhanced resilience to disruptions.

The main source of control and decision making is the energy meter data because it shows the load variation over a specific period of time thus making it easy for utility company to analyze the data and deploy models that will be used to optimize the operation of the whole system. High frequency energy meters transmit energy data to a dedicated server every small interval of time ranging from fraction of seconds to a minute [1]. The frequency at which data is recorded in the database gives more clarity about the energy change over time and the higher the frequency of measurement, the higher the accuracy of the energy measured.

Smart energy meters form a network containing multiple devices transmitting to a dedicated server. Therefore, to design a complete smart energy meter, the design of the communication protocol, and server endpoint has to be taken into consideration. The bilateral flow of information on the network must be secured to ensure data security and integrity [2]. The choice of communication protocol is very important. In the case of high frequency data transmission, HTTP is not recommended given the number of devices trying to send a request and get a response within the same

* Corresponding author: Ammar Abdussamad Umar

interval of time. Most of these request/responses will most likely not get recorded by the database given the number of requests in a second since we are dealing with a high frequency data transmission.

2. Material and methods

The design of smart energy meter requires the design of multiple systems that will work simultaneously with the devices. These systems are:

- Smart Energy Meter
- Server Endpoint and User Interface
- Communication Protocol

The entire system is represented in Figure 1.

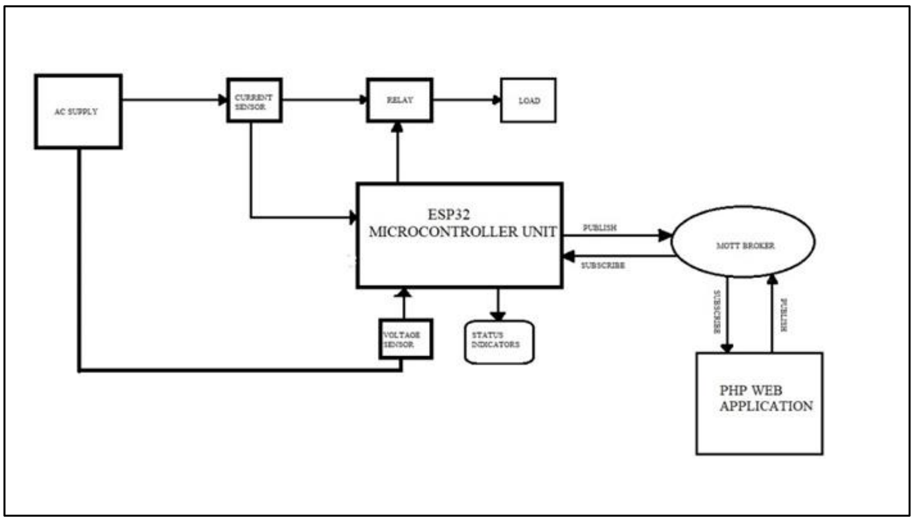


Figure 1 System Design

The section representing the smart energy meter as seen in Figure 1, consist of a microcontroller unit, relay, current & voltage sensors, and status indicators. The microcontroller serves as the heart of the system because it takes inputs such as the output of the voltage and current sensors, and it establish communication with the server through MQTT communication protocol. The smart energy meter section consists of three major components.

- Hardware
- Firmware
- Software

2.1. Hardware Design

These are the physical devices that constitute the smart energy meter. Figure 2 below shows the hardware part of the system in green square. It consists of four components:

- ESP32 Microcontroller
- ACS712 Current Sensor
- ZMPT101B Voltage Sensor
- Relay Unit

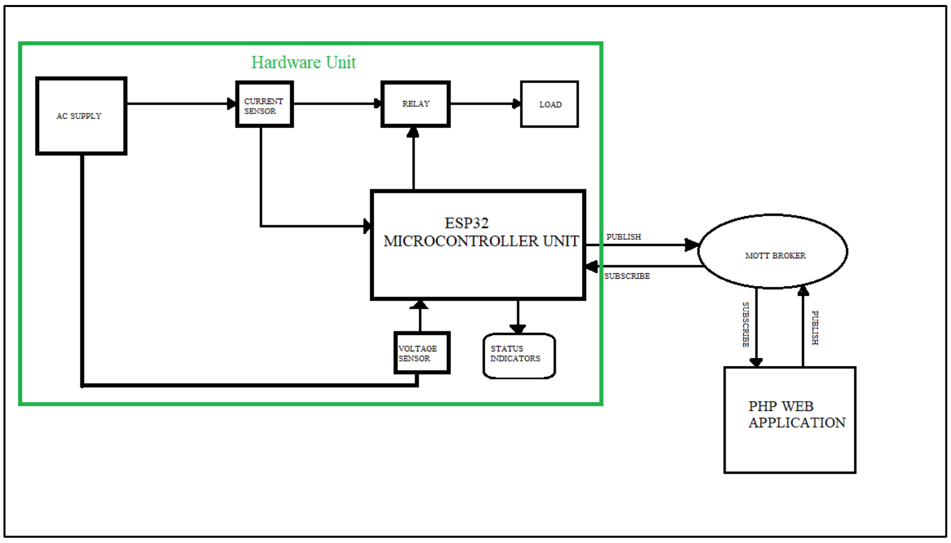
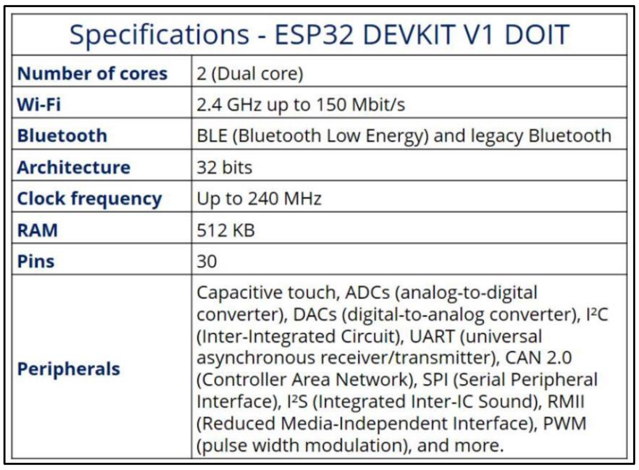


Figure 2 Hardware Unit of The System

2.2. ESP32 Microcontroller Unit

The choice of microcontroller to execute the design is based on the specifications of the system. The system requires an interface to communicate with the internet, ADC with sufficient frequency response, and sufficient clock speed to avoid latency and data loss. These requirements were pointing to ESP32 microcontroller because it has the specifications indicated by the figure below



Specifications - ESP32 DEVKIT V1 DOIT	
Number of cores	2 (Dual core)
Wi-Fi	2.4 GHz up to 150 Mbit/s
Bluetooth	BLE (Bluetooth Low Energy) and legacy Bluetooth
Architecture	32 bits
Clock frequency	Up to 240 MHz
RAM	512 KB
Pins	30
Peripherals	Capacitive touch, ADCs (analog-to-digital converter), DACs (digital-to-analog converter), I^2C (Inter-Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Network), SPI (Serial Peripheral Interface), I^2S (Integrated Inter-IC Sound), RMII (Reduced Media-Independent Interface), PWM (pulse width modulation), and more.

