International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Real - Time IoT - Based Energy and Power Monitoring and Management System for Small - Scale Applications Using a Raspberry Pi Web Interface

Abhinav Mishra[1], Dr. Jyoti Srivastav[2], Dr Manish Srivastav[3]

[1]M. Tech. Electrical Engineering (Power System), Department of Electrical Engineering, Sam Higginbottom University of Agriculture, Technology and Sciences

[2]Assistant Professor, Phd Electrical Engineering, Department of Electrical Engineering, Sam Higginbottom University of Agriculture, Technology and Sciences

[3]Associate Professor, Phd Electrical Engineering, Department of Electrical Engineering, Sam Higginbottom University of Agriculture, Technology and Sciences

Abstract: ***This paper presents a comprehensive design of an electrical energy monitoring system utilizing the PZEM - 004T energy sensor and Raspberry Pi 4. The system includes an automated metering device, a PZEM module, and a buzzer, sending real - time energy consumption data to a cloud server for display on responsive web interfaces accessible via desktop and mobile devices. The system's accuracy is validated with an average error of less than 2%. Additionally, an Energy and Power Monitoring & Management System (EPMMS) is redesigned for seamless integration with the Internet of Things (IoT) ecosystem, targeting smart industries and commercial buildings. The EPMMS, implemented on a Raspberry Pi, manages power flows between various sources and the grid, employing Artificial Intelligence and Dynamic Programming to optimize energy usage and reduce costs. Results from a test bench simulating multiple environments demonstrate the system's effectiveness, with a 2.88% uncertainty in power exchange and a potential 3.23% reduction in electricity bills. This digitized, cost - effective, and portable energy monitoring system is poised for rapid market adoption, facilitating energy conservation across residential, commercial, and industrial sectors.***

Keywords: EMS, PI, NODEJS, Rasberry Pi, EPMS, CO2, Carbon foot print

1. Introduction

Electric energy utilization is critical across various sectors, including industrial, commercial, and agricultural applications. In 2012, global electricity consumption was estimated at 20, 900 Terawatt - hours (TWh), and the demand has been rising steadily. By 2018, global electricity demand increased by 4%, marking the highest growth since 2010. Despite the rise of renewable energy sources, like nuclear power, coal and internal combustion plants still contribute significantly to CO2 emissions.

In India, electricity generation surged from 154.7 GW to an impressive 345.5 GW in 2018, positioning the country as the third - largest electricity producer globally, following China and the United States. However, India continues to face substantial power demand challenges. The International Energy Agency projects that India's electricity demand will triple between 2018 and 2040. In 2018 alone, electricity demand in India increased by 65 TWh, primarily driven by building cooling needs.

This project aims to reduce power usage through efficient dynamic power management using the Internet of Things (IoT). As industrial and commercial building automation systems evolve rapidly with advancements in electronics and information technology, smart systems for monitoring and controlling energy consumption are becoming increasingly important. In Indonesia, electricity consumption has been rising yearly due to inefficient usage and lack of consumer awareness about energy conservation. Conventional electrical measuring devices, often requiring physical presence at the installation site to retrieve data, are becoming outdated and inefficient.

Various studies have explored monitoring electric power systems, including using wireless sensor networks and Arduino - based systems to track energy consumption. However, these methods have limitations, such as the need for physical proximity to the sensors and the inefficiency of current and voltage sensors like ACS712 compared to the more precise PZEM - 004T sensor.

This research proposes a Raspberry Pi - based monitoring system using the PZEM - 004T sensor for measuring voltage, current, and power. The system aims to provide real - time energy usage data to consumers, facilitating better energy management in industrial and commercial buildings. The integration of distributed renewable energy sources (RES) into the power grid presents challenges due to their unpredictable nature. Effective Energy Management Systems (EPMMS) are essential for optimizing power flows and ensuring grid stability.

EPMMSs enable consumers to actively participate in the energy market while supporting grid operators with optimized decision - making tools. For widespread adoption, EPMMSs must integrate seamlessly with existing devices, require minimal setup, and operate efficiently within the IoT

framework. This paper focuses on redesigning an EPMMS to reduce computational complexity, implement it on a Raspberry Pi, and align it with IoT standards. The re - implemented EPMMS uses Dynamic Programming (DP) instead of Mixed Integer Linear Programming (MILP) for optimization, enhancing feasibility and effectiveness. Experimental validation involved a network of wireless sensors in a simulated grid of four smart buildings, demonstrating the system's computational efficiency and potential for reducing electricity bills and power exchange uncertainty.

This study contributes to the ongoing efforts in developing cost - effective, real - time energy monitoring systems. By leveraging IoT and advanced algorithms, the proposed EPMMS offers a practical solution for efficient energy management in industrial, commercial, and residential settings, paving the way for broader adoption of smart energy systems. Real - time data analysis significantly enhances the effectiveness of energy management systems through several key mechanisms:

Immediate Insights and Adjustments: Real - time data allows for instantaneous insights into energy consumption patterns, enabling prompt adjustments to optimize energy usage and reduce wastage.

Enhanced Predictive Maintenance: Continuous monitoring and analysis of real - time data help in predicting equipment failures and scheduling maintenance proactively. This prevents unexpected breakdowns and ensures equipment operates efficiently.

Dynamic Load Management: Real - time analysis enables dynamic management of electrical loads, adjusting energy distribution based on current demand and supply conditions. This ensures optimal use of available energy resources.

Improved Decision - Making: Access to up - to - the - minute data supports better decision - making by providing accurate and timely information about energy usage. This helps in formulating effective energy policies and strategies.
Increased Responsiveness: Real - time data analysis enhances the system's ability to respond to changing conditions quickly. This includes adjusting to fluctuations in energy supply from renewable sources and adapting to varying demand patterns.

Cost Savings: By optimizing energy consumption in real - time, businesses can reduce their energy costs significantly. This is achieved through better load management, reduced peak demand charges, and improved operational efficiency.

Regulatory Compliance: Real - time monitoring and analysis help in ensuring compliance with energy regulations and standards. It provides the necessary data to demonstrate adherence to regulatory requirements and supports sustainability initiatives.

User Engagement and Awareness: Real - time data visualization tools enhance user engagement by making energy consumption data easily accessible and understandable. This promotes energy - saving behaviors and increases awareness about energy efficiency.

Overall, real - time data analysis is a critical component in modern energy management systems, driving efficiency, reliability, and sustainability in energy usage.

