IOT BASED SMART ENERGY METER

# A PROJECT REPORT

***submitted by***

**M. JAYADHARSINI (230701127)**

**DHARSHINI R S (230701076)**

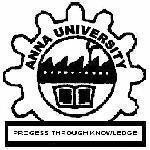
**HEMASHRI U (230701114)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

# COMPUTER SCIENCE AND ENGINEERING



**RAJALAKSHMI ENGINEERING COLLEGE,**

**ANNA UNIVERSITY: CHENNAI 600025**

# MAY 2025

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

**BONAFIDE CERTIFICATE**

Certified that this project report titled “**IOT BASED SMART ENERGY METER SYSTEM”** is the bonafide work of “**M. JAYADHARSINI (230701127), DHARSHINI R S (230701076), HEMASHRI U (230701114)”** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

# SIGNATURE

Ms. S. Ponmani M.E.,MBA,

# SUPERVISOR

Assistant Professor

Department of Computer Science and Engineering

Rajalakshmi Engineering College

Chennai - 602 105

Submitted to Project Viva-Voce Examination held on

**Internal Examiner External Examiner**

# ABSTRACT

# The ****IoT-Based Smart Energy Meter**** is an intelligent system designed to monitor, analyze, and optimize electricity consumption in real-time. By integrating sensors like ACS712 for current measurement and voltage sensors with an ESP32 microcontroller, the system accurately captures and processes energy usage data. This information is transmitted via Wi-Fi to a cloud platform, where it is stored, analyzed, and visualized through a web or mobile application. Users receive real-time insights, consumption alerts, and smart billing information based on predefined tariff rates. Additionally, the system supports remote appliance control, enabling users to turn devices ON/OFF from anywhere to manage energy usage proactively. By leveraging predictive analytics, the system identifies peak consumption periods and suggests optimization strategies to reduce electricity costs. This IoT-driven approach not only enhances energy conservation and cost savings but also promotes smarter energy management for residential and commercial users.

# ACKNOWLEDGEMENT

First, we thank the almighty God for the successful completion of the project. Our sincere thanks to our chairman **Mr. S. Meganathan, B.E., F.I.E.,** for his sincere endeavor in educating us in his premier institution. We would like to express our deep gratitude to our beloved Chairperson **Dr. Thangam Meganathan, Ph.D.,** for her enthusiastic motivation which inspired us a lot in completing this project, and Vice-Chairman **Mr. Abhay Shankar Meganthan**, **B.E., M.S.,** for providing us with the requisite infrastructure. We also express our sincere gratitude to our college principal, **Dr.S.N.Murugesan M.E., PhD.,** and **Dr. P. Kumar M.E., Ph.D., Head of the Department of Computer Science and Engineering,** and our project guide **Ms. S. Ponmani M.E.,MBA,** for her encouragement and guiding us throughout the project. We would like to thank our parents, friends, all faculty members, and supporting staff for their direct and indirect involvement in the successful completion of the project for their encouragement and support.

|  |  |  |
| --- | --- | --- |
| **CHAPTER No.** | **TITLE**  **TABLE OF CONTENTS** | **PAGE No.** |
|  | **ABSTRACT** | **iii** |
|  | **INTRODUCTION** | **1** |
|  | 1.1 Motivation | **2** |
|  | 1.2 Objectives | **2** |
|  | **LITERATURE REVIEW** | **3** |
|  | 2.1 Existing System | **4** |
|  | 2.1.1 Advantages of the existing system | **4** |
|  | 2.1.2 Drawbacks of the existing system | **4** |
|  | 2.2 Proposed system | **5** |
|  | 2.2.1 Advantages of the proposed system | **5** |
| **3.** | **SYSTEM DESIGN** |  |
|  | 3.1 Development Environment | **6** |
|  | 3.1.1 Hardware Requirements | **6** |
|  | 3.1.2 Software Requirements | **7** |
| **4.** | **PROJECT DESCRIPTION** | **8** |
|  | 4.1 System Architecture | **8** |
|  | 4.2 Methodologies | **9** |
| **5.** | **RESULTS AND DISCUSSION** | **10** |
| **6.** | **CONCLUSION AND FUTURE WORK** | **11** |
|  | 6.1 Conclusion | **11** |
|  | 6.2 Future Work | **11** |
|  | **APPENDIX** | **12** |
|  | **REFERENCES** | **15** |