Figure 3 ESP32 Specifications [3]

2.3. ACS712 Current Sensor

The ACS712 is a linear current sensor based on the Hall Effect. It offers a cost-effective and accurate method for measuring both AC and DC currents. Its design enables straightforward integration, making it suitable for industrial, commercial, and communication systems. Common uses include motor control, load monitoring and management, switched-mode power supplies, and protection against overcurrent faults [4]. This module gives a fairly accurate reading when calibrated very well and when operated under

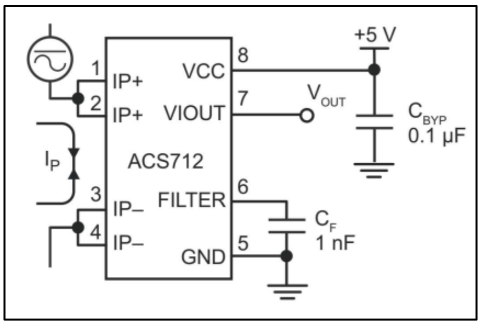


Figure 4 ACS712 Circuit Diagram [4]

In the diagram above, The ACS712 gives an output of an analog signal, V_{out}. which varies linearly with the uni- or bi-directional AC or DC primary current sensed, IP, within the range specified (0-30A in this case). C_F is recommended for noise management, with values that depend on the application [4].

2.4. ZMPT101B Voltage Sensor

This is a high precision AC voltage sensor module that uses ZMPT101B transformer and an adjustable potentiometer for sensitivity tuning. The input AC voltage is transformed into a proportional low-voltage signal (0-5V) for easy readings by microcontrollers and ADCs. The configuration used by the module is shown in Figure 4. The output of the ZMPT101B is connected across an operational amplifier with a sampling resistor in between V+ and the output. The sampling resistor can be adjusted in order to calibrate the sensor [5].

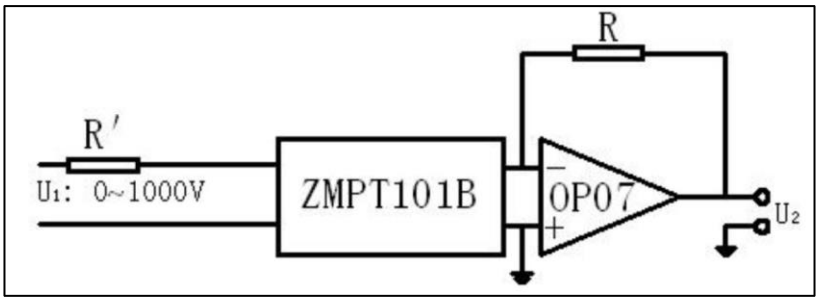


Figure 5 ZMPT101B Module Circuit Diagram [5]

2.4.1. Relay Unit

This is responsible for connecting and disconnecting the load from the network depending on the control signal received from the cloud.

2.4.2. Pin Assignment

Pin assignment is very important while connecting hardware units to a microcontroller because, it is important to connect devices to the appropriate pins. For example, where the hardware is an output device, it should be connected to a pin with ADC. Table 1 below shows the pin assignment of the hardware connected to the ESP32 microcontroller and, Figure 5 shows the complete implementation of the hardware connections to the microcontroller.

Table 1 Pin Assignment

Hardware	**Pin Number**
Relay	D23
Voltage Sensor	D13
Current Sensor	D33
LED For Wi-Fi Status	D21

LED For Data Transfer	D18
LED For Low Credit	D19
Buzzer	D5

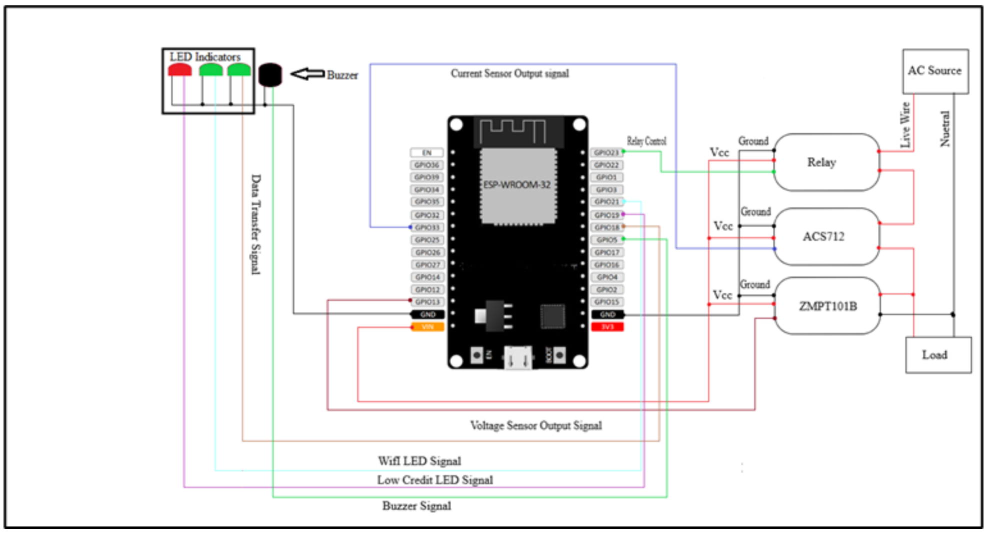


Figure 6 Circuit Diagram

2.5. Firmware Design

Firmware refers to a software embedded into hardware devices. The software is written, compiled and stored in a non-volatile memory of the microcontroller. The software is responsible for controlling the behavior of the hardware devices, manages communication with peripherals, handles network connections, and executes other tasks necessary for the proper functioning of the device. Dividing the firmware program into modules is very important to allow for easy update/changes while the device is in operation [6]. The firmware of this system is represented in Figure 7.

