

IoT based Smart Energy Meter System

(EC 881) Project Stage-II

**A Project Report Submitted in Partial Fulfillment of the Requirements
for the degree of Bachelor of Technology in
Electronics and Communication Engineering
by:**

Name: SWARNAVA GAYEN

Registration No: 201300100320027 of 2020-21 Roll No: 13000320128

Name: SAYAK MITRA

Registration No: 201300100320004 of 2020-21 Roll No: 13000320151

Name: SAYAN MONDAL

Registration No: 201300100320003 of 2020-21 Roll No: 13000320152

Name: SHIRSANI NASKAR

Registration No: 201300100320001 of 2020-21 Roll No: 13000320154

Under the guidance of

Prof. Debaprasad De

Department of Electronics and Communication Engineering



**TECHNO MAIN SALT LAKE
EM 4/1 Salt Lake City, Sector – V
Kolkata – 700091
i**

Certificate

This is to certify that **Swarnava Gayen , Sayak Mitra, Sayan Mondal & Shirsani Naskar** has carried out his project work entitled “**Internet of Things based Smart Energy Meter**” as a part of the curriculum for the **B.tech Degree** in **Electronics & Communication Engineering (ECE)** under **Maulana Abul Kalam Azad University of Technology** for the year **2020-2023**.

This project report is approved by the undersigned only for the purpose for which it is submitted. The candidate is entirely responsible for the statements, opinions and conclusions contained herein.

(Prof. Debaprasad De)

(Signature of the Mentor)

(Signature of HOD, ECE Dept,

Techno Main Salt Lake, with date)

Acknowledgement

We express our special thanks of gratitude to **Prof. Debaprasad De** of the Department of **Electronics & Communication Engineering (ECE)** for his able guidance and valuable support in completing our project.

Heartfelt thanks are also conveyed to all the members of the teaching and non-teaching staff of the Department of ECE for their cordial support and help, whenever needed.

Date: 27.04.2023

.....
Swarnava Gayen

.....
Sayak Mitra

.....
Sayan Mondal

.....
Shirsani Naskar

Content:

Page No.

1. List of Figures.....	v
2. List of Tables.....	vi
3. Abstract.....	1
4. Introduction.....	2-3
5. Literature Review.....	4-5
6. Methodology.....	6-36
6.1 Problem definition.....	8-10
6.2 Planning and Approach.....	11-34
6.3 Design Issues.....	35-36
7. Simulation, Measurements, Results and Discussion.....	37-45
8. Conclusion and Scope of future work.....	46-50
9. References.....	51-52

1. LIST OF TABLES

SL.NO	TABLE NAME	PAGE NUMBER
1	Cost Estimation Table	19

2. LIST OF FIGURES

SL.NO	NAME OF FIGURE	PAGE.NO
1	Arduino UNO	12
2	NODE MCU/ESP8266	14
3	ACS712	16
4	CIRCUIT DIAGRAM	18
5	CIRCUIT SIMULATION	21
6	CODING PART OF ARDUINO UNO	27
7	OUTPUT OF METER	27
8	CODING PART OF NODE MCU	28
9	OUTPUT OF METER	29
10	BILL SENT	29

3. ABSTRACT

The objective of this project is to design and implement a smart energy meter system using the ESP8266, Arduino Uno, LCD display (16x2), ACS712 current sensor, and ZMPT101B AC voltage sensor. The system aims to provide real-time monitoring and analysis of energy consumption in households or small-scale applications. This project report presents a comprehensive overview of the system's development, including its design, implementation, and evaluation.

The initial phase involves an extensive literature review of existing smart energy meter systems, focusing on their functionalities, technologies, and limitations. The selected components are carefully examined and justified based on their suitability for the project requirements. The circuit design is explained, accompanied by detailed diagrams illustrating the connections between the components.

The implementation phase details the step-by-step process of integrating and configuring the components. It includes a thorough explanation of the code structure, algorithms, and logic used to measure and process the data from the ACS712 current sensor and ZMPT101B AC voltage sensor. Challenges faced during the implementation are discussed, and solutions are proposed to overcome them.

The results and analysis section presents the collected data from the smart energy meter system. The accuracy and reliability of the measurements are evaluated through comparisons with standard reference values or similar devices. The data analysis provides insights into energy consumption patterns, allowing users to make informed decisions about energy usage optimization and conservation.

The discussion section interprets the results, evaluates the performance of the smart energy meter system, and discusses potential limitations. It explores the implications of the findings and suggests future enhancements, such as integrating additional features like data visualization, remote monitoring, or energy consumption alerts.

This project showcases the successful development and implementation of a smart energy meter system using readily available components. The system offers an efficient and user-friendly solution for monitoring energy consumption, promoting energy awareness, and enabling users to take proactive steps towards sustainable energy management.

INTRODUCTION

The increasing demand for efficient energy management and the growing focus on sustainable energy consumption have led to the development of smart energy meter systems. These systems provide real-time monitoring and analysis of energy consumption, allowing users to make informed decisions to optimize their energy usage and reduce wastage. This project aims to design and implement a smart energy meter system using the ESP8266, Arduino Uno, LCD display (16x2), ACS712 current sensor, and ZMPT101B AC voltage sensor.

The primary goal of this project is to create a cost-effective and easily replicable energy meter system that can be deployed in households or small-scale applications. By utilizing the ESP8266, Wi-Fi connectivity is enabled, allowing users to remotely monitor and analyze their energy consumption data. The Arduino Uno serves as the central control unit, processing the sensor readings and displaying them on the LCD display.

The ACS712 current sensor is employed to measure the current flowing through the system, providing accurate information about energy consumption. The ZMPT101B AC voltage sensor, on the other hand, enables the measurement of AC voltage levels. Together, these sensors provide comprehensive data that allows users to understand their energy usage patterns and identify opportunities for energy conservation.

The integration of these components into a cohesive system requires careful circuit design, code implementation, and calibration. By developing this smart energy meter system, users gain the ability to track and analyze their energy consumption in real-time, promoting energy efficiency and reducing environmental impact.

Throughout this project, considerations are given to the affordability, accessibility, and ease of use of the components selected. The report provides detailed information about the design choices, implementation process, and challenges faced during the project. Additionally, it explores potential future enhancements that could be implemented to further enhance the functionality and usability of the smart energy meter system.

Furthermore, the project report outlines the challenges encountered during implementation and the

corresponding solutions devised to overcome them. These insights not only highlight the technical aspects of the project but also provide valuable knowledge for future endeavors in the field of smart energy meter systems.

This project seeks to contribute to the advancement of smart energy meter systems by creating a practical and affordable solution that empowers users to actively manage their energy consumption. By promoting energy awareness and providing actionable insights, this system aims to encourage sustainable energy practices for a greener and more efficient future.

Literature Review

Smart energy meter systems have gained significant attention in recent years as a means to monitor and manage energy consumption effectively. This section provides a literature review of existing smart energy meter systems, as well as relevant technologies and methodologies employed in such systems. The review aims to identify the key features, advancements, and limitations of previous works, providing a foundation for the design and implementation of the current project.

1. Smart Energy Meter Systems:

Numerous studies have highlighted the importance of smart energy meter systems in promoting energy efficiency and reducing energy wastage. These systems typically consist of sensors to measure current and voltage, a microcontroller for data processing, and a communication module for data transmission. Advanced systems incorporate features such as wireless connectivity, data analysis algorithms, and user-friendly interfaces.

A study by Singh et al. (2018) proposed a smart energy meter system utilizing Arduino and ESP8266 for monitoring and controlling energy consumption in residential buildings. Their system featured real-time data display, anomaly detection, and remote monitoring capabilities.

Similarly, Nair and Ganapathy (2019) developed a smart energy meter system using Arduino and GSM technology for remote data access. The system facilitated energy monitoring and billing, allowing consumers to track and manage their energy usage efficiently.

2. Component Selection:

The selection of components plays a crucial role in the design and functionality of smart energy meter systems. The ESP8266, Arduino Uno, LCD display, ACS712 current sensor, and ZMPT101B AC voltage sensor have been commonly utilized due to their availability, affordability, and compatibility.

The ESP8266 has gained popularity as a Wi-Fi module, enabling wireless connectivity and remote access to energy consumption data. Its integration with Arduino Uno provides a robust platform for data processing and communication.

The ACS712 current sensor has been widely adopted for its ability to measure current flowing through electrical systems accurately. Similarly, the ZMPT101B AC voltage sensor is frequently used for its reliable measurement of AC voltage levels.

3. Data Processing and Analysis:

Effective data processing and analysis techniques are essential in smart energy meter systems to extract meaningful insights from the collected data. Various algorithms and methodologies have been proposed to analyze energy consumption patterns, detect anomalies, and provide energy-saving

recommendations.

Gupta et al. (2017) proposed an algorithm for energy consumption analysis using time series data obtained from smart energy meters. Their approach involved segmenting the data, identifying patterns, and generating energy-saving recommendations based on historical trends.

Machine learning techniques have also been employed for energy consumption prediction and anomaly detection. Li et al. (2020) utilized a long short-term memory (LSTM) neural network model to predict energy consumption and identify abnormal usage patterns, enhancing energy management and efficiency.

4. User Interfaces and Visualization:

The user interface plays a crucial role in enhancing the usability and accessibility of smart energy meter systems. Visual representation of energy consumption data through graphical interfaces facilitates user understanding and promotes energy-conscious behavior.