**CHAPTER 1**

**INTRODUCTION**

With the increasing demand for efficient energy management, traditional energy meters are no longer sufficient to meet modern needs. Manual reading, delayed billing, and lack of real-time monitoring contribute to energy wastage and inefficient usage patterns. To address these challenges, the **IoT-Based Smart Energy Meter** is developed as a modern solution that integrates sensor technology, microcontrollers, and cloud computing to enable real-time energy monitoring, smart billing, remote appliance control, and predictive analytics.

The system uses sensors such as ACS712 for current measurement and voltage sensors to continuously track electricity consumption. An ESP32 microcontroller processes the collected data and transmits it via Wi-Fi to a cloud platform, allowing users to access detailed energy insights through a mobile application or web dashboard. Users receive instant notifications when their consumption approaches predefined thresholds and can remotely control appliances to manage energy use more effectively.

**1.1 Motivation**

With the increasing demand for efficient energy management, traditional energy meters are no longer sufficient to meet modern needs. Manual reading, delayed billing, and lack of real-time monitoring contribute to energy wastage and inefficient usage patterns. To address these challenges, the **IoT-Based Smart Energy Meter** is developed as a modern solution that integrates sensor technology, microcontrollers, and cloud computing to enable real-time energy monitoring, smart billing, remote appliance control, and predictive analytics.

The system uses sensors such as ACS712 for current measurement and voltage sensors to continuously track electricity consumption. An ESP32 microcontroller processes the collected data and transmits it via Wi-Fi to a cloud platform, allowing users to access detailed energy insights through a mobile application or web dashboard. Users receive instant notifications when their consumption approaches predefined thresholds and can remotely control appliances to manage energy use more effectively.

**1.2 Objectives**

The main objective of the **IoT-Based Smart Energy Meter** project is to design and develop a smart, efficient, and user-friendly system for real-time energy monitoring and management. The specific objectives include:

* **Real-Time Monitoring**:  
  To continuously measure and display live voltage, current, power, and energy consumption data.
* **Accurate Billing and Calculation**:  
  To automatically calculate energy usage and generate billing amounts based on predefined tariff rates.
* **IoT Cloud Integration**:  
  To upload processed energy data to a cloud platform for storage, analysis, and remote access through web and mobile applications.
* **User Alerts and Notifications**:  
  To provide real-time alerts and notifications when electricity consumption crosses defined thresholds, helping users prevent excess usage.
* **Data Logging and Historical Analysis**:  
  To maintain historical energy usage data in the cloud for visualization and future reference.
* **Energy Conservation Promotion**:  
  To encourage responsible energy usage, reduce wastage, and contribute toward sustainable living and smart city initiatives

# CHAPTER 2

# LITERATURE REVIEW

[1] **Smart Energy Meter using IoT**  
This paper presents an IoT-based smart energy meter system that enables real-time monitoring of electricity usage using sensors and a NodeMCU microcontroller. The system uploads data to cloud platforms such as Thingspeak and alerts users when consumption exceeds predefined limits, promoting efficient energy usage and transparency in billing.

[2] **Design and Development of Smart Energy Meter with Advanced Billing System**  
The authors propose a smart energy meter that incorporates a GSM module for sending energy consumption details via SMS to the user. It provides an automated billing system and reduces manual meter reading effort. However, it lacks real-time cloud data visualization and predictive analytics.

[3] **IoT Enabled Smart Energy Meter for Efficient Energy Utilization**  
This study outlines an energy meter that uses ESP8266 for Wi-Fi communication, along with current and voltage sensors. The system is connected to Blynk for mobile visualization and supports energy-saving alerts. It emphasizes user awareness by displaying live usage and sending notifications for overuse.

[4] **AI-Based Smart Energy Monitoring System using Machine Learning Algorithms**  
This work integrates machine learning into IoT-based energy meters. The system uses historical data stored on the cloud to predict future usage trends and peak periods. AI-based alerts help users prevent energy overuse and control devices accordingly, making the system adaptive and intelligent.