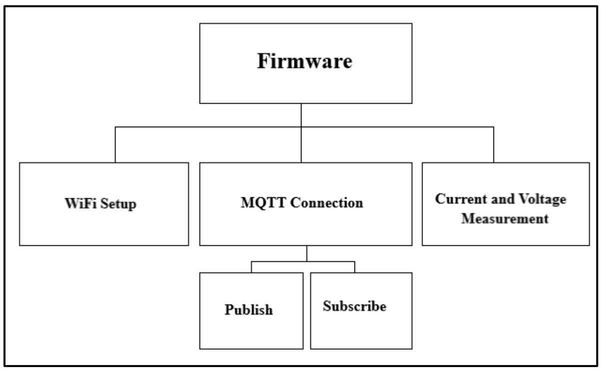


Figure 7 Modular Representation of the System Firmware

2.6. Wi-Fi Setup

This is the section of the firmware responsible for establishing a connection to a WiFi network enabling the smart energy meter to communicate with the Internet. The smart energy meter attempts to connect to a predefined WiFi Network using the SSID and password provided by user during meter configuration.

2.6.1. MQTT Connection

This module facilitates connection between the smart meter and an MQTT broker for both publishing and subscribing to topics. If the connection to the broker is successful, two processes occur at the same time. Publish and subscribe.

Publish process involves reading current and voltage sensor data and encoding it into a JSON format and publishing it to a specified topic on the MQTT broker. The publish process flowchart is shown in Figure 8.

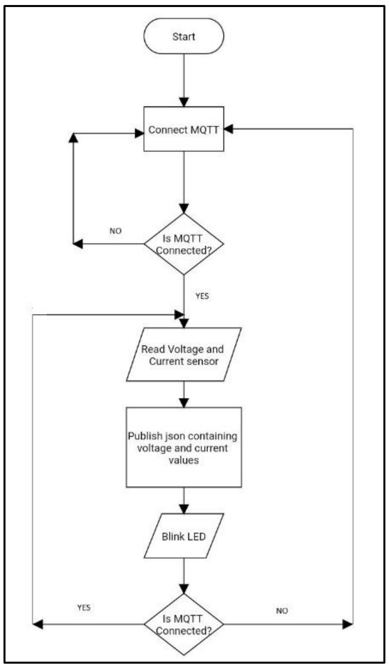

Start
Connect MQTT
Is MQTT Connected?
NO
YES
Read Voltage and Current sensor
Publish json containing voltage and current values
Blink LED
YES
Is MQTT Connected?
NO

Figure 8 Publish Process

The subscribe process is responsible for subscribing to a specific topic on the MQTT broker and receive control values in real time that will be used for controlling the load based on the value generated by a PHP script. Figure 9 shows the process flowchart describing the process. The first stage of the flowchart is to try and connect to the MQTT broker. If the connection is successful, is will get the meter status variable from the MQTT broker in real time. If meter status is 1, the meter will switch off the supply by triggering the relay and the buzzer will start beeping alerting the user that he has low credit. Else if the meter status is 0, the meter will continue its operation and the supply is ON.

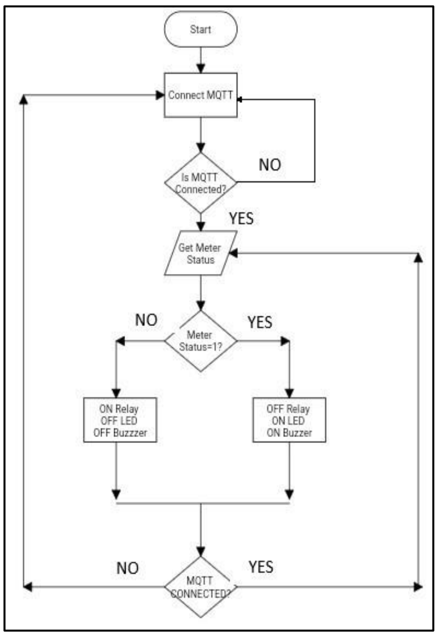


Figure 9 Subscribe Process flowchart

All of the processes listed above are required to run simultaneously and in real time. To manage \the execution of these tasks, there is a requirement to make use of a real time operating system. Complex systems require the use of real time operating system (RTOS) [7]. In figure 10, A shows a basic embedded system that does not have real time requirement. B is a bit more complex as it requires real time requirements. Therefore, real time operating system (RTOS) is required to manage simultaneous task execution.

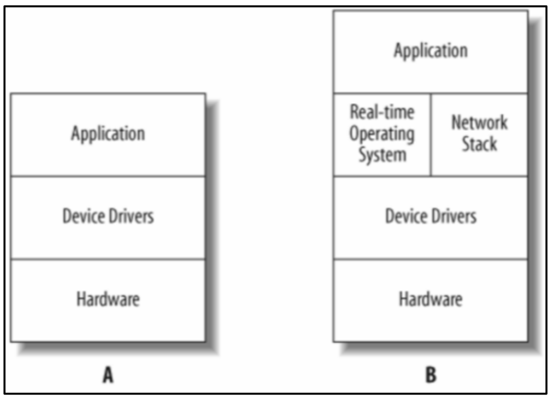


Figure 10 Types of Embedded Systems [7]

2.7. Communication Protocol

The choice of communication protocol as highlighted earlier under firmware design is MQTT communication protocol. The choice is based on the architecture of the MQTT communication protocol (subscribe/publish model) that allows multiple devices to communicate with a single TCP/IP server in ease. Additional features such as authorization, authentication and secured connections are easily integrated into an MQTT connection [8]. Figure shows a basic MQTT communication protocol configuration where multiple devices are connected via an intermediate server known as broker. Each device is a client and can be configured to either publish or subscribe to a topic or do both. The broker is the middle entity between all the clients on the network.

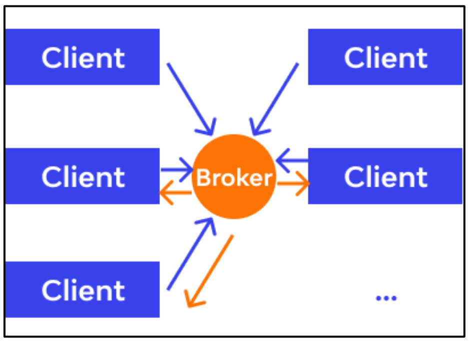


Figure 11 MQTT Architecture [9]

2.8. Meter Management System

This is an essential part of the system responsible for control and data storage. The meter management system is designed using PHP which is server-side programming language used in creating dynamic web pages. Its popularity is in such a way that 75.2% of the entire world web pages are designed using PHP [10]. The PHP web application in this project will focus on basic features that will make the system work just fine. Figure 12 shows the implementation the modular structure of the meter management system.