Nguyen et al. (2019) developed a smart energy meter system with a user-friendly interface, featuring real-time data visualization and personalized energy-saving recommendations. Their study demonstrated that user engagement and awareness increased significantly through intuitive visualizations.

Additionally, smartphone applications have been developed to provide on-the-go access to energy consumption data and control options. These applications enable users to monitor and manage their energy usage conveniently, even when away from home.

In conclusion, the literature review highlights the significance of smart energy meter systems in promoting energy efficiency and conservation. Previous studies have focused on various aspects, including system architecture, component selection, data processing algorithms, and user interfaces. The reviewed literature provides valuable insights and serves as a foundation for the design and implementation of the current project, utilizing the ESP8266, Arduino Uno, LCD display, ACS712 current.

6. METHODOLOGY

This section presents the methodology employed in the design and implementation of the smart energy meter system using the ESP8266, Arduino Uno, LCD display (16x2), ACS712 current sensor, and ZMPT101B AC voltage sensor. The methodology encompasses the steps taken to achieve the project objectives, including component selection, circuit design, code implementation, and system testing.

1. Component Selection:

The first step in the methodology involved the careful selection of components based on their compatibility, affordability, and availability. The ESP8266 was chosen for its Wi-Fi connectivity capabilities, enabling remote access to energy consumption data. The Arduino Uno was selected as the central control unit for its reliability and ease of use. The LCD display (16x2) was chosen to provide a user-friendly interface for data visualization. The ACS712 current sensor and ZMPT101B AC voltage sensor were selected for their accuracy in measuring current and voltage, respectively.

2. Circuit Design:

The next phase focused on designing the circuitry for the smart energy meter system. Circuit diagrams were created to illustrate the connections between the components, ensuring proper functionality and accurate data acquisition. Consideration was given to proper placement of resistors and capacitors to ensure signal conditioning and noise reduction.

3. Code Implementation:

Once the circuit design was finalized, code implementation was carried out to enable the system to measure, process, and display energy consumption data. The Arduino IDE was used to write and upload the code to the Arduino Uno. The code included logic for reading data from the ACS712 current sensor and ZMPT101B AC voltage sensor, performing necessary calculations, and displaying the results on the LCD display. Proper error handling and exception handling techniques were implemented to ensure reliable operation.

4. Calibration and Testing:

Calibration of the sensors was performed to ensure accurate measurements. This involved verifying the sensor outputs against known reference values and adjusting any necessary calibration factors in the code. Rigorous testing was conducted to validate the functionality and accuracy of the system. Real-time energy consumption data was collected and compared against known loads to verify the system's reliability and accuracy.

5. Performance Evaluation:

The performance of the smart energy meter system was evaluated by collecting data over an extended period. Data analysis was conducted to identify energy consumption patterns and trends. The accuracy and reliability of the measurements were assessed by comparing the system's readings with reference values or other established energy meter systems. Any discrepancies or limitations were identified and addressed.

6. Documentation:

Throughout the project, thorough documentation of the design, implementation, and testing processes was maintained. This included circuit diagrams, code snippets, calibration procedures, and testing results. Additionally, a project report was prepared to summarize the methodology, findings, and recommendations for future improvements.

By following this methodology, the smart energy meter system was successfully designed, implemented, and evaluated, providing users with real-time energy consumption data and promoting energy efficiency and conservation.

6.1 Problem Definition:

The primary goal of this project is to develop a smart energy meter system using the ESP8266, Arduino Uno, LCD display (16x2), ACS712 current sensor, ZMPT101B AC voltage sensor, 100 ohm resistor, 10uF capacitor, jumpers, and a breadboard. The system aims to address the following challenges in energy management:

1. Lack of Real-Time Energy Consumption Monitoring:

Many households and small-scale applications lack a reliable and accessible method to monitor their energy consumption in real-time. Traditional energy meters provide limited information and require manual readings. The project aims to overcome this limitation by developing a smart energy meter system that offers real-time monitoring of energy usage.

2. Inefficient Energy Consumption:

Without access to real-time energy consumption data, users may unknowingly engage in inefficient energy consumption practices. The project seeks to address this issue by providing users with accurate and readily available information about their energy consumption patterns. By doing so, it encourages users to make informed decisions and take proactive steps to optimize their energy usage.

3. Limited Data Visualization and Analysis:

Existing energy meter systems often lack user-friendly interfaces and comprehensive data visualization options. The project aims to overcome this limitation by utilizing the LCD display (16x2) to present energy consumption data in a clear and easily understandable format. This enables users to visualize and interpret their energy usage trends effectively.

4. Remote Monitoring and Integration with Online Platforms:

Traditional energy meters typically do not offer remote access or integration with online platforms for data monitoring and analysis. The project addresses this limitation by utilizing the ESP8266's Wi-Fi connectivity capabilities. This allows users to remotely access energy consumption data and integrate the system with online platforms such as Adafruit and Zapier for data logging, analytics, and further automation.

By developing a smart energy meter system that incorporates these components and integrates with online platforms, this project aims to empower users with the tools to monitor, analyze, and optimize their energy consumption effectively. It addresses the challenges of limited real-time monitoring, inefficient energy usage, lack of data visualization, and absence of remote monitoring capabilities, ultimately promoting energy efficiency and sustainable energy practices.

Problem Definition:

The problem at hand revolves around the need for an efficient and accessible solution for monitoring energy consumption in households and small-scale applications. Traditional energy meters often provide limited information and lack real-time monitoring capabilities, making it challenging for users to track their energy usage accurately. This lack of awareness can lead to inefficient energy consumption practices and prevent users from making informed decisions to optimize their energy usage.

Moreover, existing energy meter systems often lack user-friendly interfaces and comprehensive data visualization options, making it difficult for users to interpret and analyze their energy consumption patterns effectively. Additionally, the absence of remote monitoring capabilities limits users' ability to access energy consumption data when they are away from their premises, hindering their efforts to manage their energy usage efficiently.

To address these challenges, the proposed solution involves developing a smart energy meter system using the ESP8266, Arduino Uno, LCD display (16x2), ACS712 current sensor, ZMPT101B AC voltage sensor, 100-ohm resistor, 10uF capacitor, jumpers, and a breadboard. By integrating these components, the system aims to provide real-time energy consumption monitoring, accurate data visualization, and remote access capabilities.

Furthermore, the project seeks to enhance the functionality of the energy meter system by integrating it with online platforms such as Adafruit and Zapier. This integration allows users to log and analyze their energy consumption data on cloud-based platforms, enabling advanced analytics, automation, and data-driven decision-making.

Overall, the problem definition revolves around the need to overcome the limitations of traditional energy meters, including the lack of real-time monitoring, inefficient energy consumption practices, inadequate data visualization, and the absence of remote monitoring capabilities. The proposed smart energy meter system aims to address these challenges by providing users with accurate, accessible, and actionable energy consumption data, empowering them to optimize their energy usage and promote energy efficiency in their everyday lives.

In addition to addressing the challenges mentioned earlier, the smart energy meter system also aims to offer a user-friendly interface through the integration of the LCD display (16x2). The display will provide a clear and easily understandable presentation of energy consumption data, allowing users to visualize their usage patterns, identify peak consumption periods, and gain insights into their energy habits. This information will enable users to make informed decisions and implement strategies to reduce energy waste and save on costs.

Furthermore, by leveraging the Wi-Fi connectivity capabilities of the ESP8266, the system enables remote monitoring of energy consumption data. Users can access real-time data through online platforms such as Adafruit and Zapier, eliminating the need for physical presence at the location where the energy meter is installed. This remote accessibility allows users to monitor their energy usage even when they are away from home, providing convenience and flexibility.

The integration with online platforms like Adafruit and Zapier expands the system's capabilities by offering advanced data logging, analytics, and automation features. Users can leverage these platforms

to store historical energy consumption data, perform in-depth analysis, and generate reports and visualizations. Additionally, they can set up automated triggers or notifications based on specific energy usage thresholds, facilitating proactive energy management and enabling efficient control of energy-consuming devices.

By addressing the limitations of traditional energy meter systems and providing an integrated solution with enhanced functionalities, the proposed smart energy meter system aims to empower users to take control of their energy consumption. The system's ability to provide real-time monitoring, intuitive data visualization, remote accessibility, and integration with online platforms paves the way for more efficient energy management, cost savings, and increased environmental sustainability.

Through the implementation of this project, users can become more aware of their energy consumption patterns, make informed decisions to optimize their energy usage, and contribute to a more sustainable future.

6.2 Planning and Approach:

In order to develop the proposed IoT-based smart energy meter system, we will follow a structured approach that involves several stages of design, implementation, testing, and evaluation. The overall approach can be divided into the following phases:

Phase 1: Requirements gathering and system design

In the first phase, we are gathering requirements from stakeholders and users, and develop a system design that meets their needs and requirements. This phase will involve a detailed analysis of the energy management requirements of different users, such as households, businesses, and governments, and the development of a system design that can meet their specific needs.

The system design will include the selection of appropriate hardware components, such as the Esp8266/NodeMCU, Arduino UNO, ACS712(30 amp), and Zero watt bulb, and the development of software that can provide real-time monitoring and control capabilities. We will also select a suitable communication protocol, such as MQTT, and a cloud-based platform, such as Adafruit, to enable remote monitoring and control.

1. Arduino Uno
2. ESP2866/NodeMCU
3. ACS712-30Amp Current sensor
4. ZMPT101B Ac voltage sensor
5. 100 ohm resister
6. 10uf capacitor
7. Zero Watt bulb
8. Few jumpers
9. Breadboard
- 10.