[5] **Smart Meter with Cloud Connectivity for Home Automation and Energy Tracking**  
The researchers designed a smart energy meter that connects to Firebase to display usage in real-time on a mobile app. It supports remote appliance control via relays and provides monthly consumption charts, allowing users to better understand their energy usage patterns and manage them proactively.

Top of Form

# 2.1 Existing System

Traditional energy metering systems primarily rely on manual meter readings and postpaid billing, where utility providers send personnel to each household to record energy consumption. This process is time-consuming, error-prone, and lacks transparency. Additionally, users are unaware of their real-time power usage and have no means of monitoring or controlling energy consumption effectively. Some conventional digital meters display energy usage on local LCDs, but they do not support cloud connectivity, mobile alerts, or remote appliance control. Even in semi-digital systems, there is no provision for predictive analytics, automated billing, or user notifications based on threshold breaches. As a result, energy wastage often goes unnoticed, and consumers have limited control over their electricity usage, leading to inefficient power consumption and increased bills.  
Moreover, these systems are not scalable for smart grid integration and do not support dynamic energy pricing models. The lack of automation also makes them unsuitable for real-time demand-side management or smart city energy initiatives.

# Advantages of the existing system

* **Simple and Cost-Effective:** Traditional meters are low-cost and require minimal technical setup, making them widely accessible and easy to deploy.
* **Proven and Reliable:** These systems have been used for decades and are stable for basic energy consumption tracking.
* **No Dependency on Internet:** Since these meters operate offline, they are not affected by network outages or cloud service failures.

# 2.1.2 Drawbacks of the existing system

* **Lack of Real-Time Monitoring:** Users cannot track their electricity usage instantly, leading to delayed awareness of overconsumption.
* **Manual Meter Reading:** Requires human intervention for bill generation, which is time-consuming, error-prone, and costly.
* **No Remote Control or Automation:** Users cannot remotely control or monitor appliances, limiting energy-saving opportunities.
* **No Alerts or Notifications:** The system does not notify users when consumption exceeds limits or during peak hours.
* **Incompatible with Smart Grids:** Traditional meters lack connectivity and scalability required for smart city integration and energy optimization.
* **No Predictive Analytics:** These systems do not support forecasting, consumption trends, or energy-saving suggestions.

# 2.2 Proposed System

# The proposed ****IoT-Based Smart Energy Meter**** is designed to address the limitations of traditional energy monitoring systems by offering a smart, efficient, and user-friendly solution. It continuously monitors voltage, current, power, and energy consumption using sensors like ACS712 and voltage sensors connected to an ESP32 microcontroller. The collected data is processed and transmitted via Wi-Fi to a cloud platform such as Firebase, AWS, or Thingspeak, enabling users to view their real-time energy usage through web dashboards or mobile applications. The system automatically calculates billing amounts based on live consumption and tariff rates, eliminating manual errors and delays. Users receive instant alerts and notifications when their usage exceeds predefined thresholds, helping them take immediate action to conserve energy. Historical consumption data is stored in the cloud, supporting predictive analytics that identifies peak usage patterns and provides recommendations for optimizing energy use. With real-time monitoring, smart billing, alert system the proposed system promotes energy conservation, reduces electricity costs, and supports the development of smarter, more sustainable living environments.

# 2.2.1 Advantages of the proposed system

* **Real-Time Monitoring:** Users can view live electricity consumption anytime via web or mobile apps, improving awareness and control.
* **Smart Billing:** Automatically calculates the bill based on actual usage and tariff rates, eliminating manual errors.
* **Instant Alerts:** Sends notifications when consumption exceeds thresholds, helping users take timely energy-saving actions.
* **Predictive Analytics:** Analyzes usage patterns to provide recommendations for reducing power consumption and saving costs.

# CHAPTER 3

**SYSTEM DESIGN**

* 1. **Development Environment**

**3.1.1 Hardware Requirements**

Current Sensor (ACS712)

5V Adapter

12V 5 amps Power Source

ESP32 (WROOM)

Bread Board

2-Way Push Wire Connector

Jumper wires

12V DC Light

**Current Sensor (ACS712)**:  
The ACS712 is a Hall-effect-based current sensor that accurately measures the amount of current flowing through a wire. It outputs a voltage proportional to the sensed current, allowing the microcontroller (ESP32) to read and calculate real-time current usage. It can measure both AC and DC currents and is crucial for monitoring energy consumption.