2. Literature Review on Energy Management Systems

Energy management systems (EMS) have become a crucial component in addressing the global energy crisis, promoting sustainability, and improving efficiency across various sectors. This literature review synthesizes key findings from a collection of studies on EMS, highlighting advancements, methodologies, and diverse applications of these systems The paper "Implementing Energy Management System to Increase Energy Efficiency in Manufacturing Companies" (2015) emphasizes the importance of energy efficiency in manufacturing due to rising costs of non - renewable energy. It presents a methodology for systematically implementing energy management systems, highlighting the Plan - Do - Check - Act (PDCA) cycle. This approach is essential for continuous improvement and sustainable production in manufacturing companies. “According to Design and Development of Real - Time Small - Scale IoT - Based Energy Monitoring System" (2021) discusses the development of an IoT - based EMS for real - time energy monitoring. The system aims to promote energy conservation by providing users with continuous feedback on their energy usage, which is crucial for fostering energy - efficient behavior. Design of Raspberry Pi Web - Based Energy Monitoring System for Residential Electricity Consumption" (2022) presents a web - based EMS using Raspberry Pi and IoT technology. The system tracks real - time energy usage and provides users with detailed insights into their consumption patterns, helping them to identify opportunities for energy savings. Several studies highlight the significant role of IoT and data analytics in enhancing the capabilities of EMS. The integration of IoT allows for real - time data collection and monitoring, which is crucial for accurate energy management. For instance, a study demonstrated the use of smart metering techniques to provide accurate readings and un - tampered data for monitoring energy consumption of each appliance. This approach helps consumers visualize, monitor, and optimize their energy usage based on data analytics [1].

Another study showed the effectiveness of integrating IoT with EMS to create a more reliable system compared to conventional methods. Data collection is automated via Wi - Fi, which updates in short intervals, eliminating the need for manual data collection [2]. An Arduino esp8266 microcontroller was programmed to perform energy management tasks, sending details to the consumer's mobile through IoT and GSM modules, and displaying information on an LCD screen [3].

A proposed energy monitoring system provided accurate energy consumption values and identified damaged loads quickly, allowing users to control their loads efficiently [4]. An IoT - enabled smart energy meter was developed with real - time load control capabilities and a mobile application for visualizing energy usage and generating tariffs, enabling remote control of appliances [5].

The integration of IoT and blockchain technologies was proposed for monitoring household energy consumption and controlling unnecessary energy loss. A smart meter monitored energy consumption, and an Android application allowed users to set limits and receive alerts when nearing thresholds. Data from the application and smart meter were stored in a blockchain for comparison, and devices were turned off or switched to power - saving mode if limits were exceeded [9].

3. Proposed System

This research proposes an IoT - based Energy Power Monitoring and Management System (EPMMS) utilizing the Raspberry Pi and PZEM - 004T energy sensor. The system is designed to monitor and manage energy consumption in real - time, providing a comprehensive and cost - effective solution for small - scale industries. IoT enhances energy management by enabling real - time monitoring, predictive maintenance, automation, and data - driven insights. It facilitates efficient energy use, reduces downtime, and integrates renewable energy sources. Dynamic Programming (DP) optimizes energy management by solving complex problems efficiently, adapting to real - time data, and enhancing decision - making. DP improves scalability, reliability, and cost - effectiveness in energy systems. Predictive maintenance uses data analysis to predict equipment failures, optimizing maintenance schedules and reducing downtime. This ensures machines operate efficiently, conserving energy. IoT and machine learning synergize to enhance energy efficiency in smart buildings through several key mechanisms:

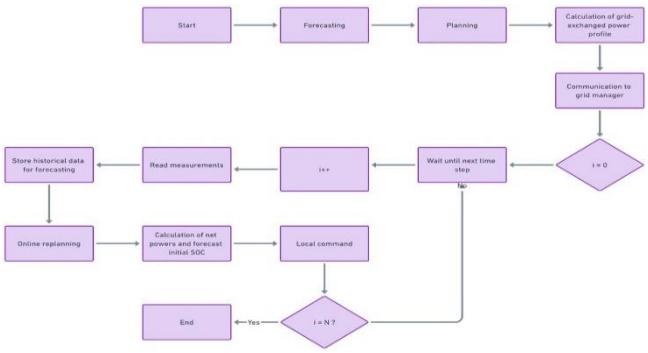

Figure 1: Illustrates the flowchart of the Energy Power Monitoring & Management System (EPMMS).

Real - Time Monitoring and Data Collection: IoT sensors gather real - time data on various parameters such as temperature, humidity, occupancy, and energy usage. This data is continuously collected and transmitted to a central system for analysis.

Predictive Analytics: Machine learning algorithms analyze the collected data to identify patterns and predict future energy usage. These predictions enable proactive adjustments to heating, ventilation, and air conditioning (HVAC) systems, lighting, and other energy - consuming processes.

Optimization of Energy Usage: Machine learning models optimize energy consumption by dynamically adjusting systems based on real - time data and predictions. For instance, adjusting HVAC settings based on occupancy patterns or natural light levels can significantly reduce energy consumption.

Fault Detection and Predictive Maintenance: IoT sensors can detect anomalies in equipment performance. Machine learning algorithms can predict potential failures, allowing for timely maintenance before issues escalate, ensuring equipment runs efficiently and reducing energy waste.

Automated Control Systems: IoT devices can automate energy management tasks. For example, smart thermostats and lighting systems can adjust settings automatically based on occupancy data, time of day, and weather forecasts.

User Behavior Insights: Machine learning can analyze user behavior and preferences, providing insights into how energy is used and suggesting more efficient habits. Personalized recommendations can help users make informed decisions about their energy consumption.

Integration with Renewable Energy Sources: IoT and machine learning can optimize the use of renewable energy sources. By predicting energy production and consumption patterns, the system can efficiently manage the balance between renewable and non - renewable energy sources, maximizing the use of clean energy.

Cost Benefits of Implementing Predictive Maintenance in Manufacturing Implementing predictive maintenance in manufacturing offers several cost benefits:

Reduced Downtime: Predictive maintenance minimizes unplanned downtime by addressing issues before they lead to equipment failures. This ensures continuous production and reduces the costs associated with unexpected stoppages.