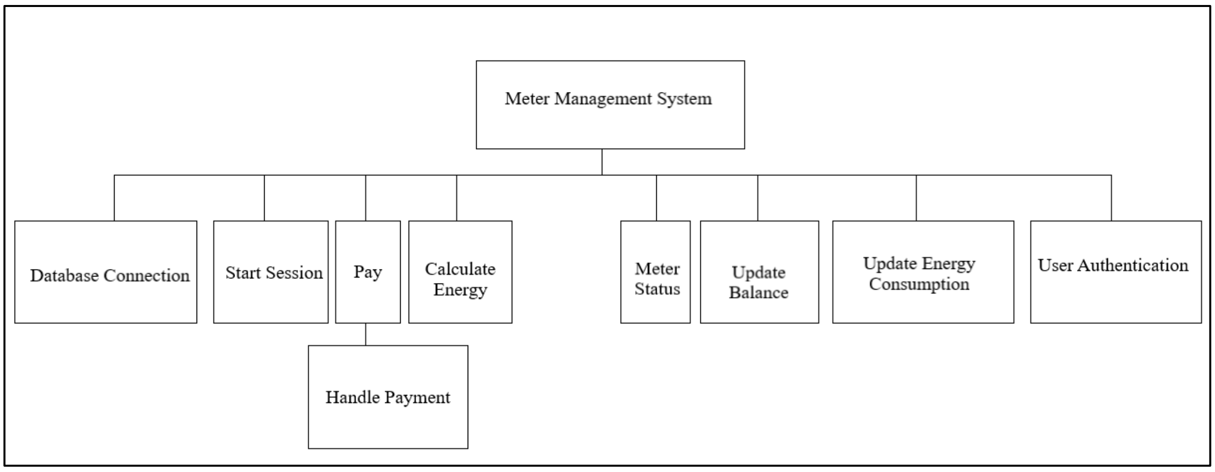


Figure 12 Modular Structure of The Meter Management System

- **Database Connection**: This section of the script facilitates the connection of the application with an existing structured SQL database. This module plays a crucial role in routing and fetching data from the database.
- **Authentication:** This control user access across the meter management system in such a way that only verified costumers, staff, and system administrators are given access to specific information depending on the type of user. The information of each user is personalized based the record of such costumer in the database.
- **Start Session:** This involves starting a session variable to store customer information that can be accessed everywhere in the application using the **$SESSION** associative array. This reduces the need to constantly access the database.
- **Calculate Energy:** This module calculates the total energy consumed over a period by analyzing power data retrieved from the database, computing energy consumption based on power usage and time intervals between data points. To avoid accumulating the energy consumption where the meter is offline, the module makes sure that whenever the intervals between two data points is up to 30 seconds, the accumulation skips the data points with 30 seconds interval or more and continue on the subsequent sets of data points.
- **Update Balance**: This module handles the deduction of balance based on the energy consumed by a user, updating the corresponding database records accordingly. The deduction of balance is done whenever the energy consumed is up to 1kWh. Each time balance is deducted, the total energy and timestamp at the time of deduction is stored in the database.
- **Update Energy Consumption**: This function maintains records of energy consumption for each meter, either updating existing records or inserting new ones, ensuring accurate tracking of energy usage over time.
- **Pay:** This module handles payment reception and processing. The user is prompted to input the intended amount to top up to his account. Upon submission of the form, the same amount is communicated to the payment gateway where the payment reception and processing is done by the payment service provider. Upon completion of the transaction, the payment service provider returns a JSON that contains the status, meter Id, transaction Id, and amount. Upon the reception of these parameters via an API, a PHP script handle the transaction properly by displaying the status of the transaction to the user, adding records to the database and if the transaction is successful, increment the balance pointing to the meter.
- **Meter Status:** This module retrieves the timestamp of the latest entry in the "EnergyData" table and compares it with the current time to determine whether the meter is online or offline. If the time difference between the latest entry and the current time exceeds a predefined threshold (2 minutes), the meter is considered offline. otherwise, it's considered online.

In addition to the modules shown in Figure 12, there is another block of PHP scripts to handle communication between the MQTT broker and the apache web server. These scripts are standard libraries created by the PHP community accessible via Composer. Composer is dependency manager that manages the integration of third party solutions to an existing PHP application [11]. Figure 13 shows the modular structure of the MQTT libraries. Therefore, we need to build on the scripts of the MQTT libraries to handle communication between the meter management system and the MQTT broker

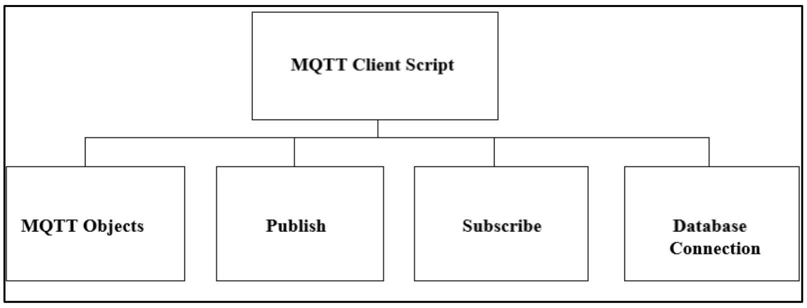


Figure 13 MQTT Integration Section

MQTT Objects: These are standard objects that are included in the MQTT client script and therefore more and more classes can be created using the objects. MQTT functionalities such authentication, security, publishing, and subscription can be derived from these objects.

Publish: This module continuously checks the balance of a meter identified by a specific ID pointing to costumer. If the balance is less than a threshold amount(Price of a unit of electricity), it publishes 1 to an MQTT topic 'meterStatus', indicating a low balance; otherwise, it publishes 0, indicating a normal balance. The value of 'meterStatus' is used for controlling the meter to either keep the supply ON or OFF as explained in the flowchart in Figure 9.