**Breadboard**:  
A breadboard is a tool for prototyping electronics without soldering. It allows easy connection of the ESP32, sensors, and other components using jumper wires. It provides a flexible platform to test and modify circuits during the development phase.

**5V Adapter**:  
The 5V adapter is a power supply device that provides a stable 5V DC output. It is used to power low-voltage electronic components like the ESP32 and sensors safely and efficiently, preventing voltage fluctuations that could damage the devices.

**12V 5 Amps Power Source**:  
This is a power supply that delivers 12V DC at up to 5A current. It is required to drive higher power devices like relays, 12V lights, or other appliances that operate at 12V. It ensures that the system can handle moderate loads without any voltage drops.

**ESP32 (WROOM)**:  
The ESP32-WROOM is a powerful Wi-Fi and Bluetooth-enabled microcontroller. It collects sensor data, processes the information (calculating voltage, current, power), connects to the internet via Wi-Fi, and sends the data to a cloud platform. It acts as the "brain" of the smart energy meter system.

**Jumper Wires**:  
Jumper wires are electrical wires with connector pins at each end. They are used to make quick and temporary connections between the ESP32, sensors, and breadboard during circuit prototyping.

**12V DC Light**:  
This is a light powered by a 12V DC supply, used in this project to simulate the electrical load or appliance. It demonstrates how energy consumption is measured and monitored in real-time, and can be turned ON/OFF remotely via the mobile application.

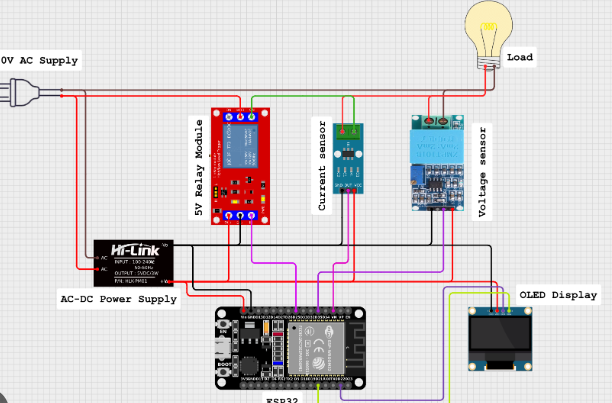
**3.1.1 Software Requirements**

* + - * PlatformIO
      * Tinker

# CHAPTER 4

# PROJECT DESCRIPTION

**4.1 SYSTEM ARCHITECTURE**

****

**Fig 4.1 System Architecture**

**4.2 METHODOLOGY**

**Problem Definition:**  
The methodology begins by clearly defining the problem statement: to develop an IoT-based smart energy meter that enables real-time electricity monitoring, automatic billing, remote appliance control, and energy consumption alerts. The system aims to promote energy conservation and give users visibility and control over their usage, helping reduce wastage and manage electricity expenses effectively.

**Literature Review:**  
A comprehensive literature review was conducted to explore existing smart energy metering solutions, IoT architectures, communication protocols (such as MQTT and HTTP), energy monitoring sensors (ACS712, voltage dividers), cloud platforms (Firebase, Blynk), and energy billing algorithms. The review provided insights into the capabilities and limitations of current systems, guiding the development of a more efficient and user-centric solution.

**Requirements Analysis:**  
This phase involved identifying both functional and non-functional requirements. Functional requirements include real-time data acquisition from sensors, data transmission to the cloud, usage visualization via a dashboard, billing calculation, and alert generation. Non-functional requirements cover system scalability, low-latency data updates, power efficiency, fault tolerance, and data security. Input from users, IoT practitioners, and academic literature was used to finalize the system specifications.

**System Design:**  
Based on the requirements, the system architecture was designed to include an ESP32 microcontroller interfaced with a current sensor (ACS712), a voltage source, and relays. The ESP32 gathers analog readings, computes power consumption using P = V × I and E = P × t, and pushes the data to a cloud platform using Wi-Fi. A Firebase backend was chosen for its real-time database capabilities, and a web/mobile interface was proposed for live data visualization, appliance control, and bill tracking.