Extended Equipment Lifespan: Regular and timely maintenance prevents excessive wear and tear on machinery, extending the lifespan of equipment. This reduces the frequency of expensive replacements and capital expenditures.

Lower Maintenance Costs: Predictive maintenance optimizes maintenance schedules, ensuring that maintenance is performed only when necessary. This reduces the costs associated with unnecessary maintenance activities and labor.

Energy Efficiency: Well - maintained equipment operates more efficiently, consuming less energy. This reduces energy costs and contributes to overall cost savings.

Inventory Management: Predictive maintenance allows for better planning and management of spare parts inventory. Knowing in advance when parts will need replacement helps in maintaining optimal inventory levels, reducing storage costs and avoiding stockouts.

Improved Product Quality: Consistently performing equipment ensures that products are manufactured to the desired quality standards, reducing the costs associated with defects and rework.

Enhanced Safety: Predictive maintenance identifies potential issues that could lead to unsafe operating conditions. Addressing these issues proactively reduces the risk of accidents and associated costs.

Increased Production Efficiency: By preventing equipment failures and optimizing maintenance activities, predictive maintenance ensures that manufacturing processes run smoothly, improving overall

4. Key Components of an Web Interface for EPMMS IoT - Based Energy Monitoring System

The system's web interface is divided into two parts: Raspberry Pi offers cost - effectiveness, flexibility, compact size, robust connectivity, and extensive community support, making it an ideal choice for energy monitoring systems. Minimized User Cash Flow Effectively manages power consumption and generation to optimize financial outcomes. Reduced GPP Uncertainty Online Replanning minimizes deviations between actual and planned GPP, enhancing predictability. Optimized Power Generation and DistributionFacilitates accurate and reliable power management, aiding public utilities in economic policy planning. Fundamentals of the Chosen Energy Power Monitoring and Management System (EPMMS) enhances efficiency and reliability in power management through several key stages and optimization processes:

Forecasting:
Predicts 24 - hour profiles of load demand and environmental variables using a Nonlinear Autoregressive network with exogenous inputs (NARX). Utilizes historical data for accurate predictions.

Planning: Plans to meet forecasted load demand while optimizing user benefits and grid management. Divided into Planning and Online Replanning tasks.

Calculation of Grid - Exchanged Power Profile (GPP):
Optimizes the user's cash flow over a 24 - hour period, producing power reference values for system components.
Computes the optimal GPP for the next day, indicating when the grid supplies power to the smart house and when renewable energy is fed back into the grid.

Communication to Grid Manager: Transmits the planned GPP to the grid manager via the Internet, serving as a user commitment.

Real - Time Measurements: Collects real - time data on power consumption and generation.

Historical Data Storage: Stores collected data to improve future forecasting accuracy.

Online Replanning: Minimizes deviations between actual and forecasted GPP by realigning performance with planned GPP.

Local Command: Issues commands to system components based on re - planning results.

Real- Time Data Display: Provides graphical representations of daily, weekly, or monthly data for each node (sensor and microcontroller set). Monitors multiple appliances with nodes for each.

Energy Consumption and Cost Estimation: Displays real - time total energy consumption and estimated electricity costs for each node.

5. Development of Energy Power Monitoring and Management System (EPMMS) Industries Energy Monitoring Using Raspberry Pi IoT

In this proposed system, loads are monitored using a PZEM - 004T module. The core microcontroller is the Raspberry Pi, featuring Wi - Fi capabilities. Key components include, The web interface, built using Python, Adobe Dreamweaver, HTML, and PHP, with MySQL for database interactions, ensures data protection through a secure login page.

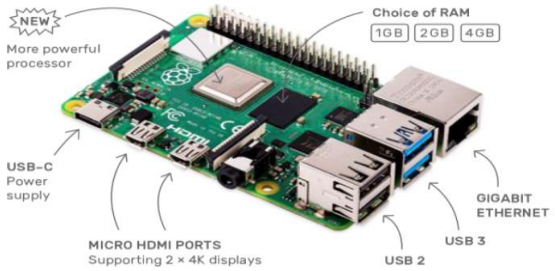

Figure 2: Schematic of Raspberry Pi 4

The process begins with the detection of electrostatic fields on appliances using a sensor. Sensor Detection and Signal Amplification The sensor detects electrostatic fields present on the appliances. The detected small current is amplified by a cascade amplifier circuit. Analog to Digital ConversionThe amplified signal is sent to the NodeMCU/Node. js's analog input.
This analog input is then converted into a decimal value ranging from 0 to 1024 by the Analog - to - Digital Converter (ADC) in the NodeMCU/ Node. js. and then Data Sampling The system is programmed to delay for 1 minute between readings. This process repeats until five samples are collected. Average Data Calculation and Transmission Once five samples are collected, the system calculates the average value of the data. The averaged data is sent to the ESP8266 Wi - Fi module. Data Transmission to Server: The ESP8266 Wi - Fi module transmits the data to the database server. The data along with the timestamp is stored in the database. Sampling Interval The microcontroller is programmed to read sensor values and send data to the database every 5 minutes. This interval balances data capture efficiency and database storage requirements.

6. Material Used

System Overview
- Raspberry Pi 4
- PZEM - 004T Sensor

- Modbus TCP NodeMCU ESP8266 12 - E Master Slave Communication
- Buzzer
- AWS Software

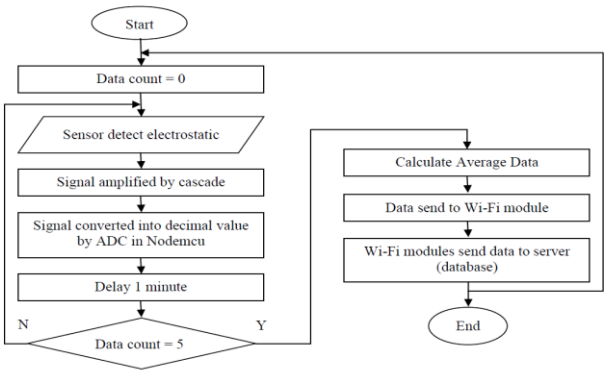

Figure 3: Illustrates the schematic of the Raspberry Pi 4 web - based Energy Power Monitoring & Management System