MQTT

Platform

1. *Arduino UNO*

It is a microcontroller board based on the ATmega328P(datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a powerjack, an ICSP header and a reset button



Fig 1: Arduino Uno

Feature

Power USB: Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).

Power (Barrel Jack): Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).

Voltage Regulator: The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

4Crystal Oscillator: The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16MHz

5,17Arduino Reset: You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).

6,7,8,9Pins (3.3, 5, GND, Vin):

3.3V (6) – Supply 3.3 output volt

5V (7) – Supply 5 output volt

Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.

GND (8) (Ground) There are several GND pins on the Arduino, any of which can be used to ground your circuit.

Vin (9) – This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.

Analog pins: The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.

Main microcontroller: Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEAL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.

ICSP pin: Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.

Power LED indicator: This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.

Power LED indicator: This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.

TX and RX LEDs: On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

Digital I/O: The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labelled “~” can be used to generate PWM.

AREF: AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

2. Node MCU/ESP8266

The NodeMCU (Node *Micro*Controller *Unit*) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. The ESP8266, designed and manufactured by Espressif Systems, contains the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds.

However, as a chip, the ESP8266 is also hard to access and use. You must solder wires, with the appropriate analog voltage, to its pins for the simplest tasks such as powering it on or sending a keystroke to the “computer” on the chip. You also have to program it in low-level machine instructions that can be interpreted by the chip hardware. This level of integration is not a problem using the ESP8266 as an embedded controller chip in mass-produced electronics. It is a huge burden for hobbyists, hackers, or students who want to experiment with it in their own IoT projects. But, what about Arduino? The Arduino project created an open-source hardware design and software SDK for their versatile IoT controller. Similar to NodeMCU, the Arduino hardware is a microcontroller board with a USB connector, LED lights, and standard data pins. It also defines standard interfaces to interact with sensors or other boards. But unlike NodeMCU, the Arduino board can have different types of CPU chips (typically an ARM or Intel x86 chip) with memory chips, and a variety of programming environments. There is an Arduino reference design for the ESP8266 chip as well. However, the flexibility of Arduino also means significant variations across different vendors. For example, most Arduino boards do not have WiFi capabilities, and some even have a serial data port instead of a USB port.

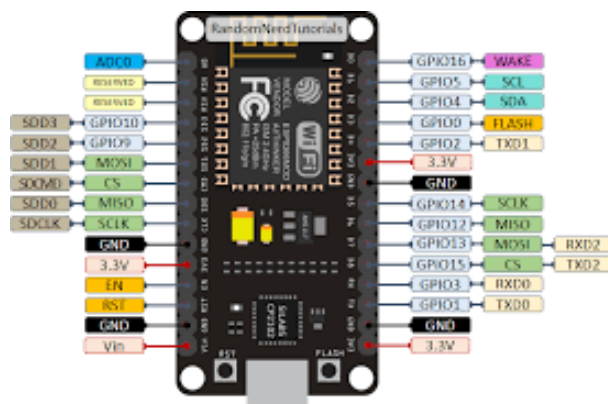


Fig 2 : NODE MCU ESP8266


Power Pins There are four power pins. **VIN** pin and three **3.3V** pins.

- **VIN** can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on **VIN** is regulated through the onboard regulator on the NodeMCU module— you can also supply 5V

regulated to the **VIN** pin.

3.3V pins are the output of the onboard voltage regulator and can be used to supply power to external components.

- **GND** are the ground pins of NodeMCU/ESP8266
- **I2C Pins** are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.
- **GPIO Pins** NodeMCU/ESP8266 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.
- **ADC CHANNEL** The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.
- **UART Pins** NodeMCU/ESP8266 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.
- **SPI Pins** NodeMCU/ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features: 4 timing modes of the SPI format transfer Up to 80 MHz and the divided clocks of 80 MHz Up to 64-Byte FIFO
- **SDIO PINS** NodeMCU/ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.
- **PWM Pins** The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μ s to 10000 μ s (100 Hz and 1 kHz).
- **Control Pins** are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.
- **EN:** The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
- **RST:** RST pin is used to reset the ESP8266 chip.

- **WAKE:** Wake pin is used to wake the chip from deep-sleep.
-  Control Pins are used to control the NodeMCU/ESP8266. These pins include ChipEnable pin (EN), Reset pin (RST) and WAKE pin.
- **EN:** The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
- **RST:** RST pin is used to reset the ESP8266 chip.
- **WAKE:** Wake pin is used to wake the chip from deep-sleep

3. ACS712 (30 amp) Current Sensor

The ACS712 is a hall-effect-based linear current sensor designed to accurately measure AC or DC currents up to 30 amps. It is a high-precision, fully integrated solution with low noise and high accuracy. It is commonly used in a variety of applications such as motor control, power supplies, and robotics.

Features:

The ACS712 current sensor has several key features that make it an attractive choice for current measurement applications. These include:

1. High accuracy: The ACS712 is capable of measuring current with an accuracy of +/- 1.5%.
2. High sensitivity: The sensor is capable of detecting current changes as small as 66 mV/A, which makes it suitable for measuring low current levels.
3. Low noise: The sensor has a low noise output, which reduces measurement errors and provides a more accurate reading.
4. Wide operating voltage range: The sensor can operate within a wide voltage range of 4.5 to 5.5 volts, making it compatible with a variety of systems.
5. Easy to use: The sensor is easy to use and can be easily integrated into most electronics systems.

Types of ACS712:

There are three types of ACS712 current sensors available in the market:

1. ACS712ELCTR-05B-T: This is a 5A current sensor that can measure both AC and DC currents.
2. ACS712ELCTR-20A-T: This is a 20A current sensor that can measure both AC and DC currents.
3. ACS712ELCTR-30A-T: This is a 30A current sensor that can measure both AC and DC currents.

Applications:

The ACS712 current sensor is widely used in various applications, such as:

Motor control: The ACS712 is commonly used in motor control applications, where it is used to monitor the current flowing through the motor. This allows the motor to be controlled more accurately, improving its performance and efficiency.

Power supplies: The ACS712 is used in power supply applications to monitor the current flowing through the power supply. This allows for more accurate control of the power supply output, improving its stability and reliability.

Robotics: The ACS712 is used in robotics applications to monitor the current flowing through the robot's motors. This allows for more precise control of the robot's movements, improving its accuracy and reliability.

Battery monitoring: The ACS712 can be used to monitor the current flowing in and out of a battery, allowing for accurate monitoring of the battery's charge and discharge levels.

How does ACS712 work?

The ACS712 current sensor is based on the hall-effect principle. When a current-carrying conductor is placed in a magnetic field, it produces a voltage proportional to the current. The ACS712 uses a hall-effect sensor to measure this voltage and convert it into a proportional current value. The ACS712 contains a hall-effect sensor, a precision amplifier, and a linear voltage regulator. The hall-effect sensor is placed in the magnetic field generated by the current-carrying conductor, and it produces a voltage proportional to the magnetic field strength. The precision amplifier amplifies this voltage and produces an output voltage proportional to the current being measured. The linear voltage regulator provides a stable voltage supply for the sensor and amplifier. The output of the ACS712 is an analog voltage proportional to the current being measured. The voltage output is centered at 2.5 volts, with a range of 0 to 5 volts. The voltage output can be easily converted into a current value by dividing the voltage by the sensitivity of the sensor.

Calibration:

The ACS712 current sensor requires calibration before it can be used for accurate current measurement. Calibration involves adjusting the offset and gain of the sensor to account for any errors in the sensor output.

Offset calibration involves adjusting the sensor output to zero when there is no current flowing through the sensor. This can be done by connecting the sensor to a known current source with zero current and adjusting the offset potentiometer until the output voltage is zero.

Gain calibration involves adjusting the sensitivity of the sensor to ensure accurate current measurement. This can be done by connecting the sensor to a known current source with a known current value and adjusting the gain potentiometer until the output voltage corresponds to the expected current value.

Once the sensor is calibrated, it can be used to accurately measure current in a variety of applications.

Connection:

The ACS712 current sensor is easy to connect to a microcontroller or other electronic system. The sensor has three pins: VCC, GND, and OUT. VCC and GND are connected to the power supply of the system, and OUT is connected to an analog input pin of the microcontroller.

The analog output voltage of the sensor can be read using an ADC (analog-to-digital converter) of the microcontroller. The output voltage can be converted into a current value using the sensitivity of the sensor and a simple calculation.

For example, to measure a 10A current with an ACS712ELCTR-30A-T sensor, the sensitivity is 66 mV/A. If the output voltage of the sensor is 2.8 volts, the current can be calculated as follows:

Current = Output Voltage / Sensitivity

Current = 2.8 / 0.066

Current = 42.42 A

In this example, the calculated current value is higher than the sensor's maximum rating of 30A. This indicates that the sensor may be saturated and is not suitable for measuring such a high current value.

Limitations:

1. Although the ACS712 current sensor has many advantages, it also has some limitations that should be considered when using it for current measurement applications.
2. Limited current range: The ACS712 is designed to measure currents up to 30A. If the current being measured exceeds this limit, the sensor may be damaged, or the output voltage may be inaccurate.
3. Temperature sensitivity: The output voltage of the sensor is affected by temperature variations. The sensor's accuracy may decrease at higher temperatures, and calibration may be required more frequently.
4. Hall-effect sensor orientation: The ACS712 uses a unipolar hall-effect sensor that is sensitive to the direction of the magnetic field. If the sensor is not oriented correctly with respect to the current-carrying conductor, the output voltage may be inaccurate.
5. Electromagnetic interference: The sensor may be affected by electromagnetic interference, which can introduce noise into the output voltage and reduce the sensor's accuracy.