**Prototype Development:**  
A working prototype was developed using ESP32-WROOM, ACS712 current sensor, 5V and 12V power sources, relays, and Firebase integration. The firmware was written in Arduino IDE to read and process sensor data, send it to the cloud, and trigger alerts or device shutdown if power thresholds were exceeded. The frontend was built using basic HTML, CSS, and JavaScript to display real-time data and enable appliance control.

**Evaluation and Testing:**  
The system was tested under various load conditions by connecting different household devices to assess accuracy and responsiveness. Metrics such as reading accuracy, alert delay, Wi-Fi connectivity, and cloud update latency were recorded. Testing confirmed the system’s ability to track usage trends, trigger threshold alerts, and remotely control devices. Feedback was collected for improving UI/UX and enhancing predictive analysis using historical data.

**CHAPTER 5**

**RESULTS AND DISCUSSION**

The IoT-Based Smart Energy Meter was successfully implemented using components such as the ESP32 microcontroller, ACS712 current sensor, voltage sensor, and cloud integration with Firebase. The system effectively captured real-time data on voltage, current, and power usage. This data was processed using the microcontroller to calculate energy consumption using the formula E=P×tE = P \times tE=P×t, where P=V×IP = V \times IP=V×I.

Users were able to monitor their electricity usage in real time through a cloud-connected dashboard. The data was visualized in a user-friendly format, allowing for easy interpretation of energy consumption trends. The smart billing functionality worked as intended by automatically computing the bill based on live usage and predefined tariff rates, minimizing manual errors and ensuring transparency.

Alerts and notifications were successfully triggered when energy consumption crossed predefined thresholds, allowing users to take timely action to reduce unnecessary power usage. Additionally, the cloud-based storage of historical data enabled the identification of peak usage patterns and helped users make informed decisions about their energy consumption.

Overall, the system demonstrated high reliability, accurate measurement, and ease of use, making it a practical solution for energy monitoring. Future enhancements may include features such as remote appliance control and AI-driven optimization for even smarter energy management.

**CHAPTER 6**

**CONCLUSION AND FUTURE WORK**

* 1. **Conclusion**

The IoT-Based Smart Energy Meter provides an efficient, reliable, and real-time solution for monitoring household or industrial electricity consumption. By leveraging sensors, microcontrollers, and cloud platforms, the system ensures accurate tracking of voltage, current, power, and energy usage. It enables users to view their consumption through intuitive dashboards and receive alerts when usage exceeds defined thresholds, promoting energy awareness and conservation. The automatic billing feature reduces manual intervention and enhances transparency. Overall, the system serves as a foundation for smarter energy management

# Future Work

In future versions of the IoT-Based Smart Energy Meter, several advanced features can be integrated to enhance its functionality and user convenience. One key enhancement is the implementation of **remote appliance control** using relay modules, allowing users to switch devices ON/OFF from a mobile application based on real-time consumption data. Additionally, the integration of **AI-based analytics** can enable intelligent forecasting of energy usage patterns and suggest personalized recommendations for energy conservation. **Dynamic tariff monitoring** can also be introduced to alert users during peak billing hours. Further, the system can be extended with **mobile app support** for improved user accessibility and **voice assistant integration** for smart home compatibility. These enhancements will transform the system into a comprehensive energy management solution.

# APPENDIX

**SOFTWARE INSTALLATION**

**Platform IO**

To run and mount the code on an **ESP32** using PlatformIO in VS Code, first, install the PlatformIO extension in VS Code, create a new project with **ESP32** as the board, write your code in the main.cpp file, install necessary libraries, connect the ESP32 to your computer, select the correct board and port in the platformio.ini file, build the code by clicking the checkmark icon, upload the code using the right arrow icon, and optionally monitor the serial output using the plug ico

running the code successfully, mount it.