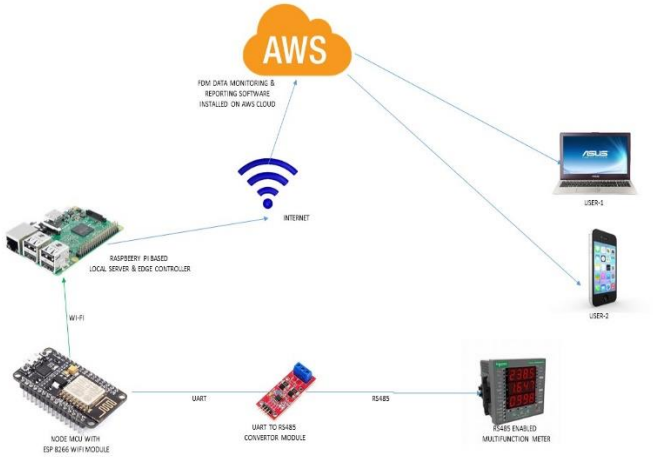

Figure 4: Illustrates the schematic of the Raspberry Pi 4 web - based Energy Power Monitoring & Management System

Hardware Requirements for IoT EPMMS

Display – Support: The Raspberry Pi 4 can be connected to external LCDs via a two - lane MIPI DSI port, bypassing the need for using the GPIO header for display connectivity.

USB Ports: Four USB ports, two of which are USB 2.0 and two USB 3.0, allowing for fast data transfer.

PoE Header: Supports Power over Ethernet (PoE), enabling power to be delivered through an Ethernet cable with an external PoE HAT. The Raspberry Pi 4 includes a PoE header, facilitating its use in IoT and other smart projects. With the addition of an external PoE HAT, the device can be powered through an Ethernet cable, simplifying installation and reducing cable clutter.

Camera Support: Two - lne MIPI CSI camera port for direct camera connectivity.

Display Support: Two - lane MIPI DSI port for external LCD connection.

Audio: Separate 4 - pole audio port for audio communication. The device supports audio transmission through HDMI. Additionally, it features a separate 4 - pole audio port for sending and receiving audio signals, which can be used by internal programs or other devices via the GPIO header.

SD Card: The operating system is stored on an SD card inserted into the Raspberry Pi 4's SD card slot.

Sensors and Measurement Devices: Capture real - time data on energy consumption.

Microcontroller/Microcomputer: Process data, e. g., Raspberry Pi

Communication Modules: Enable data transmission, e. g., Wi - Fi, Bluetooth.

Power Supply: Ensure reliable power to devices.

PZEM - 004T Module: The PZEM - 004T module is a multi - functional AC power monitor capable of measuring voltage, current, power, and energy. The Peacefair PZEM - 004T is a versatile AC power monitor widely used for electrical consumption measurement projects. It measures voltage, current, power, and energy in systems. It is ideal for single - phase AC measurement projects and communicates using TTL serial communication, making it compatible with various microcontrollers such as Arduino, ESP8266, STM32, WeMos, NodeMCU/ Node. js, and Raspberry Pi 4. Power the PZEM - 004T board with a suitable power supply or the AC source being measured. Connect the circular sensor to the board and run the wire through the sensor to begin measurements. The module includes a TTL to USB connector for easy connection to a PC or microcontroller.

ESP8266 Wi - Fi Module: The ESP8266 is a low - cost Wi - Fi module embedded within the NodeMCU, operating on the 802.11b/g/n protocol. It is utilized to transmit data from the NodeMCU to a web server database. In this project, the NodeMCU reads data from a sensor every 5 minutes and sends this data to the database via the ESP8266 Wi - Fi module. Before communication can occur, the Wi - Fi module must be initialized with the local Wi - Fi SSID and password.

- **Reset Energy Knob:** Present on V2.0; V3.0 resets energy via software.
- **Measurement Accuracy:** V3.0 has improved accuracy and faster reading time compared to V2.0.
- **Communication Protocol:** Varies between V2.0 and V3.0.

Software Requirements for IoT EPMMS

Raspbian OS: An easy - to - use operating system installation manager for Raspberry Pi.

Advanced IP Scanner: A tool for scanning and managing devices on the local network.

VNC Viewer: A remote desktop application to access the Raspberry Pi GUI.

Python 3 IDLE: The integrated development environment for Python programming.

NODE. JS: Node. js uses an event - driven, non - blocking I/O model, which makes it suitable for applications that require high performance and scalability, such as real - time

applications, APIs, and micro - services. This model allows Node. js to handle many connections simultaneously, optimizing the use of system resources

Adobe Dreamweaver: A web development tool for designing and coding web pages.

Software and Analytics Tools: Analyze data and provide insights.

User Interface: Web/mobile applications for data access.

Security Measures: Protect data and communications.

AWS cloud software: AWS cloud services, we can use AWS IoT Core to manage the connection and data transmission Factory data monitoring and management (FDM) Store and analyze data

Implementation experimental setup & working of Hardware and Communication

Step 1: Algorithm PHP Script: Connects to a MySQL database. Receives the sensor data sent by the NodeMCU. Inserts the data into the sensor_data table. Calculates the total energy consumed based on the sensor values and a predefined threshold. Outputs the total energy consumed in kWh.

```
<?php
$servername = "your_server";
$username = "your_username";
$password = "your_password";
$dbname = "your_database";
// Create connection
$conn = new mysqli ($servername, $username, $password, $dbname);
// Check connection
if ($conn - >connect_error) {
 die ("Connection failed: ". $conn - >connect_error);
}
// Use prepared statements to prevent SQL injection
$stmt = $conn - >prepare ("INSERT INTO sensor_data (value, timestamp) VALUES (?, NOW ()) ");
$stmt - >bind_param ("i", $sensorValue);

// Get the sensor value from the URL
$sensorValue = $_GET ['value'];

// Execute the statement
if ($stmt - >execute ()) {
 echo "New record created successfully";
} else {
 echo "Error: ". $stmt - >error;
}
$stmt - >close ();
$conn - >close ();
?>
```

Step 2: Hardware Setup Components Needed: Raspberry Pi (any model with GPIO pins), ESP8266 Wi - Fi module (e. g., NodeMCU), USB cable for the NodeMCU, Jumper wires, Breadboard (optional), Power supply for Raspberry Pi

Connections: Connect the NodeMCU to the laptop using the USB cable. Connect the Raspberry Pi to its power supply and boot it up. Optionally, you can use a breadboard and jumper wires to connect the GPIO pins of the Raspberry Pi to the ESP8266 if needed for specific configurations (e. g., serial communication).