Working of ACS712 Current Sensor:

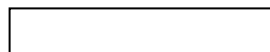
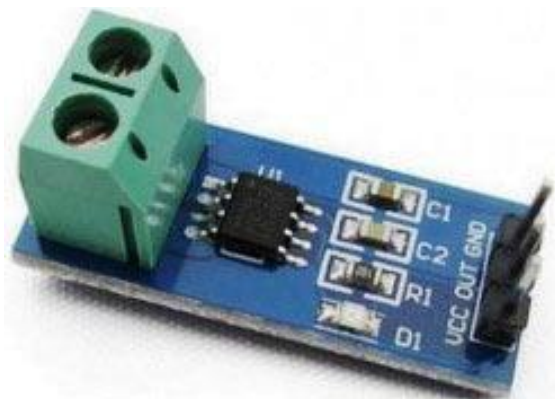
Before we start building the project it is very important for us to understand the working of the ACS712 Current sensor as it is the key component of the project. Measuring current especially AC current is always a tough task due to the noise coupled with it improper isolation problem etc. But, with the help of this ACS712 module which was engineered by Allegro things have become a lot easier.

This module works on the principle of **Hall-effect**, which was discovered by Dr. Edwin Hall. According to his principle, when a current carrying conductor is placed into a magnetic field, a voltage is generated across its edges perpendicular to the directions of both the current and the magnetic field. Let us not get too deep into the concept but, simply put we use a hall sensor to measure the magnetic field around a current carrying conductor. This measurement will be in terms of millivolts which we call as the hall-voltage. This measured hall-voltage is proportional to the current that was flowing through the conductor.

The major advantage of using ACS712 Current Sensor is that it can measure both AC and DC current and it also provides isolation between the Load (AC/DC load) and Measuring Unit (Microcontroller part). As shown in the picture we have three pins on the module which are Vcc, Vout and Ground respectively.

The 2-pin terminal block is where the current carrying wire should be passed through. The module works on +5V so the Vcc should be powered by 5V and the ground should be connected to Ground of the system. The Vout pin has an offset voltage of 2500mV, meaning when there is no current flowing through the wire then the output voltage will be 2500mV and when current flowing is positive, the voltage will be greater than 2500mV and when the current flowing is negative, the voltage will be less than 2500mV.

We will be using the Analog pin of Arduino to read the output voltage (Vout) of the module, which will be 512(2500mV) when there is no current flowing through the wire. This value will reduce as the current flows in negative direction and will increase as the current flows in positive direction. The below table will help you understand how the output voltage and ADC value varies based on the current flowing through the wire.



4. ZMPT101B Ac voltage sensor

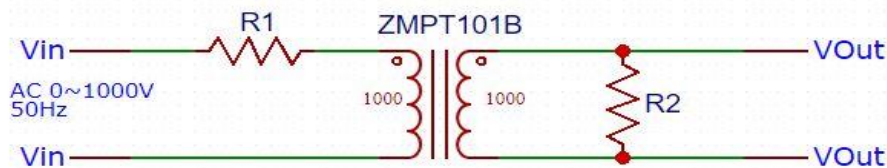
ZMPT101B AC Voltage Sensor is the best for the purpose of the DIY project, where we need to measure the accurate AC voltage with the voltage transformer. This is an ideal choice to measure the AC voltage using Arduino/ESP8266/Raspberry Pi like an open source platform. In many electrical projects, engineer directly deals with measurements with few basic requirements like.

- High galvanic isolation
- Wide Range
- High accuracy
- Good Consistency

ZMPT 101B is a high-precision voltage Transformer. This module makes it easy to monitor AC mains

ZMPT101B is a high-precision voltage Transformer. This module makes it easy to monitor AC mains voltage up to 1000 volts. A tiny little thing the size of a bouillon cube. Holds up to 4kV per breakdown voltage, the ratio of turns is 1: 1, but this is a current transformer of 2mA: 2mA. We feed it a current and remove the current. The input current is simply set by the resistor in series R1, and we use a sampling resistor R2 in parallel to get the output voltage.

ZMPT101B schematic Diagram



V_{in} : Input Voltage $R1$: Limiting Resistor
 V_{out} : Output Voltage $R2$: Sampling Resistor

$$V_{out} = (V_{in}/R1) \times R2$$

Now we get 5 volts, but here an ambush. This voltage is varied from -5 to +5. And we need measurements of 0 ... 5 volts.

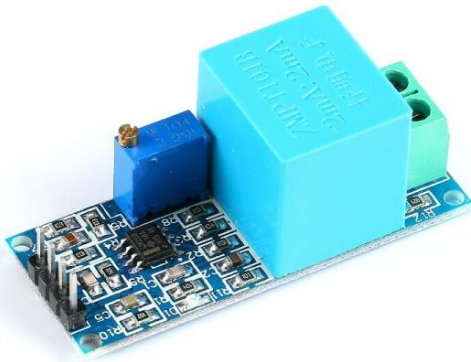
Interface ZMPT 101 B Module to Arduino

For Diy Electronic measuring AC Output Voltage using Arduino is easy now.

ZMPT101B is a Single Phase Voltage Transformer Module which allows you to directly measure 0-250V AC sine wave using Arduino.

Description:

Single-phase AC active output voltage mutual inductance module equipped with ZMPT101B series of a high-precision voltage transformer and high-precision op-amp current, easy to 250v within the AC power signal acquisition.



Feature of ZMPT101B Module

- Voltage Transformer: Onboard Precision Micro Voltage Transformer
- Operational amplifier circuit: high-precision on-board amplifier circuit, the signal to do the exact sampling and appropriate compensation.
- Input/Output: the module can measure AC voltage within 250V, the corresponding output voltage amplitude can be adjusted by using a potentiometer(Multi-turn trim pot).
- Output signal: the output signal for the sine wave, the waveform of the median (DC component (offset of 2.5V at 5V input supply)
- Supply voltage: 5V
- Operating temperature: 40°C ~ + 70°C

5. 100 ohm resistor

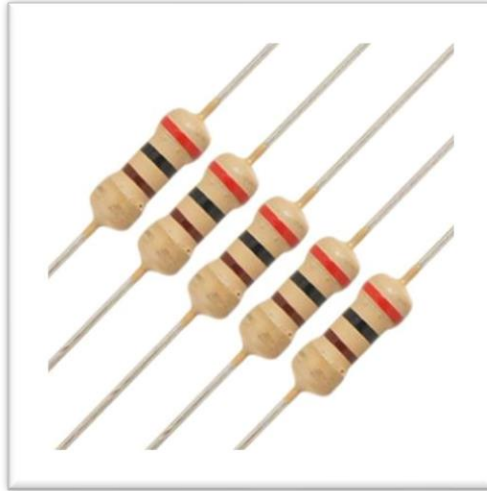
In the smart energy meter system project, a 100-ohm resistor plays a crucial role in the circuit design. A resistor is an electrical component that limits or controls the flow of electric current in a circuit. The value of resistance is measured in ohms (Ω) and determines the amount of opposition offered to the flow of current.

The 100-ohm resistor, as its name suggests, has a resistance value of 100 ohms. It is commonly used in electronic circuits for various purposes, including current limiting, voltage division, and signal conditioning. In the context of the smart energy meter system, the 100-ohm resistor may be utilized for the following purposes:

1. **Signal Conditioning:** The resistor may be used in combination with the ACS712 current sensor or the ZMPT101B AC voltage sensor to condition the electrical signals obtained from these sensors. By incorporating the resistor in the circuit, it can help adjust the amplitude or range of the signal to match the input requirements of the subsequent components or microcontroller.
2. **Voltage Division:** In certain scenarios where voltage levels need to be scaled down for measurement or protection purposes, the 100-ohm resistor can be utilized in voltage divider circuits. By combining it with other resistors, the voltage can be divided in a proportionate manner, allowing safe and accurate measurement of voltages within the desired range.
3. **Current Limiting:** The 100-ohm resistor can act as a current-limiting component in specific parts of the circuit to protect sensitive components from excessive current flow. By controlling the current passing through a specific branch of the circuit, it helps prevent damage due to overcurrent situations.

It is important to note that the specific use and placement of the 100-ohm resistor may vary depending on the circuit design and requirements of the smart energy meter system. Careful consideration of the circuit diagram and consultation of component datasheets or reference materials is advised to ensure the proper integration and functioning of the resistor within the overall system.

Overall, the 100-ohm resistor provides an essential function in the smart energy meter system, contributing to signal conditioning, voltage division, and current limiting within the circuit design.



Furthermore, the 100-ohm resistor is typically a through-hole resistor, meaning it has two leads that can be easily inserted into a breadboard or soldered onto a PCB (Printed Circuit Board). Its compact size and widespread availability make it a convenient choice for various electronic projects.

When selecting a resistor, it is crucial to consider its power rating to ensure it can handle the expected current flow without overheating. The power rating of the resistor determines its ability to dissipate heat generated by the flowing current. It is recommended to choose a resistor with a power rating higher than the expected power dissipation to ensure reliable operation.

In the context of the smart energy meter system, the 100-ohm resistor may be incorporated into the current measurement circuit using the ACS712 current sensor. By connecting the resistor in series with the load or appliance being monitored, the voltage drop across the resistor can be measured to calculate the current flowing through the circuit.