Subscribe: This module subscribes to an MQTT topic named ‘SmartEnergyMeter’. Upon receiving a message, it triggers the ‘procMsg’ function from the MQTT object. Inside ‘procMsg’, it decodes the received JSON message, checks if it contains ‘voltage’ and ‘current’ keys, and inserts these values into a database.

Database Connection: Just like in the previous block, database connection is mandatory since there is a need of saving and retrieving data from the database.

2.9. Database

This is one of the most important part of the meter management system because it is responsible for storing information about consumption. In which, the data stored in the database is retrieved for control purposes and decision making. The energy consumption stored in the database gives full insights into how, where, and when the energy is consumed [12]. The database is designed using SQL which is a relational database. Basic information about customers(users), meter, bills, and energy consumption & tracking constitute the database where all the tables are related through normalization. Figure *14* shows the database schema of the whole system.

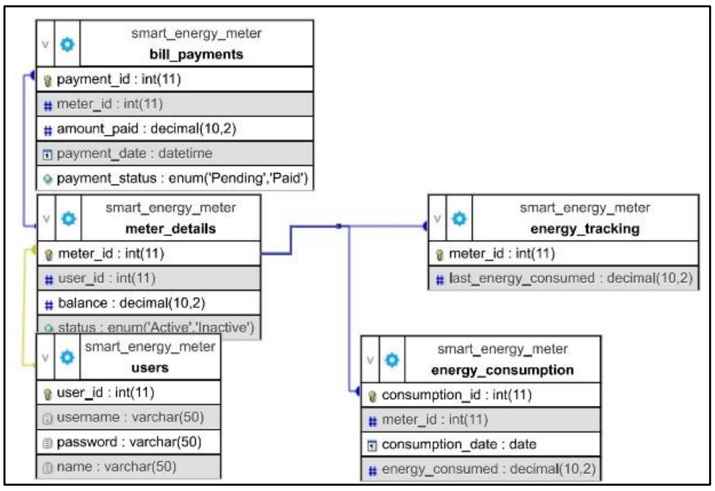


Figure 14 Database Schema

3. Theory and Calculations

3.1. Hardware Implementation

3.1.1. ACS712 Current Sensor

While integrating this sensor to the design, the output is expected to be a low voltage AC signal which represents a scale of the input. A standard C/C++ library was created by [13]. In the library, he tried to create functions that model the sensor's output signal as peak-peak, RMS, and DC current. Our interest is in the use of the function for RMS, because it aligns with our application. One of the first features of the library is to convert the sensors voltage output into current equivalent given by the mathematical equation

$$I_{(mA)} = \frac{V_{out} - V_{midpoint}}{mV_{per\ Amp}}$$

Where:
$Vout$ is the sensor's analog output voltage
$Vmidpoint$ is the midpoint voltage when no current
$mV\ per\ Amp$ is the sensor's sensitivity

The ADC on the microcontroller represents the output of the current sensor based on a number ranging from 0 to the maximum ADC resolution. There is need to convert such number to represent the voltage equivalent of the sensor's output.

$$mV_{per\ step} = \frac{1000 \times V_{ref}}{MAX_{adc\ value}}$$

Where:
V_{ref} is the reference voltage
$maxADC\ value$ is the ADC's maximum value (e.g., 1023 for a 10bit ADC)

The instantaneous measured current can therefore be derived from the sensor's sensitivity since sensitivity is the measure of change in output to input.

$$sensitivity = mV_{per\ amp} = \frac{mV_{per\ step}}{mA_{per\ step}}$$

From the above equation, the output of the sensor can be represented by $mA\ per\ step$

$$mA_{per\ step} = \frac{mV_{per\ step}}{mV_{per\ amp}}$$

For the computation of the RMS value of the measured current, we have to capture some samples of the instantaneous current and use it in the following equation.

$$I_{rms} = \frac{\sum_{n=1}^{n=n} I_n}{n}$$

An object was created in the library and it can be instantiated as class anywhere in the program where its application is needed.

3.1.2. ZMPT101B Voltage Sensor

[14] Used polynomial regression to calculate the estimate RMS voltage of the output of ZMPT101B Voltage sensor and decided on the best method polynomial to use based on comparison between the value calculated using polynomial and the actual measured value using standard FLUKE 115 meter. The work tried to find out the performance of the polynomials from order 1 to order 5 in relation to the accurate value

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \sum x_i^5 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \sum x_i^5 & \sum x_i^6 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \sum x_i^5 & \sum x_i^6 & \sum x_i^7 \\ \sum x_i^3 & \sum x_i^4 & \sum x_i^5 & \sum x_i^6 & \sum x_i^7 & \sum x_i^8 \\ \sum x_i^4 & \sum x_i^5 & \sum x_i^6 & \sum x_i^7 & \sum x_i^8 & \sum x_i^9 \\ \sum x_i^5 & \sum x_i^6 & \sum x_i^7 & \sum x_i^8 & \sum x_i^9 & \sum x_i^{10} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i x_i \\ \sum y_i x_i^2 \\ \sum y_i x_i^3 \\ \sum y_i x_i^4 \\ \sum y_i x_i^5 \end{bmatrix}$$

Figure 15 Polynomial Regression of Nth Order [14]

The first five equations of the fifth order used in the methods are:

$$y_1 = 2.709x - 8.35$$

$$y_2 = 0.0007711x^2 + 2.506x + 0.2662$$

$$y_3 = 0.00000412x^3 + 0.000857x^2 + 2.675x - 3.198$$

$$y_4 = -0.00000004888x^4 + 0.00002986x^3 - 0.005183x^2 + 2.922x - 6.085$$

$$y_5 = 0.0000000001278x^5 + 0.00000003529x^4 + 0.00001023x^3 - 0.003271x^2 + 2.853x - 5.57$$

When the above equations were compared with the measured instantaneous values, the following results were obtained:

Table 2 Voltage measurement using instantaneous voltage calculation [14]