Step 3: Software Setup Install Python on Raspberry Pi: Ensure Python is installed on the Raspberry Pi. Most distributions come with Python pre - installed. You can check this by running:

```
python3 –version
sudo apt - get update
sudo apt - get install python3
```

Install Required Python Libraries: Install libraries such as pyserial for serial communication:

```
pip3 install pyserial
```

Install NodeMCU Firmware: You will need the NodeMCU firmware on the ESP8266. You can use the Arduino IDE or other firmware flashing tools.

Install Arduino IDE: Download and install the Arduino IDE from Arduino's official website. Add ESP8266 to the Arduino IDE: Open the Arduino IDE. Go to File > Preferences. In the Additional Board Manager URLs field, add the following URL: http: //arduino. esp8266. com/stable/package_esp8266com_index. jsonGo to Tools > Board > Board Manager, search for ESP8266, and install it.

Step 4: Programming the NodeMCU ESP8266 Code: Connects to the specified WiFi network. Reads data from a sensor connected to the analog pin (A0). Sends the sensor value to the server every 5 minutes.

```
#include <ESP8266WiFi. h>
#include <ESP8266HTTPClient. h>
#include <WiFiClientSecureBearSSL. h>
const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
const char* server Name = "https: //your_server_address/submit_data. php"; // HTTPS URL
const unsigned long interval = 300000; // 5 minutes
unsigned long previousMillis = 0;
void setup () {
 Serial. begin (115200);
 WiFi. begin (ssid, password);
 while (WiFi. status () != WL_CONNECTED) {
 delay (1000);
 Serial. println ("Connecting to WiFi. . . ");
 }
 Serial. println ("Connected to WiFi");
}
void loop () {
 unsigned long currentMillis = millis ();
 if (currentMillis - previousMillis >= interval) {
 previousMillis = currentMillis;
 int sensorValue = analogRead (A0); // Replace with your sensor reading code
 Serial. println (sensorValue);
 if (WiFi. status () == WL_CONNECTED) {
 std:: unique_ptr<BearSSL:: WiFiClientSecure> client (new BearSSL:: WiFiClientSecure);
```

```
client - >setInsecure (); // Disable SSL certificate validation
HTTPClient http;
String serverPath = serverName + "?value=" + String (sensorValue);
http.begin (*client, serverPath);
int httpResponseCode = http.GET ();
if (httpResponseCode > 0) {
String payload = http.getString ();
Serial. println ("HTTP Response code: " + String (httpResponseCode));
Serial. println ("Response: " + payload);
} else {
Serial. println ("Error in HTTP request");
}
http.end ();
} else {
Serial. println ("WiFi Disconnected");
}
}
}
```

Upload Code to NodeMCU: Connect the NodeMCU to your laptop via USB. Select the appropriate board and port in the Arduino IDE: Tools > Board > NodeMCU 1.0 (ESP - 12E Module) Tools > Port > Select the appropriate port Upload the code to the NodeMCU.

Step 5: Interfacing with Raspberry Pi Python Script on Raspberry Pi: Create a Python script to communicate with the NodeMCU over Wi - Fi. Example script to send a request to the NodeMCU server:

```
import requests.
url = "http: //<NodeMCU_IP>"
response = requests. get (url)
print (response. text)
```

Step 6: Run the Python Script: Execute the Python script to communicate with the NodeMCU.

```
python3 your_script. py
```

By following these steps, you can integrate the ESP8266 Wi - Fi module with Python and NodeMCU code, along with a Raspberry Pi, to create a comprehensive IoT solution. This setup allows for seamless communication between the Raspberry Pi and the NodeMCU, enabling you to perform various tasks such as data acquisition, remote monitoring, and control.

Step 7: Algorithm Node. Js Script: we need to interface the ESP8266 with an RS485 Modbus sensor and send the data to a cloud server using Node. js and Modbus TCP NodeMCU ESP8266 12 - E Master Slave Communication on a Raspberry Pi. This setup involves the following steps:

Set up the ESP8266 to read data from the RS485 Modbus sensor.

Send the data from the ESP8266 to the Raspberry Pi.

Use the Raspberry Pi to send the received data to the cloud.

Step 8: ESP8266 Code to Read Modbus Data: First, let's write the code for the ESP8266 to read data from the Modbus sensor. This example uses the ModbusMaster library to communicate with the sensor.

```
#include <ESP8266WiFi. h>
#include <ModbusMaster. h>
const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
const int modbusBaudRate = 9600;
const int modbusID = 1; // Replace with your Modbus device ID
ModbusMaster node;
void setup () {
Serial. begin (115200);
WiFi. begin (ssid, password);
while (WiFi. status () != WL_CONNECTED) {
delay (1000);
Serial. println ("Connecting to WiFi. . . ");
}
Serial. println ("Connected to WiFi");

Serial1. begin (modbusBaudRate); // RS485 serial connection
node. begin (modbusID, Serial1);
}
void loop () {
static uint32_t timer = millis ();
uint8_t result;
uint16_t data [6];
if (millis () - timer > 5000) {
timer = millis ();
result = node. readInputRegisters (0x0000, 6); // Replace with your register address and number
if (result == node. ku8MBSuccess) {
for (int i = 0; i < 6; i++) {
data [i] = node. getResponseBuffer (i);
}
String dataString = "";
for (int i = 0; i < 6; i++) {
dataString += String (data [i]) + " ";
}
Serial. println (dataString); // Send data to Raspberry Pi via Serial
} else {
Serial. println ("Failed to read Modbus data");
}
}
}
```

Step 9: Raspberry Pi Code to Receive and Send Data: the code for the Raspberry Pi to receive the data from the ESP8266 and send it to the cloud. This example uses the serialport library to read data from the serial port and node - fetch to send data to the cloud. Install the necessary libraries

```
npm install serialport node - fetch
```

Create a script to read the data from the serial port and send it to the cloud

```
const SerialPort = require ('serialport');
const Readline = require ('[at]serialport/parser - readline');
const fetch = require ('node - fetch');
const port = new SerialPort ('/dev/ttyUSB0', { baudRate: 115200 });
const parser = port. pipe (new Readline ({ delimiter: '\n' }));
const serverName = 'https: //your_server_address/submit_data. php'; // Replace with your server address
parser. on ('data', async (data) => {
```

```
 console. log ('Received data from ESP8266: ', data);
 // Send the data to the cloud server
 try {
 const response = await fetch (serverName, {
 method: 'POST',
 headers: { 'Content - Type': 'application/x - www - form - urlencoded' },
 body: new URLSearchParams ({ value: data. trim () })
 });
 const result = await response. text ();
 console. log ('Server response: ', result);
 } catch (error) {
 console. error ('Error sending data to server: ', error);
 }
});
```