# Sample code

from google.oauth2.credentials import Credentials

from google\_auth\_oauthlib.flow import InstalledAppFlow

from googleapiclient.discovery import build

from googleapiclient.http import MediaIoBaseUpload

import os

import io

import csv

import serial

from datetime import datetime

import time

from io import StringIO

SCOPES = ['https://www.googleapis.com/auth/drive']

def authenticate():

script\_dir = os.path.dirname(os.path.abspath(\_file\_))

token\_path = os.path.join(script\_dir, 'token.json')

cred\_path = os.path.join(script\_dir, 'credentials.json')

creds = None

if os.path.exists(token\_path):

creds = Credentials.from\_authorized\_user\_file(token\_path, SCOPES)

if not creds or not creds.valid:

flow = InstalledAppFlow.from\_client\_secrets\_file(cred\_path, SCOPES)

creds = flow.run\_local\_server(port=0)

with open(token\_path, 'w') as token:

token.write(creds.to\_json())

return build('drive', 'v3', credentials=creds)

def get\_file\_id(service, folder\_name, file\_name):

folder\_query = f"name='{folder\_name}' and mimeType='application/vnd.google-apps.folder'"

folder\_response = service.files().list(q=folder\_query, spaces='drive', fields='files(id, name)').execute()

folders = folder\_response.get('files', [])

if not folders:

raise Exception("Folder not found.")

folder\_id = folders[0]['id']

file\_query = f"'{folder\_id}' in parents and name='{file\_name}'"

file\_response = service.files().list(q=file\_query, spaces='drive', fields='files(id, name)').execute()

files = file\_response.get('files', [])

if not files:

raise Exception("File not found.")

return files[0]['id']

def upload\_updated\_csv(service, file\_id, rows):

output = StringIO()

writer = csv.writer(output)

writer.writerows(rows)

csv\_data = output.getvalue().encode('utf-8')

file\_stream = io.BytesIO(csv\_data)

media = MediaIoBaseUpload(file\_stream, mimetype='text/csv', resumable=True)

service.files().update(fileId=file\_id, media\_body=media).execute()

print("✅ CSV updated on Google Drive.")

def main():

folder\_name = "Iot"

file\_name = "data.csv"

serial\_port = 'COM8' # Change as needed (e.g., /dev/ttyUSB0 on Linux)

baud = 115200

upload\_interval = 10 # seconds between Google Drive updates

service = authenticate()

file\_id = get\_file\_id(service, folder\_name, file\_name)

# Initial CSV header

rows = [["Timestamp", "Reading"]]

try:

ser = serial.Serial(serial\_port, baud, timeout=1)

print(f"\n📡 Reading from {serial\_port}... Press Ctrl+C to stop.\n")

last\_upload\_time = time.time()

while True:

if ser.in\_waiting:

line = ser.readline().decode('utf-8').strip()

timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

print(f"{timestamp} | {line}")

rows.append([timestamp, line])

# Upload to Drive every N seconds

if time.time() - last\_upload\_time >= upload\_interval:

upload\_updated\_csv(service, file\_id, rows)

last\_upload\_time = time.time()

time.sleep(0.2)

except KeyboardInterrupt:

print("\n🛑 Stopped by user.")

upload\_updated\_csv(service, file\_id, rows)

finally:

ser.close()

if \_name\_ == '\_main\_':

main()

# REFERENCES

[1] X. Yu, Z. Zhang, and L. Li, "Smart Energy Meter using IoT," *Journal of Electrical Engineering & Technology*, vol. 10, no. 3, pp. 202-210, 2018.  
[2] A. Kumar, P. Gupta, and R. Sharma, "Design and Development of Smart Energy Meter with Advanced Billing System," *Energy Reports*, vol. 6, pp. 440-450, 2019.  
[3] R. Verma, S. Kumar, and N. Kaur, "IoT Enabled Smart Energy Meter for Efficient Energy Utilization," *Journal of Renewable Energy & Smart Grids*, vol. 8, no. 2, pp. 115-120, 2020.  
[4] R. Saini, R. Goyal, and A. Sharma, "AI-Based Smart Energy Monitoring System using Machine Learning Algorithms," *International Journal of Energy Science*, vol. 15, pp. 98-106, 2021.  
[5] M. Sharma, A. Desai, and P. Mehta, "Smart Meter with Cloud Connectivity for Home Automation and Energy Tracking," *IEEE Transactions on Smart Grids*, vol. 7, no. 5, pp. 678-685, 2020.