Input voltage fluke meter (V)	**Voltage measurements of the voltage sensor from the polynomial regression equations using the instantaneous calculation.**									
	y_1		y_2		y_3		y_4		y_5	
	Voltage (V)	**Error (%)**	Voltage (V)	**Error (%)**	**Voltage (V)**	**Error (%)**	**Voltage (V)**	**Error (%)**	**Voltage (V)**	**Error (%)**
250	246.48	1.41	229.15	8.340	248.5	0.60	305.39	-22.16	269.55	-7.82
248	247.17	0.33	228.54	7.847	247.67	0.13	301.14	-21.43	268.96	-8.45
246	244.92	0.44	226.14	8.073	245.91	0.04	301.46	-22.54	226.17	8.06
244	244.33	-0.14	234.17	4.029	242.11	0.77	297.93	-22.10	264.68	-8.48
242	241.76	0.10	222.46	8.074	242.11	-0.05	297.93	-23.11	264.68	-9.37
240	240.53	-0.22	220.61	8.079	240.31	-0.13	295.03	-22.93	263.53	-9.80
238	237.38	0.26	217.48	8.622	238.42	-0.18	292.03	-22.70	260.74	-9.55
236	236.80	-0.34	216.69	8.182	236.61	-0.26	289.06	-22.48	257.69	-9.19
234	235.26	-0.54	215.44	7.932	234.76	-0.32	287.44	-22.84	255.41	-9.15
232	231.56	0.19	214.47	7.556	232.65	-0.28	284.41	-22.59	245.48	-5.81
230	230.97	-0.42	213.74	7.070	230.62	-0.27	282.62	-22.88	250.80	-9.04

228	229.34	-0.59	212.44	6.825	228.68	-0.30	278.48	-22.14	247.93	-8.74
226	227.25	-0.55	211.60	6.372	226.58	-0.26	276.78	-22.47	246.40	-9.03
224	225.78	-0.79	209.38	6.527	224.95	-0.42	274.27	-22.44	244.95	-9.35
222	222.91	-0.41	207.76	6.414	222.79	-0.36	270.44	-21.82	243.60	-9.73
220	222.19	-1.00	205.98	6.373	221.42	-0.65	268.41	-22.00	242.51	-10.23
218	220.39	-1.10	202.82	6.963	218.66	-0.30	265.55	-21.81	239.42	-9.83
216	217.29	-0.60	199.68	7.556	217.12	-0.52	262.33	-21.45	235.33	-8.95
214	216.22	-1.04	197.53	7.696	215.83	-0.86	260.25	-21.61	233.99	-9.34
212	213.99	-0.94	196.24	7.434	213.76	-0.83	257.19	-21.32	231.73	-9.31
210	212.10	-1.00	194.42	7.419	211.84	-0.88	255.47	-21.65	230.07	-9.56
208	210.40	-1.15	192.78	7.317	210.33	-1.12	252.35	-21.32	228.32	-9.77
206	208.71	-1.32	191.31	7.131	208.51	-1.22	249.72	-21.22	226.76	-10.08
204	206.42	-1.19	190.42	6.657	206.38	-1.17	247.53	-21.34	224.89	-10.24
202	205.18	-1.57	188.42	6.723	204.47	-1.22	244.42	-21.00	222.58	-10.19
200	204.04	-2.02	186.25	6.875	203.84	-1.92	241.26	-20.63	220.53	-10.27

From the above results, they concluded that the polynomial with the third order, y_3 has the best accuracy given that it has the least percentage error(highlighted in green) compared to the rest of the polynomials whose percentage error are highlighted in red. Therefore, they used the y_3 polynomial in the C firmware to capture instantaneous sample in real time so that the RMS voltage can be calculated using the following relation:

$$V_{rms} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} V_i^2} \quad [14]$$

Where V_i is the instantaneous voltage and n is the number of samples captured by the ADC of the microcontroller at uniform intervals. The flowchart in figure 17 below is used in the process of sensing and recording the voltage being measured.

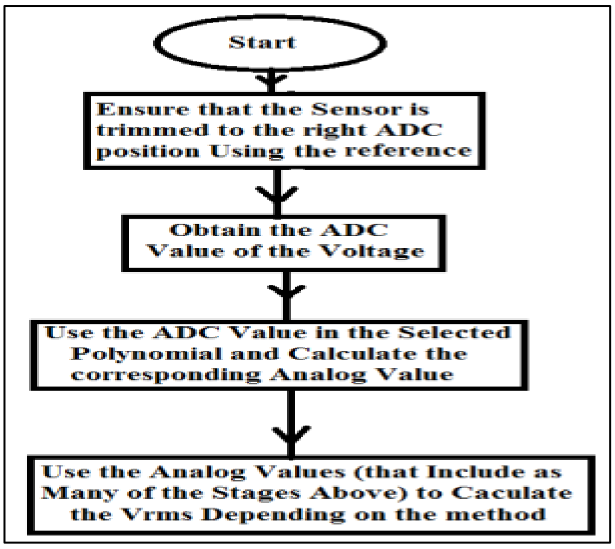

Start
Ensure that the Sensor is trimmed to the right ADC position Using the reference
Obtain the ADC Value of the Voltage
Use the ADC Value in the Selected Polynomial and Calculate the corresponding Analog Value
Use the Analog Values (that Include as Many of the Stages Above) to Caculate the Vrms Depending on the method

Figure 16 The flow chart of voltage calculation [14]

3.1.3. RELAY UNIT

The relay unit is very important because it is responsible for connecting and disconnecting the load. Figure 17 below shows its connection to the whole system. A typical AC/DC relay has 5 terminals which are the VCC, Ground, Control Terminal, Normally Closed (NC) Terminal, and Normally Open (NO) Terminal. In this case, the VCC and Ground terminals are connected to a 5V and ground terminals respectively to energize the relay. While the control terminal is connected to the microcontroller unit. And the whole supply of the system is connected to the normally closed terminal (NC) so that voltage will only be available to the ZMPT101B sensor and the load only if the microcontroller sends a signal to the control terminal of the relay module.

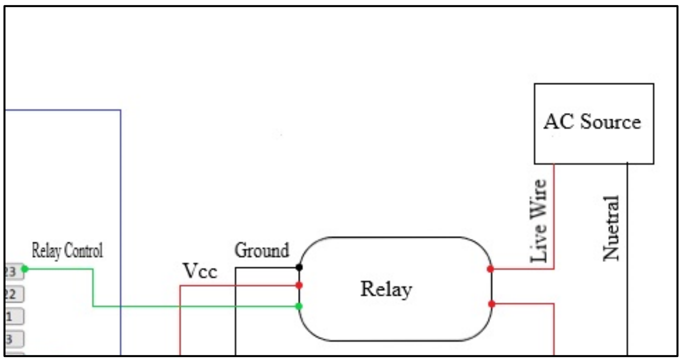


Figure 17 Relay Interfacing

3.2. Firmware Implementation

3.2.1. Real Time Operating System

As highlighted in the design of this system, there is a need for a real time operating system to manage task execution due to the constraint timing requirement of the system. The Real Time Operating System used in this project is FreeRTOS which is lightweight kernel compatible with many microcontrollers including ESP32. FreeRTOS allows us to execute several tasks simultaneously by distributing computing resources equally among all the tasks. Since the ESP32 microcontroller has two cores, we distributed the tasks at hand to the two cores of the microcontroller. The following are the tasks of the whole firmware:

- Sense Task
- Publish Task
- MQTT Task
- MQTT Callback Task
- Wi-Fi Task

Sense Task: This task is responsible for measuring voltage and current values in real time then making those values available to a pointer in the flash memory of the microcontroller.