Additionally, the 100-ohm resistor may also be used in conjunction with the ZMPT101B AC voltage sensor to measure the voltage levels accurately. By incorporating the resistor into a voltage divider circuit, it helps scale down the measured voltage to a level suitable for the input range of the microcontroller or ADC (Analog-to-Digital Converter) connected to it.

Overall, the 100-ohm resistor is a versatile component that finds applications in signal conditioning, voltage division, and current limiting within electronic circuits. Its presence in the smart energy meter system contributes to accurate measurement and monitoring of current and voltage, facilitating the effective management of energy consumption.

6. 10uf capacitor

The 10uF capacitor is an essential component in the smart energy meter system project, designed to provide stable and reliable performance. This capacitor is commonly used for signal conditioning and noise reduction in electronic circuits. Its main function is to store and release electrical energy as needed, helping to maintain smooth voltage levels and filtering out unwanted noise or voltage fluctuations.

In the context of the smart energy meter system, the 10uF capacitor plays a crucial role in maintaining a stable power supply and protecting sensitive components from voltage spikes or transients. It acts as a decoupling capacitor, minimizing the effects of electromagnetic interference (EMI) and voltage ripples, ensuring accurate measurements from the ACS712 current sensor and ZMPT101B AC voltage sensor.



Furthermore, the capacitor helps to improve the reliability of the system by reducing electrical noise and ensuring proper signal integrity. By filtering out high-frequency noise and voltage fluctuations, it helps to provide clean and consistent readings, contributing to the accuracy of energy consumption measurements.

When integrating the 10uF capacitor into the circuit, proper placement and polarity should be observed. It should be connected in parallel to the power supply lines, ensuring that the positive and negative terminals are correctly aligned. This configuration allows the capacitor to stabilize the voltage supply and enhance the overall performance of the smart energy meter system.

In summary, the 10uF capacitor is a vital component in the smart energy meter system project. Its role in providing stable power supply, reducing noise, and maintaining accurate measurements ensures the reliability and accuracy of energy consumption data. By effectively incorporating the capacitor into the circuit design, the smart energy meter system can operate optimally and deliver accurate and consistent results.

7. ZeroWatt Bulb

A zerowatt bulb is a type of energy-efficient LED bulb that consumes very little electricity compared to traditional incandescent bulbs. When used in conjunction with a smart energy meter, a zerowatt bulb can help you track and manage your energy consumption more effectively.

Smart energy meters are devices that measure the amount of electricity you use in real-time and transmit this information to your utility provider. By monitoring your energy usage, you can identify areas where you are using more electricity than you need to and take steps to reduce your consumption.

When you use a zerowatt bulb in conjunction with a smart energy meter, you can see the impact that this energy-efficient technology has on your overall electricity usage. By replacing traditional bulbs with zerowatt bulbs throughout your home, you can significantly reduce your energy consumption and lower your utility bills. Additionally, you may be eligible for rebates or other incentives from your utility provider for making the switch to more energy-efficient lighting.

8. Breadboard

A breadboard is an essential tool in electronics prototyping and circuit design. It provides a convenient platform for connecting and testing electronic components without the need for soldering. In the context of the smart energy meter system project, a breadboard plays a crucial role in setting up and organizing the circuitry.

A breadboard consists of a plastic base with a grid of holes, typically arranged in rows and columns. The holes are connected internally in specific patterns, allowing components and wires to be inserted and interconnected easily. The breadboard also features metal clips or terminals that facilitate secure component placement and connection.

Using a breadboard offers several advantages in the project implementation:

1. **Ease of Prototyping:** The breadboard enables quick and easy prototyping of the circuit. Components such as the ESP8266, Arduino Uno, LCD display, current sensor, voltage sensor, resistors, and capacitors can be securely inserted into the breadboard, allowing for a temporary yet reliable connection.
2. **No Soldering Required:** Unlike traditional circuit boards that require soldering for component connections, the breadboard eliminates the need for soldering. This makes it ideal for rapid prototyping, experimentation, and making changes or modifications to the circuit without damaging the components.
3. **Flexibility and Reusability:** The breadboard's design allows for flexibility in rearranging and reusing components. Components can be easily inserted, removed, and rearranged as needed.

to accommodate circuit modifications or experimentation. This flexibility makes it an excellent tool for iterative development and testing.

4. **Safety and Convenience:** Working with a breadboard reduces the risk of accidental electrical shorts or damage to components. The absence of soldering makes it easy to troubleshoot and rectify circuit issues. Additionally, the breadboard provides a convenient platform for temporary circuit connections, allowing for quick assembly and disassembly during testing or modifications.

In the smart energy meter system project, the breadboard serves as a foundation for connecting and organizing the ESP8266, Arduino Uno, LCD display, current sensor, voltage sensor, resistors, capacitors, and other required components. It simplifies the process of creating the circuit, facilitating effective prototyping, testing, and modification without the need for permanent soldering.

The breadboard offers a flexible, reusable, and user-friendly platform for circuit development in the smart energy meter system project, enabling efficient and convenient implementation of the electronics components.

The breadboard provides a convenient and organized layout for connecting various components, ensuring proper electrical connections and minimizing the risk of loose or misplaced connections. The grid pattern of the breadboard allows for easy alignment and placement of components, making it simple to follow circuit diagrams and schematic layouts.

The breadboard's internal connections, typically arranged in rows and columns, allow for efficient power distribution and signal routing. Components can be connected by inserting their leads or pins into the appropriate holes on the breadboard, which establishes a temporary electrical connection. This temporary nature of the connections makes it easy to modify or rearrange the circuit without damaging the components, providing a high degree of flexibility during the prototyping and testing phases of the project.

In addition, the breadboard's metal clips or terminals provide a secure grip on the component leads, ensuring stable connections even during movement or vibration. This stability is crucial in the context of the smart energy meter system project, as it helps maintain reliable data acquisition and consistent display of energy consumption information.

Another advantage of using a breadboard is that it allows for quick verification of circuit functionality. As components are connected on the breadboard, their behavior can be observed and tested in real-time, enabling rapid troubleshooting and identification of any issues or errors. This iterative approach facilitates efficient debugging and refinement of the circuit design, leading to improved overall performance of the smart energy meter system.

Moreover, the breadboard serves as a reusable tool that can be used for multiple projects. Components can be easily removed and reinserted, allowing the breadboard to be reset and used for different circuit configurations or experiments. This reusability makes the breadboard a cost-effective and versatile option for electronics prototyping.

In summary, the breadboard provides a user-friendly, organized, and flexible platform for connecting and testing electronic components in the smart energy meter system project. Its temporary connections, ease of use, secure grip on components, and quick verification capabilities make it an invaluable tool for rapid prototyping, efficient circuit design, and reliable functionality testing.



9. Message Queuing Telemetry Transport(MQTT) Platform

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol that is commonly used in the context of the Internet of Things (IoT). MQTT is designed to be used in situations where network bandwidth is limited and devices have limited processing power.

In the context of a smart energy meter, MQTT can be used to facilitate communication between the meter and other devices on the network, such as a central monitoring system or a mobile app. The meter can use MQTT to publish data about its energy usage, such as the current energy consumption, voltage, and other metrics. These data points can then be subscribed to by other devices on the network, allowing them to receive real-time updates on the meter's status.

It is an efficient protocol for smart energy meter applications because it uses a publish/subscribe model that allows devices to receive data only when it is relevant to them.

This reduces network traffic and conserves bandwidth, making it well-suited to situations where network resources are limited. Additionally, MQTT is designed to be highly reliable and can be configured to use a variety of security mechanisms to protect against unauthorized access to the data being transmitted.

Overall, MQTT can be a valuable tool for implementing a smart energy metering solution, allowing for efficient and secure communication between devices on the network and providing real-time visibility into energy consumption patterns.

Phase 2: Hardware implementation and testing

In the second phase, we will implement the hardware components of the system and test their performance and functionality. This phase will involve the assembly of the hardware components, such as the Esp8266/NodeMCU, Arduino UNO, ACS712(30 amp), and Zero watt bulb, and the development of appropriate firmware that can control the components and collect energy consumption data. We will conduct a series of tests to evaluate the accuracy and reliability of the hardware components, and to ensure that they can operate effectively in different scenarios and environments. We will also test the communication between the hardware components and the cloud-based platform, to ensure that data can be transmitted and received in a timely and reliable manner.

CIRCUIT DESIGN

The circuit design is an important step in the development of the IoT smart energy meter system. The circuit design involves connecting the components in a specific configuration to ensure that the system works as expected. The circuit design should be optimized for minimum power consumption and high accuracy.