Data Transmission and Calibration

The data transmitted from the NodeMCU to the server represents the strength of the electrostatic field detected by the sensor on various appliances. This data, in the form of decimal values ranging from 0 to 1024, indicates the ON or OFF state of the appliances. These values can differ between appliances and are affected by the distance between the antenna and the appliance's power supply unit. Hence, a calibration procedure is required for each new appliance installation. the calibration process is conducted via a website interface where high and low threshold values are set. The high threshold value is the minimum value indicating that the appliance is ON, while the low threshold value is the maximum value indicating that the appliance is OFF. To integrate your ESP8266 with an RS485 Modbus sensor and send the data to AWS cloud services, we can use AWS IoT Core to manage the connection and data transmission. Here's a step - by - step guide on how to achieve this:

Step 10: Set up AWS IoT Core: Create an AWS Account: If you don't already have one, create an AWS account. Create an IoT Thing: Navigate to the AWS IoT Core console. Create a new IoT thing. Download the security credentials (certificates and keys). Configure AWS IoT Policies Attach a policy to your IoT thing that allows it to publish and subscribe to topics.

Step 11: ESP8266 Code to Send Data to AWS IoT: First, let's set up the ESP8266 to read data from the RS485 Modbus sensor and send it to AWS IoT. This example uses the ESP8266WiFi library for WiFi connectivity and the PubSubClient library for MQTT communication with AWS IoT.

ESP8266 Code:

```
#include <ESP8266WiFi. h>
#include <PubSubClient. h>
#include <ModbusMaster. h>
#include <WiFiClientSecureBearSSL. h>
const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
// AWS IoT Core credentials
const char* awsEndpoint = "your_aws_iot_endpoint"; // Example: "your - aws - endpoint. iot. us - east - 1. amazonaws. com"
const int awsPort = 8883;
const char* awsCert = " - - - - - BEGIN CERTIFICATE - - - - - \n. . . \n - - - - - END CERTIFICATE - - - - - ";
const char* awsPrivateKey = " - - - - - BEGIN PRIVATE KEY - - - - - \n. . . \n - - - - - END PRIVATE KEY - - - - - ";
const char* awsCA = " - - - - - BEGIN CERTIFICATE - - - - - \n. . . \n - - - - - END CERTIFICATE - - - - - ";
const char* awsTopic = "your/topic";
WiFiClientSecure wifiClient;
PubSubClient client (wifiClient);
ModbusMaster node;
void setup () {
 Serial. begin (115200);
 WiFi. begin (ssid, password);
 while (WiFi. status () != WL_CONNECTED) {
 delay (1000);
 Serial. println ("Connecting to WiFi. . . ");
 }
 Serial. println ("Connected to WiFi");
 wifiClient. setCACert (awsCA);
 wifiClient. setCertificate (awsCert);
 wifiClient. setPrivateKey (awsPrivateKey);
 client. setServer (awsEndpoint, awsPort);
 Serial1. begin (9600); // RS485 serial connection
 node. begin (1, Serial1); // Modbus ID 1
}
void loop () {
 if (!client. connected ()) {
 Serial. println ("Connecting to AWS IoT. . . ");
 if (client. connect ("ESP8266Client")) {
 Serial. println ("Connected to AWS IoT");
 } else {
 Serial. print ("Failed to connect, rc=");
 Serial. print (client. state ());
 delay (5000);
 return;
 }
 }
 uint8_t result = node. readInputRegisters (0x0000, 6); // Replace with your register address and number
 if (result == node. ku8MBSuccess) {
 String payload = "{";
 for (int i = 0; i < 6; i++) {
 payload += "\"data" + String (i) + "\": " + String (node. getResponseBuffer (i));
 if (i < 5) payload += ", ";
 }
 payload += "}";

 Serial. println (payload);
 if (client. publish (awsTopic, payload. c_str ())) {
 Serial. println ("Publish succeeded");
 } else {
 Serial. println ("Publish failed");
 }
 } else {
 Serial. println ("Failed to read Modbus data");
 }
 delay (5000); // Adjust the interval as needed
}
```

Step 11: Run Script ESP8266 Code and Step 3: Setting Up Lambda Function and API Gateway Create a Lambda Function Go to the AWS Lambda console and create a new function.

Write the Lambda function to process the incoming data. Create an API Gateway Create a new API and configure a POST endpoint. Link the endpoint to the Lambda function created earlier.

Step 12: Raspberry Pi Code to Send Data to AWS Lambda: Install the necessary libraries on your Raspberry Pi: npm install serialport node - fetch

Raspberry Pi Node. js Script:

```
const SerialPort = require ('serialport');
const Readline = require ('[at]serialport/parser - readline');
const fetch = require ('node - fetch');
const port = new SerialPort ('/dev/ttyUSB0', { baudRate: 115200 });
const parser = port. pipe (new Readline ({ delimiter: '\n' }));
const lambdaEndpoint = 'https: //your_api_gateway_url'; // Replace with your API Gateway URL
parser. on ('data', async (data) => {
 console. log ('Received data from ESP8266: ', data);
 try {
 const response = await fetch (lambdaEndpoint, {
 method: 'POST',
 headers: { 'Content - Type': 'application/json' },
 body: JSON. stringify ({ value: data. trim () })
 });
 const result = await response. text ();
 console. log ('Server response: ', result);
 } catch (error) {
 console. error ('Error sending data to server: ', error);
 }
});
console. log ('Listening for data from ESP8266. . . ');
```

Energy Calculation

Based on the data stored in the database, the duration that each appliance remains ON is calculated. Each data point represents a 5 - minute duration, which is converted to hours for energy calculation on the website using HTML and PHP. The energy consumption is calculated using the following formula:

Energy = P \ t

Where:

(P) is the appliance's power rating (kW)

(t) is the appliance's usage duration (hours)

The estimated electricity bill is then calculated based on the tariff rates provided by electricity board

Sensor Detection and Signal Amplification: The sensor detects electrostatic fields present on the appliances. The detected small current is amplified by a cascade amplifier circuit.