Publish Task: This task is responsible for making getting the values of voltage and current stored on the flash memory of the microcontroller to a topic on the MQTT broker.

MQTT Task: This task is responsible for handling MQTT connection between the microcontroller and the MQTT broker.

MQTT Callback Task: This task is responsible for processing incoming MQTT messages and updates the meter status.

Wi-Fi Task: This task is responsible for managing Wi-Fi connection and it serves as the communication layer for the microcontroller and the internet.

While implementing FreeRTOS in the firmware, the normal configuration of the firmware is altered because the main loop where code is usually written is empty. This forces the microcontroller to reset due to inactivity in the main loop. A watchdog timer is therefore placed in the main loop to prevent the microcontroller from resetting.

4. Results

From the design and implementation in the previous sections of this paper, we successfully created the smart energy meter composing of two components which are the hardware and software. The most important part of the system is in the transmission of data from the meter to the database server which includes the timestamps of the recorded data. It is very important to note that the novelty of this work lies in the ability of the system transmit the data and record the readings without much latency in the system.

4.1. Hardware

The previous sections of this paper dealt with the hardware design. The design was taken and tested before implementation. The test involved was a unit test for all the hardware components to verify the accuracy of the system and ensure that each component is giving out the expected output or at least close the expected output. The firmware code written in C was also tested alongside some of the hardware components to ensure smooth operation. Figure 18 below shows the hardware prototype.

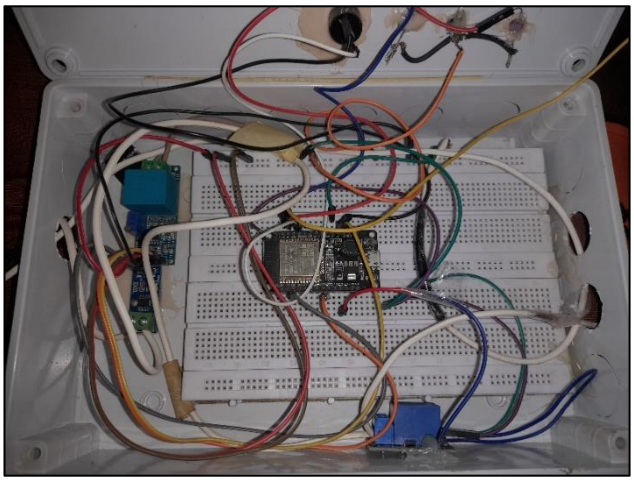


Figure 18 System Prototype

From figure 18 above, the hardware prototype having the ESP32 microcontroller as the heart of the system with other important units like the ACS712 current sensor, ZMPT101B voltage sensor, and the relay unit interconnected by wires in accordance with the design schematic diagram presented in the design section of this paper.

4.2. Software

The software design was implemented based on the design specifications listed in the design section of this paper. Just like the hardware, unit tests were carried on the different modules of the software. The figure below shows the user authentication of the web application where a user is required to input a unique ID dedicated to the user and a password.

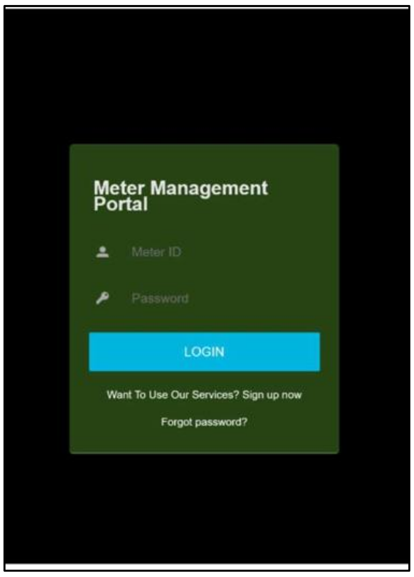
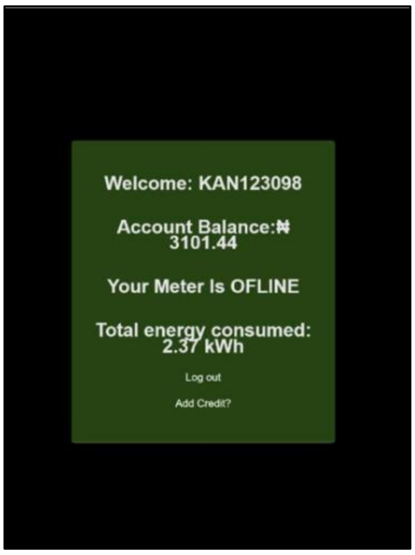


Figure 19 User authentication interface

The next stage after a successful user authentication is the dashboard shown in figure 20. As it can be seen, details of the meter such as the meter ID, the balance of the user in local currency, the status of the meter in real time, and the energy consumption. There is also a log out and add credit buttons which are used to terminate a user session and add money to user's wallet respectively. It should be noted that these details displayed on the database are fetched from the database in real time with the exception of the meter status which is directly controlled by a script monitoring MQTT data transfer in real time.



Welcome: KAN123098

Account Balance:₦ 3101.44

Your Meter Is OFLINE

Total energy consumed: 2.37 kWh

Figure 20 Meter Management System Dashboard

Another important part of the system is the payment page shown in figure 21. In this work, the payment page is just an input form where each number added to the form serves as an increase to the account by the added balance. In deployed system, upon form submission, a request with user's information along the payment amount is sent to a payment service provider via an API where a response is sent by the service provider validating the user. Upon each transaction, the basic details of the transaction such as its status, date, reference number, are recorded in the database. Also, an email is sent the user confirming the state of payment. Upon a successful transaction, a balance of the equivalent transaction amount is added to the user's wallet and if the meter is already beeping, the credit status of meter is changed and supply is restored to the costumer immediately.