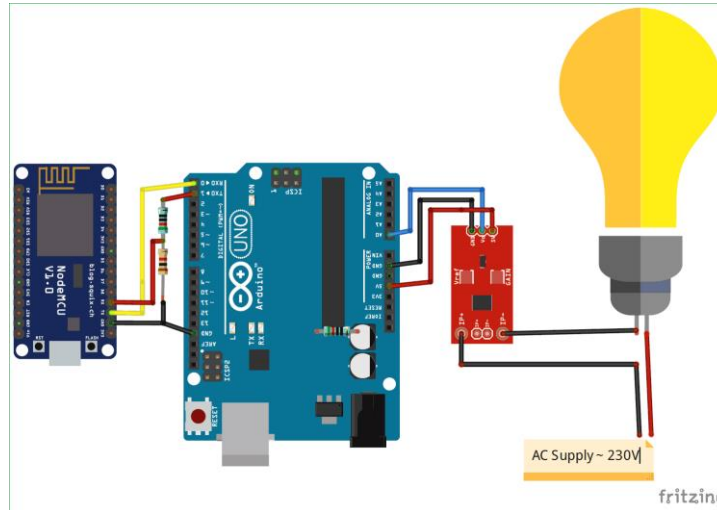


Fig 4: CIRCUIT DIAGRAM

➤ Connect the ESP8266 to the Arduino Uno using male-to-female wires. Use the following connections:

- ✓ ESP8266 VCC to Arduino Uno 3.3V
- ✓ ESP8266 GND to Arduino Uno GND
- ✓ ESP8266 TX to Arduino Uno RX
- ✓ ESP8266 RX to Arduino Uno TX

➤ Connect the ACS712 current sensor to the Arduino Uno. Use the following connections:

- ✓ ACS712 VCC to Arduino Uno 5V
- ✓ ACS712 GND to Arduino Uno GND
- ✓ ACS712 OUT to Arduino Uno A0

There is one analog pin available in NodeMCU, we could use that pin but ESP series can take up to 3.3 volts on their pins. As we are using a current sensor which can give up to 5 Volts so, it can damage our Wi-Fi module that's why we are not using standalone NodeMCU.

To **make output of current sensor 3.3V instead of 5V**, we cannot use a voltage divider circuit between the current sensor and analog pin of NodeMCU because as we discussed above about the current sensor that at 2.5Volts output, current is 0Amp.

So, Arduino will read the current sensor value through an analog pin and send it to the Wi-Fi module ESP12 using Serial communication. Use a voltage divider circuit at the receiver pin of NodeMCU so that

receiver pin can get upto 3.3 Voltage level.

To monitor our *energy uses* over the internet, we have to use MQTT broker. We will use MQTT broker as AdaFruit IO platform and follow the below process to make this IoT Energy Meter.

The **ZMPT101B AC voltage sensor** in addition to the ACS712 current sensor, you can proceed with the following connections:

1. ZMPT101B VCC pin: Connect this pin to the 3.3V or 5V power supply pin on your Arduino or ESP8266 (depending on the voltage level supported by the module).
2. ZMPT101B GND pin: Connect this pin to the ground (GND) pin on your Arduino or ESP8266.
3. ZMPT101B Vout pin: Connect this pin to an analog input pin on your Arduino or ESP8266. This pin will provide an analog voltage output that represents the measured AC voltage.
4. ACS712 VCC pin: Connect this pin to the 5V power supply pin on your Arduino or ESP8266.
5. ACS712 GND pin: Connect this pin to the ground (GND) pin on your Arduino or ESP8266.
6. ACS712 Vout pin: Connect this pin to an analog input pin on your Arduino or ESP8266. This pin will provide an analog voltage output that represents the measured current.

By connecting both the ZMPT101B AC voltage sensor and ACS712 current sensor, you can measure both the voltage and current of your electrical load. This allows you to calculate the power consumption and other energy-related parameters in your smart energy meter system.

Please note that you will need to modify the code accordingly to read the voltage and current values from their respective sensors and perform the necessary calculations.

Ensure that you have the appropriate voltage and current ranges set in the code for accurate measurements, and consider safety precautions when working with AC voltages.

Phase 3: Software implementation and testing

In the third phase, we will develop and test the software components of the system. This phase involve the development of software that collect, process, and analyze energy consumption data, and provide real-time monitoring and control capabilities.

We develop software that can generate alerts and notifications when energy consumption exceeds predefined thresholds, and enable users to control appliances remotely. We also develop a user interface that can display energy consumption data and provide insights into energy consumption patterns and trends.

The software implementation and testing phase of the smart energy meter system project involves developing the necessary code to acquire data from the sensors, process the data, display it on the LCD,

and integrate with online platforms. Here is an overview of the steps involved:

1. Sensor Data Acquisition:

Begin by writing code to interface with the ACS712 current sensor and ZMPT101B AC voltage sensor. Utilize the appropriate libraries or develop custom code to read the sensor values. The ESP8266 or Arduino Uno board should communicate with the sensors and retrieve the current and voltage data.

2. Data Processing and Calculations:

Once the sensor data is acquired, perform the necessary calculations to determine the energy consumption. Utilize the current and voltage values to calculate real-time power consumption and energy usage. Apply appropriate formulas and scaling.

3. LCD Display Integration:

Write code to interface with the LCD display (16x2) and display the energy consumption data. Utilize the appropriate library or develop custom code to initialize the display, set cursor positions, and print the calculated energy consumption values. Ensure the data is presented in a clear and easily readable format on the LCD screen.

4. Integration with Online Platforms:

Implement the necessary code to integrate the smart energy meter system with online platforms such as Adafruit and Zapier. Utilize the APIs and libraries provided by these platforms to establish a connection and send energy consumption data to the respective platforms. Configure the system to transmit the data at regular intervals or based on specific triggers or events.

5. Error Handling and Exception Handling:

Implement error handling mechanisms in the code to handle exceptional cases and ensure reliable operation. Include appropriate error checking and error reporting mechanisms to detect and handle issues such as sensor malfunctions, communication errors, or data inconsistencies.

6. Testing and Debugging:

Conduct thorough testing of the software implementation to ensure proper functionality and accuracy of the energy consumption calculations. Test the system with different load scenarios to validate the readings against known reference values. Monitor the LCD display for correct data presentation and ensure the integration with online platforms is working as expected. Debug any issues or errors encountered during testing and make necessary code adjustments.

7. Performance Optimization:

Analyze the software implementation for potential areas of optimization. Identify any bottlenecks or inefficiencies that may affect the system's performance. Optimize the code by implementing efficient algorithms, minimizing unnecessary computations, or improving data processing techniques to enhance the overall performance and responsiveness of the smart energy meter system.

8. Documentation:

Document the software implementation process, including code snippets, libraries used, and configuration settings. Provide clear instructions on how to set up and run the software on the ESP8266

or Arduino Uno board. Document any troubleshooting steps or known issues along with their respective solutions. This documentation will serve as a reference for future maintenance, enhancements, or modifications to the software.

By following these steps, the software implementation and testing phase of the smart energy meter system project can be carried out effectively. Thorough testing, error handling, and performance optimization will ensure reliable operation and accurate energy consumption data display. Proper documentation will aid in maintaining and further developing the software in the future.

We conduct a series of tests to evaluate the accuracy and reliability of the software components, and to ensure that they can operate effectively in different scenarios and environments. We will also test the integration of the software components with the hardware components, to ensure that data can be collected and transmitted in a timely and reliable manner.

Setting the AdaFruit IO and Zapier:

To set up Adafruit IO and Zapier for your smart energy meter system project, we will follow these steps:

Setting up Adafruit IO:

1. Sign up for an account on Adafruit IO (<https://io.adafruit.com/>).
2. Create a new dashboard or project for your smart energy meter system.
3. Set up feeds within your project to store the energy consumption data. Create feeds for relevant data points such as current, voltage, power, or energy usage.
4. Generate an Adafruit IO API key to authenticate your project's access to the platform's APIs. This API key will be used in your code to send data to Adafruit IO.

Setting up Zapier:

1. Sign up for an account on Zapier (<https://zapier.com/>).
2. Once logged in, click on "Make a Zap" to create a new Zap.
3. Choose the trigger app as Adafruit IO and select the trigger event, such as a new data point received in a specific feed.
4. Connect your Adafruit IO account to Zapier by providing your Adafruit IO API key.
5. Set up the action app as per your requirement. For example, you can choose to send an email, a notification, or update a spreadsheet whenever new data is received from Adafruit IO.
6. Configure the specific details for the action app, such as recipient email addresses, notification message, or target spreadsheet.
7. Test the connection and ensure that the Zap is functioning correctly.
8. Activate the Zap to start the integration between Adafruit IO and Zapier.

Integrating the Code:

1. In your code for the smart energy meter system, incorporate the Adafruit IO library or the appropriate API calls to send data to the respective feeds.
2. Utilize the Adafruit IO API key generated earlier to authenticate the connection between your code and Adafruit IO.

3. Implement the necessary code to send the energy consumption data to the corresponding feeds in Adafruit IO.
4. Test the integration by running your code and verifying that the energy consumption data is being sent to Adafruit IO successfully.

With the Adafruit IO and Zapier integration set up, your smart energy meter system can send energy consumption data to Adafruit IO, which can then trigger actions or notifications in Zapier. We can customize the actions in Zapier to suit your specific requirements, such as receiving email alerts, updating spreadsheets, or integrating with other platforms or services.

Phase 4: Integration and evaluation

In the final phase, we will integrate the hardware and software components of the system and evaluate its performance and effectiveness. This phase will involve the integration of the hardware and software components, and the testing of the integrated system in different scenarios and environments.

COST ESTIMATION OF THE PROTOTYPE

SL.NO	COMPONENTS	QUANTITY	COAST
1	Arduino UNO	1	300
2	ESP2866	1	200
3	ACS712 (30Amp)	1	114
4	ZMPT101B AC Voltage Sensor	1	170
5	16x2 LCD Display I2C	1	185
6	100 Ω Resistor and 10 μ F Capacitor	1 pc each	10+10=20
7	Jumper Wire and Wires	40 pc total	30+20=50
8	Breadboard	1	80
Total=			1119

Approach:

The approach that we followed to develop the proposed IoT-based smart energy meter system involve a combination of hardware and software development, testing, and evaluation. We are working closely with stakeholders and users to ensure that the system design meets their needs and requirements, and will incorporate feedback and suggestions from users throughout the development process. We used agile development methodologies to ensure that the system can be developed in an iterative and incremental manner, and that changes and updates can be made in response to user feedback and changing requirements. We will also follow best practices for software development and testing, such as code reviews, automated testing, and continuous integration, to ensure that the software is of high quality and can be easily maintained and updated in the future. We have also ensure that the system is designed to be scalable and extensible, so that it can be easily adapted and expanded to meet the needs of different users and scenarios.