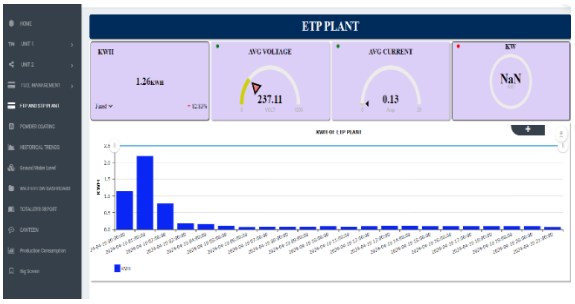

Figure 4: Illustrates the graph of the Raspberry Pi 4 web - based Energy Power Monitoring & Management System

Analog to Digital Conversion: The amplified signal is sent to the NodeMCU's analog input. This analog input is then converted into a decimal value ranging from 0 to 1024 by the Analog - to - Digital Converter (ADC) in the NodeMCU.

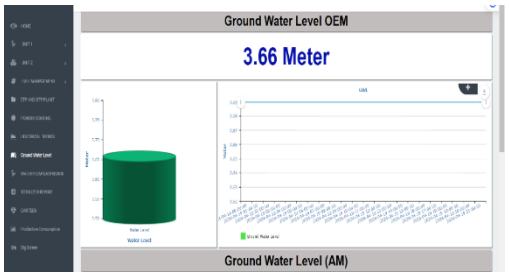

Figure 5: Illustrates the analog to digital conversion with the Raspberry Pi 4 web - based Energy Power Monitoring & Management System

Data Sampling: The system is programmed to delay for 1 minute between readings. This process repeats until five samples are collected.

Figure 5: Illustrates the meter data collection with the Raspberry Pi 4 web - based Energy Power Monitoring & Management System

Average Data Calculation and Transmission: Once five samples are collected, the system calculates the average value of the data. The averaged data is sent to the ESP8266 Wi - Fi module.

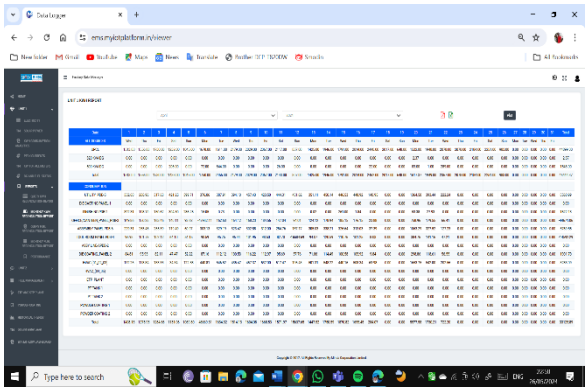

Figure 6: Illustrates the Calculation with the Raspberry Pi 4 web - based Energy Power Monitoring & Management System

Data Transmission to Server:
The ESP8266 Wi - Fi module transmits the data to the database server. The data, along with the timestamp, is stored in the database.

Figure 7: Illustrates the Report with the Raspberry Pi 4 web - based Energy Power Monitoring & Management System

Sampling Interval: The microcontroller is programmed to read sensor values and send data to the database every 5 minutes. This interval balances data capture efficiency and database storage requirements.

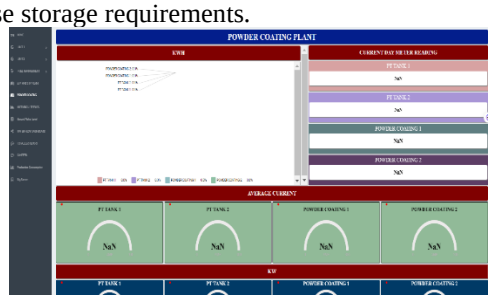

Figure 8: Illustrates the Report with efficiency with the Raspberry Pi 4 web - based Energy Power Monitoring & Management System

7. Conclusion

An attempt has been made to make a practical model of "Real - Time IoT - Based Energy and Power Monitoring and Management System for Small - Scale Applications Using a Raspberry Pi Web Interface". The energy power monitoring and management system leveraging Raspberry Pi 4, Arduino Uno, and PZEM - 004T and smart meters with modbus converter in master slave configuration offers a comprehensive solution for reducing carbon footprints through efficient energy usage monitoring and management. By providing real - time energy readings at one's fingertips, this system can significantly enhance energy conservation efforts.

The propagated model is used to calculate the energy consumption of the Load, and even make the energy reading to be handy. Hence it reduces the wastage of energy with a 6.88% reduction in electricity and bring awareness among all. Even it will deduct the manual intervention. This project proposes a secure, ubiquitously accessible controlled solution for Industrial and commercial building EPMMS automation. Looking at the current scenario we have chosen web based platform so that most of the people can get the benefit.

Performance Evaluation: The system has demonstrated a good level of accuracy in measuring electrical parameters, with average deviations for small loads (65KW, 90KW, 75KW) at 0.11%, 1.94%, and 1.32%, respectively, and for large loads (200KW, 300KW,) at 1.22%, 1.03%, and 3.08%. & reduce carbon footprint by 6.67% These results indicate that the system performs reliably across different load levels.

Reducing Carbon Footprint & Real - Time Monitoring: The system allows users to monitor electrical parameters such as voltage, current, power, and energy consumption accurately. By understanding their energy usage patterns, users can take proactive steps to reduce unnecessary consumption, thus lowering their carbon footprint by 6.67%
Base line Energy consumption = 300KWH
Post - EMS Energy Consumption - =280KWH
Emission Factor=0.5 kg CO2/kWh
*Energy Consumption Reduction (ECR) = ((300 - 280) /300) *100=6.67%*
*Baseline Carbon Emissions (CE) =300KWH*0.5=150KG CO2*
*Post - EMS Carbon Emissions (PCE) =280 KWH*0.5=140KG CO2.*
*Carbon Emission Reduction (CER) = ((150 - 140) /150) *100) = 6.67%*

Energy Usage Awareness: The availability of detailed energy consumption data promotes awareness among users about their energy habits. This awareness can drive behavioral changes that contribute to energy savings and reduced environmental impact.

Automated Alerts and Actions: With the ability to set up alerts for unusual energy consumption patterns or potential energy meter tampering, users can address issues promptly, preventing energy wastage.

Predictive Analytics: we are able to integrated into the system to analyze historical energy usage data and predict future consumption trends. This predictive capability can help users plan and optimize their energy usage, further reducing their carbon footprint.