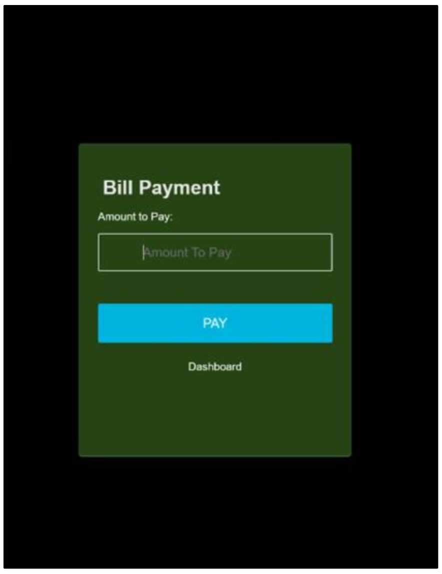
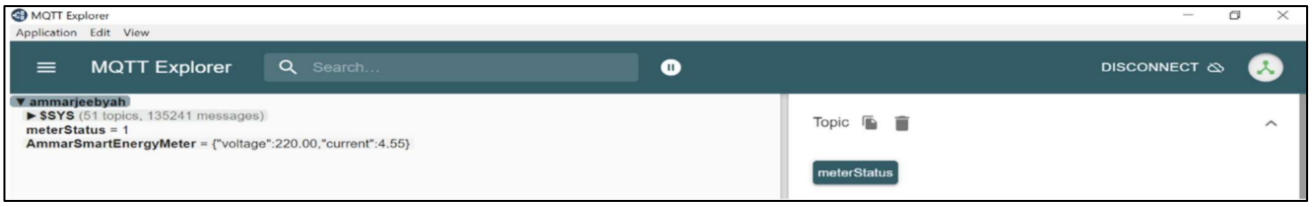


Figure 21 Payment Page

4.3. MQTT Broker

This being the intermediate server that facilitates communication between the meter and apache server. MQTT Explorer was used since it has a user interface. The broker IP address was used as the communication endpoint for both the apache and firmware of the energy meter. A topic was also created for the meter and the apache subscribes to the topic. A JSON data format of voltage and current values are published to a topic. Figure 22 shows the MQTT Explorer user interface and the JSON values of current and voltages transmitted in real time. On the MQTT explorer, you get to the number of message posted on the server and the topics registered on it. You can also access all of the data on the server.



```
▼ ammarjeebyah
  ▶ $SYS (51 topics, 135241 messages)
  meterStatus = 1
  AmmarSmartEnergyMeter = {"voltage":220.00,"current":4.55}
```

Figure 22 MQTT Explorer Interface

5. Discussion

The data transmitted from the smart energy meter and stored in the database gives the opportunity to have full information about the changes in energy consumption over time. This is very important in load forecasting, possible system expansion and operation optimization. The data itself can be used in steering policy changes in accordance with the load specific changes of the demand over a specific period of time.

Few changes to the system on the web application side of the system, advance system control is possible where remote control of a specific costumer's supply is possible. This is very important to costumers because it can enable energy saving and serves as a footprint for smart city

Injecting the data into a narrow trained AI model where the model is trained to solve a specific load problem or system operation optimization moist especially in a power plant.

6. Conclusion

This design can be implemented on all types of power systems whether it being a distributed or an integrated system having too many nodes. Also, additional information data such as frequency and other parameters describing the quality of power at the consumption can be integrated into the system.

The work in this article is just a basic illustration of the function of a high frequency energy meter. More modifications to the different parts of the system is possible in order to improve the functionality of the high frequency energy meter.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] F. Kabir, T. K. Araghi and D. Megías, "Privacy-preserving protocol for high-frequency smart meters using reversible watermarking and Paillier encryption," Computers and Electrical Engineering.

[2] A. Akkad, G. Wills and A. Rezazadeh, "An information security model for an IoT-enabled Smart Grid in the Saudi energy sector," Computers and Electrical Engineering, 2023.

[3] "Espressif," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [Accessed 04 11 2024].

[4] Spark Fun, "Spark Fun," 08 11 2024. [Online]. Available: https://www.sparkfun.com/datasheets//BreakoutBoards/0712.pdf. [Accessed 08 11 2024].

[5] Micro Transformers, "Micro Transformers," 13 11 2024. [Online]. Available: https://www.micro-transformer.com/2ma-2ma-voltage-transformer-ZMPT101B.html. [Accessed 13 11 2024].

[6] H. D. Nguyen, N. Le Sommer and Y. Maheo, "Over-the-Air Firmware Update in LoRaWAN Networks: A New Module-based Approach," in The 21st International Conference on Mobile Systems and Pervasive Computing (MobiSPC), WV, USA, 2024.

[7] M. Barr and A. Massa , Programming Embedded Systems, O'Reilly.

[8] A. Gerodimos , L. Maglaras, M. A. Ferrag, N. Ayres and I. Kantzavelou, "IoT: Communication protocols and security threats," Internet of Things and Cyber-Physical Systems, vol. 3, pp. 1-13, 2023.

[9] wallarm, "wallarm," 28 December 2024. [Online]. Available: https://www.wallarm.com/what/coap-protocol-definition. [Accessed 3 January 2024].

[10] https://w3techs.com/technologies/details/pl-php, "w3techs," [Online]. [Accessed 2 January 2025].

[11] "Composer," [Online]. Available: https://getcomposer.org/doc/00-intro.md. [Accessed 5 January 2025].

[12] T. Fawcett, J. Palmer, N. Terry, B. Boardman and U. Narayan, "Using smart energy meter data to design better policy: Prepayment meter customers, fuel poverty and policy targeting in Great Britain," Energy Research & Social Science, pp. 1-11, 2024.

[13] RobTillaart, "Github," 21 March 2020. [Online]. Available: https://github.com/RobTillaart/ACS712/tree/master. [Accessed 15 July 2024].

[14] A. I., K. N. S., M. W. M., S. Hussain and M. M., "Calibration of ZMPT101B Voltage Sensor Module Using Polynomial Regression for Accurate Load Monitoring," ARPN Journal of Engineering and Applied Sciences, vol. 12, no. 4, pp. 1076-1084, 2017.

[15] Sheetal, Shriraksha, S. Rehman, Vinutha and Sayeesh, "Smart Energy Meter Using IoT," International Journal of Research in Engineering, Science and Management, vol. 3, no. 7, pp. 303-307, 2020.