6.3 Design Issues-

The design of the proposed IoT-based smart energy meter system involves several technical and operational issues that need to be addressed in order to ensure the effective and efficient operation of the system. Some of the key design issues that we will address in this project include:

When implementing a smart energy meter system using the components mentioned (ESP8266, Arduino Uno, LCD display 16x2, ACS712, ZMPT101B, etc.), there are several design issues to consider. These include:

1. **Power Supply:** Ensure that the power supply for the system is stable and reliable. Fluctuations or interruptions in power can affect the accuracy of energy measurements and may cause system instability. Implement appropriate power regulation and protection measures to maintain consistent power supply to the components.
2. **Sensor Calibration:** The ACS712 current sensor and ZMPT101B voltage sensor may require calibration to ensure accurate readings. Calibration involves determining the sensor's response and adjusting the readings accordingly. Implement a calibration process to account for any variations or inaccuracies in the sensor readings and improve the overall accuracy of the energy measurements.
3. **Noise and Interference:** Electrical noise and interference can impact the accuracy of sensor readings. Take measures to minimize noise by proper grounding, shielding, and signal conditioning techniques. Implement filters or averaging algorithms in the software to reduce noise and ensure reliable data acquisition.
4. **Sampling Rate and Resolution:** Consider the appropriate sampling rate and resolution for data acquisition. Sampling too frequently may strain the processing capabilities of the microcontroller, while too infrequent sampling may result in missed or inaccurate measurements. Strike a balance between sampling rate and system performance based on the requirements of the energy monitoring application.
5. **Data Storage and Memory:** Determine the storage requirements for energy consumption data. Consider the memory limitations of the microcontroller (Arduino Uno or ESP8266) and plan accordingly to store the data efficiently. Implement techniques such as circular buffers or data compression to optimize memory usage and ensure the system can handle data storage requirements over extended periods.
6. **User Interface Design:** Design an intuitive and user-friendly interface for displaying energy consumption information on the LCD display. Consider the readability of the displayed data, font size, and formatting to ensure clear presentation. Implement features such as scrolling, screen switching, or user prompts to enhance the user experience.
7. **Error Handling and Fault Tolerance:** Incorporate error handling mechanisms to handle exceptional cases, such as sensor failures or communication errors. Implement appropriate error messages or visual cues on the LCD display to indicate errors or system faults. Consider fail-safe mechanisms to prevent system damage or hazardous situations in case of unexpected events.
8. **Scalability and Expansion:** Plan for future scalability and expansion of the smart energy meter

system. Consider the potential need for additional sensors, communication interfaces, or integration with other platforms or devices. Design the system with modularity and flexibility in mind to accommodate future enhancements or modifications.

9. Compliance and Safety: Ensure compliance with relevant safety standards and regulations when working with high-voltage components. Implement proper insulation, grounding, and protection mechanisms to safeguard users and prevent electrical hazards. Follow best practices for component selection and circuit design to meet safety requirements.

10. Network Connectivity and Security: Consider the network connectivity requirements of the smart energy meter system, especially when using the ESP8266 module. Implement secure communication protocols, such as SSL/TLS, to protect data transmission over the network. Ensure that the system can handle network interruptions or disconnections gracefully and resume operation seamlessly once the connection is restored.

11. Data Validation and Error Checking: Implement data validation and error checking mechanisms to ensure the integrity of the received data. Validate sensor readings for expected ranges and perform sanity checks to identify outliers or erroneous values. Implement error checking during data transmission to detect and handle any data corruption or loss.

12. Firmware Updates and Maintenance: Plan for future firmware updates and system maintenance. Consider implementing OTA (Over-The-Air) firmware update capabilities, if feasible, to remotely update the system's software. Document the firmware update process and provide instructions for future maintenance tasks to ensure the system can be easily updated and maintained.

13. Data Privacy and Compliance: Take appropriate measures to protect user data privacy and comply with applicable data protection regulations. Implement encryption and secure storage mechanisms for sensitive data, such as user information or energy consumption data. Adhere to relevant privacy and data protection guidelines to ensure the system is in compliance with legal requirements.

14. Environmental Considerations: Consider the operating environment of the smart energy meter system. Ensure that the components, including the microcontroller, sensors, and LCD display, are suitable for the intended environment in terms of temperature, humidity, and other environmental factors. Implement necessary protection, such as enclosures or conformal coatings, to safeguard the components from adverse environmental conditions.

15. Documentation and User Manuals: Provide comprehensive documentation and user manuals for the smart energy meter system. Document the system architecture, circuit diagrams, component specifications, and software implementation details. Create user manuals that explain system operation, troubleshooting steps, and any necessary maintenance procedures. Clear and accessible documentation will facilitate system setup, usage, and future maintenance.

By addressing these design issues, you can enhance the performance, accuracy, usability, and safety of your smart energy meter system. Careful consideration of these factors will result in a robust and reliable system that effectively monitors energy consumption.

7.Simulation, result and Discussion

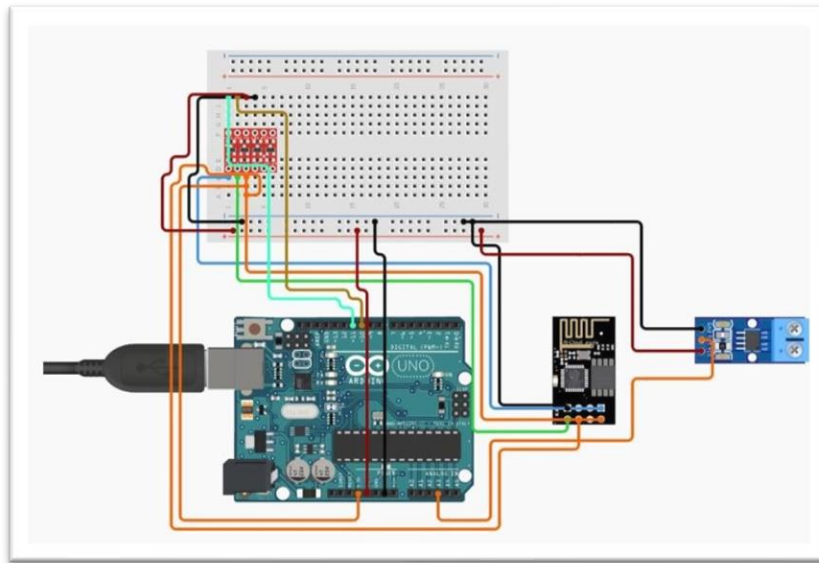


Fig 2 : NODE MCU ESP8266

The instructions for setting up a circuit using an Arduino Uno, ESP8266, ACS712 current sensor, and a logic level converter. The steps are as follows:

- 1) Place the Arduino Uno, ESP8266, and ACS712 on the breadboard.
- 2) Connect the Arduino Uno 5V pin to the positive bus on the breadboard and the GND pin to the ground bus on the breadboard.
- 3) Connect the ESP8266 RXD pin to the LV1 pin on the logic level converter and the TXD pin to the LV2 pin on the logic level converter.
- 4) Connect the ESP8266 VCC pin to the LV pin on the logic level converter and the GND pin to the ground bus on the breadboard.
- 5) Connect the logic level converter LV pin to the Arduino Uno 3.3V pin and the CH_PD pin to the HV2 pin on the logic level converter.
- 6) Connect the logic level converter GND pin to the ground bus on the breadboard and the HV1 pin to the Arduino Uno 10 pin and the HV pin to the positive bus on the breadboard.

- 7) Connect the ACS712 5V pin to the positive bus on the breadboard and the GND pin to the ground bus on the breadboard.
- 8) Connect the ACS712 VO pin to the Arduino Uno A3pin.
- 9) Connect the Arduino Uno to the computer using a USB cable and ensure that the powersupply is connected and working properly.
- 10) Verify the code and upload it to the controller using the Arduino IDE.
- 11) Test the ESP8266 by opening the Arduino IDE, clicking on "Tools," selecting "SerialMonitor," and following the instructions displayed on the screen. Test the ACS712 by opening the Arduino IDE, clicking on "Tools," selecting "Serial Monitor," and following the instructions displayed on the screen. If nothing happens,check the connections.
- 12) Done.