Smart Alerts: we are able to enhance the system by generating intelligent alerts based on usage patterns, such as reminding users to turn off appliances during peak energy hours or when a device is consuming more power than usual.

Automated Control: we are able to enable automated control of household devices, shutting down appliances when not in use or adjusting settings based on real - time data to optimize energy efficiency.

Unified Platform: we are able to designed to provide a unified platform for viewing energy usage, receiving alerts, and paying bills online, supporting the digital India initiative and making energy management more convenient for users.

Enhanced User Interface: we are able to provide user interface improved to provide more intuitive and accessible features for both desktop and mobile users, ensuring a seamless experience across devices.

8. Future Scope

Improved Sensor Capabilities: With AI, a single sensor can be developed to measure, alert, and notify users about their energy consumption, streamlining the hardware requirements and improving system efficiency.

Practical Application: By implementing this IoT - based power management system, users can effectively monitor and control their household energy consumption, reducing energy wastage and contributing to environmental sustainability. The use of an Android platform ensures widespread accessibility, making it a practical solution for modern Industrial and commercial building EPMMS automation.

Energy Meter Tampering Detection: The system can be extended to detect energy meter tampering, ensuring accurate billing and preventing energy theft.

In conclusion, the integration of AI and IoT technologies in energy power monitoring and management systems presents a promising approach to achieving significant reductions in carbon footprints while enhancing user convenience and system efficiency.

Acknowledgement
It gives us great pleasure in presenting the paper on “Real - Time IoT - Based Energy and Power Monitoring and Management System for Small - Scale Applications Using a Raspberry Pi Web Interface”. We would like to take this opportunity to thank our internal guide of the Electrical Engineering Department, Prof. Dr. Jyoti Srivastav, for giving us all the help and guidance we needed. We are also thankful to Mr. Dr. Manish Kr Srivastav, Head of the Electrical Engineering Department, for guiding us through the project selection process. We are grateful to them for their kind support. Their valuable suggestions were very helpful.
We extend our thanks to the project coordinators, Mr. Anupam Mashi, Assistant Professor, and Assistant Professor Mr. Sudhanshu Tripathi, of Electrical Department, for their indispensable support and suggestions. I would also like to gratefully acknowledge the assistance of Vishal Razdan from Schneider Electric India for mentoring us and Mr. Ashish Kumar, MD of M/s Caiman Solution Lucknow, for the financial support. Their collective guidance and support were instrumental in the successful completion of this project, and we are sincerely thankful for their contributions.

References

[1] S. Wasoontarajaroen, K. Pawasan, and V. Chamnanphrai 2017 Development of an IoT device for monitoring electrical energy consumption 2017 9th Int. Conf. Inf. Technol. Electr. Eng. ICITEE 2017 pp.1–4 2017 doi: 10.1109/ICITEED.2017.8250475.
[2] F. Benzi, N. Anglani, E. Bassi, and L. Frosini 2011 Electricity smart meters interfacing the households IEEE Trans. Ind. Electron. vol.58 no.10 pp.4487–4494 2011 doi: 10.1109/TIE.2011.2107713.
[3] R. Rashed Mohassel, A. Fung, F. Mohammadi, and K. Raahemifar 2014 A survey on Advanced Metering Infrastructure Int. J. Electr. Power Energy Syst. vol.63 pp.473–484 2014 doi: 10.1016/j. ijepes.2014.06.025.
[4] A. Zaballos, A. Vallejo, M. Majoral, and J. M. Selga 2009 Survey and performance comparison of AMR over PLC standards IEEE Trans. Power Deliv. vol.24 no.2 pp.604–613 2009 doi: 10.1109/TPWRD.2008.2002845.
[5] F. D. Garcia, F. P. Marafao, W. A. De Souza, and L. C. P. Da Silva 2017 Power Metering: History and Future Trends in IEEE Green Technologies Conference 2017 pp.26–33 doi: 10.1109/GreenTech.2017.10
[6] National_Energy_Council, “Indonesia EnergyOutlook building energy monitoring and management system based on wireless sensor networks, ” Proc. - 2015 10th Int. Conf. Comput. Eng. Syst. ICCES 2015, pp.230–233, 2016.
[7] Nouby M. Ghazaly, M. M. A. . (2022). A Review on Engine Fault Diagnosis through Vibration Analysis. International Journal on Recent Technologies in Mechanical and Electrical Engineering, 9 (2), 01–06. https: //doi. org/10.17762/ijrmee. v9i2.364
[8] Pawan Kumar Tiwari, Mukesh Kumar Yadav, R. K. G. A. (2022). Design Simulation and Review of Solar PV Power Forecasting Using Computing Techniques. International Journal on Recent Technologies in Mechanical and Electrical Engineering, 9 (5), 18–27. https: //doi. org/10.17762/ijrmee. v9i5.370
[9] Ghazaly, N. M. . (2022). Data Catalogue Approaches, Implementation and Adoption: A Study of Purpose of Data Catalogue. International Journal on Future Revolution in Computer Science &Amp; Communication Engineering, 8 (1), 01–04. https: //doi. org/10.17762/ijfrcsce. v8i1.2063
[10] Chaudhary, D. S. . (2022). Analysis of Concept of Big Data Process, Strategies, Adoption and Implementation. International Journal on Future Revolution in Computer Science &Amp; Communication Engineering, 8 (1), 05–08. https: //doi. org/10.17762/ijfrcsce. v8i1.2065
[11] Malla, S., M. J. Meena, O. . Reddy. R, V. . Mahalakshmi, and A. . Balobaid. “A Study on Fish Classification Techniques Using Convolutional Neural Networks on Highly Challenged Underwater Images”. International Journal on Recent and Innovation Trends in Computing and Communication, vol.10, no.4, Apr.2022, pp.01 - 09, doi: 10.17762/ijritcc. v10i4.5524.
[12] A. Chawla, “Phishing website analysis and detection using Machine Learning”, Int J Intell Syst Appl Eng, vol.10, no.1, pp.10–16, Mar.2022.
[13] M. Dursun and N. Goker, “Evaluation of Project Management Methodologies Success Factors Using Fuzzy Cognitive Map Method: Waterfall, Agile, And Lean Six Sigma Cases”, Int J Intell Syst Appl Eng, vol.10, no.1, pp.35–43, Mar.2022.