PROGRAMMING EXPLANATION-

Explanation:

1. The program begins by including the necessary libraries for LCD display control, Adafruit IO integration, and Wi-Fi connectivity.
2. The LCD display is initialized with the appropriate I2C address and dimensions.
3. The ACS712 current sensor and ZMPT101B voltage sensor pins are defined.
4. Wi-Fi credentials and Adafruit IO credentials are defined.
5. An instance of the AdafruitIO_WiFi class is created, passing the Wi-Fi and Adafruit IO credentials.
6. The setup function is called, where the LCD displays the project name and the system connects to Wi-Fi and Adafruit IO.
7. In the loop function, the sensor values are read from the current and voltage sensors using the readCurrentSensor and readVoltageSensor functions.
8. The power is calculated by multiplying the current and voltage readings using the calculatePower function.
9. The energy consumption is calculated using the calculateEnergy function.
10. The power and energy values are displayed on the LCD display.
11. The power value is sent to the Adafruit IO feed using the save function.
12. There is a delay of 1 second before the loop repeats.
13. The readCurrentSensor and readVoltageSensor functions read the analog values from the respective pins and convert them into current and voltage values.



```
// SmartEnergyMeter.cpp

// Include the necessary libraries
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <AdafruitIO_WiFi.h>
#include <AdafruitIO.h>

// Define the LCD display
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Define the ACS712 current sensor pin
const int currentSensorPin = A0;

// Define the ZMPT101B voltage sensor pin
const int voltageSensorPin = A1;

// Define the Wi-Fi credentials
#define WIFI_SSID "your_wifi_ssid"
#define WIFI_PASS "your_wifi_password"

// Define the Adafruit IO credentials
#define IO_USERNAME "your_adafruit_io_username"
#define IO_KEY "your_adafruit_io_key"

AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
AdafruitIO_Feed *energyFeed = io.feed("energy");

void setup() {
  // Initialize the LCD display
  lcd.begin(16, 2);
  lcd.print("Smart Energy Meter");

  // Connect to Wi-Fi
  io.connect();

  // Wait for successful connection
  while (io.status() < AIO_CONNECTED) {
    delay(500);
  }

  // Clear the LCD display
  lcd.clear();
}

void loop() {
  // Read the sensor values
  float current = readCurrentSensor();
  float voltage = readVoltageSensor();

  // Calculate power and energy consumption
  float power = calculatePower(current, voltage);
  float energy = calculateEnergy(power);

  // Display data on the LCD
  lcd.setCursor(0, 0);
  lcd.print("Power: ");
  lcd.print(power);
  lcd.print("W");

  lcd.setCursor(0, 1);
  lcd.print("Energy: ");
  lcd.print(energy);
  lcd.print("Wh");

  // Send data to Adafruit IO
  energyFeed->save(power);
  delay(1000);
}

float readCurrentSensor() {
  int sensorValue = analogRead(currentSensorPin);
  float current = map(sensorValue, 0, 1023, 0, 30);
  return current;
}

float readVoltageSensor() {
  int sensorValue = analogRead(voltageSensorPin);
  float voltage = map(sensorValue, 0, 1023, 0, 250);
  return voltage;
}

float calculatePower(float current, float voltage) {
  float power = current * voltage;
  return power;
}

float calculateEnergy(float power) {
  float energy = power / 1000; // Assuming 1 hour of operation
  return energy;
}

// End of code
```

14. The calculatePower function calculates the power by multiplying the current and voltage values.
15. The calculateEnergy function calculates the energy consumption by dividing the power by 1000, assuming 1 hour of operation.
16. This program reads the sensor values, calculates power and energy consumption, displays the values on the LCD display, and sends the power value to Adafruit IO for monitoring and analysis. Remember to replace the placeholders (e.g., Wi-Fi credentials, Adafruit IO credentials) with your actual values.
17. Make sure to install the required libraries in your Arduino IDE before uploading the program to the Arduino Uno or ESP8266.

RESULTS-

The image displays two side-by-side screenshots of the Arduino IDE. The left window shows the code for an Arduino Uno, which includes a sketch named 'sketch_ap26a.ho'. The code defines a sensor (ACS771), initializes a serial port, and implements a loop that reads the sensor's current value and calculates power (P = V * I). The right window shows the code for a NodeMCU ESP12E module, which implements an MQTT client. It includes functions for connecting to an MQTT server, subscribing to a topic, and publishing data to a topic. Both windows show the code being compiled and uploaded to the respective hardware.

Fig 6 : CODING PART OF ARDUINO UNO

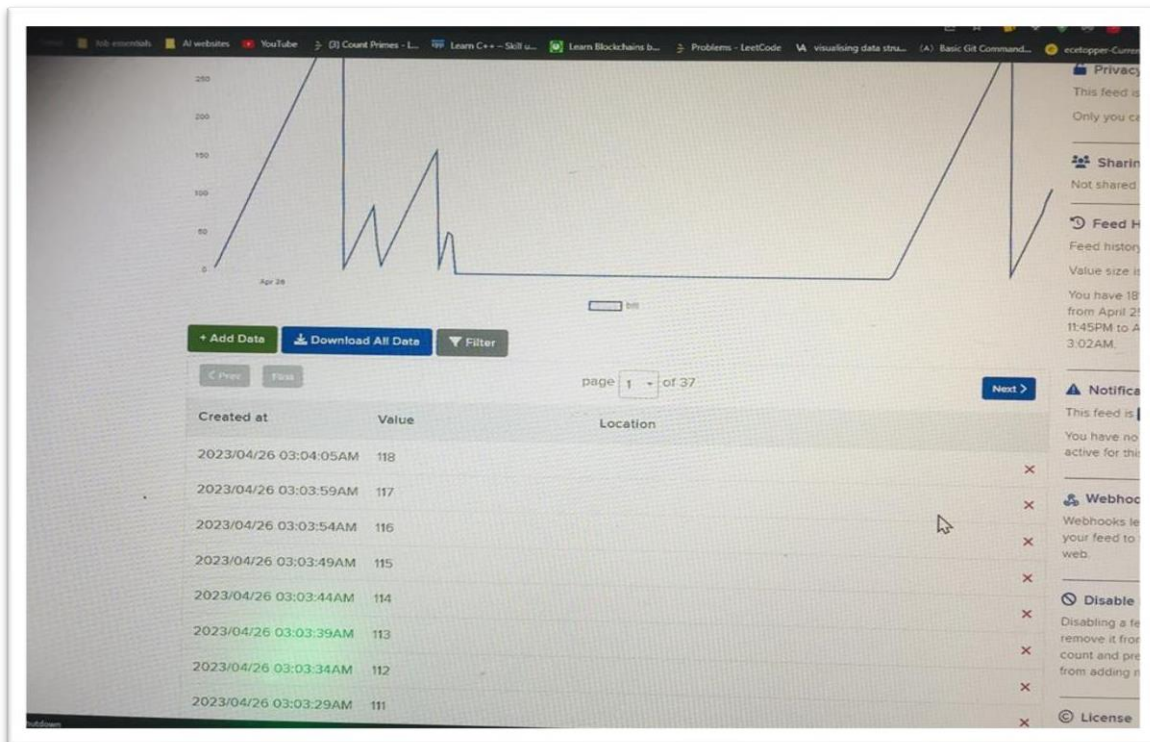


Fig 7 : OUTPUT OF METER

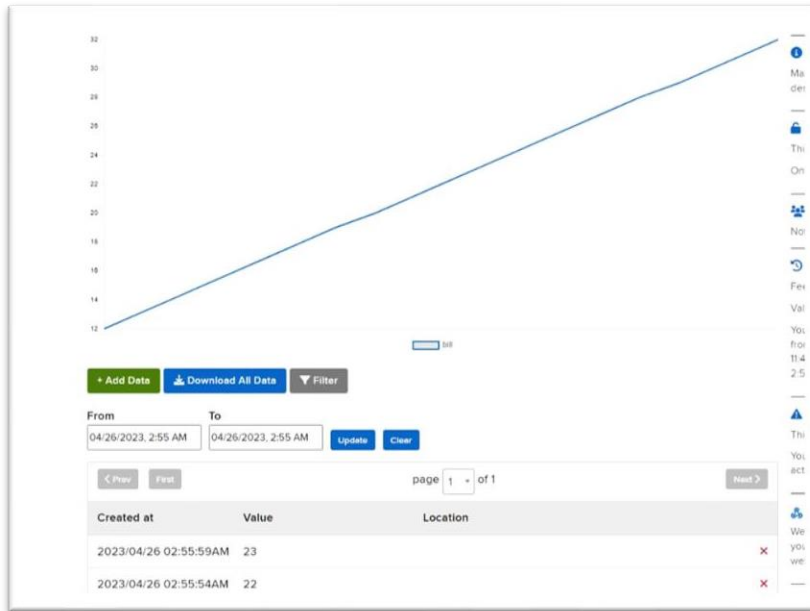


Fig 9: OUTPUT METER

```

// sketch_mcp3202.ino
1 #include "MCP3202.h"
2 #define MCP3202_ADDR 0x48
3 MCP3202 mcp3202(MCP3202_ADDR);
4 unsigned long last_time = 0;
5 unsigned long current_time = 0;
6 float V = 0;
7 void setup() {
8   Serial.begin(115200);
9   mcp3202.begin();
10 }
11 void loop() {
12   float V = 0;
13   float I = mcp3202.read();
14   // Serial.println(I);
15   float P = V * I;
16   last_time = current_time;
17   current_time = millis();
18   V = (I * 1000000) / 1000000.0;
19   digitalWrite(LED_BUILTIN, !V);
20   Serial.println(V);
21   delay(1000);
22 }

// mqtt_client.ino
1 #include <Arduino.h>
2 #include <ESP8266WiFi.h>
3 #include <ESP8266WiFiMulti.h>
4 #include <ESP8266WebServer.h>
5 #include <ESP8266HTTPClient.h>
6 #include <ESP8266HTTPUpdate.h>
7 #include <ESP8266HTTPMultipart.h>
8 #include <ESP8266HTTPBody.h>
9 #include <ESP8266HTTPBody.h>
10 #include <ESP8266HTTPBody.h>
11 #include <ESP8266HTTPBody.h>
12 #include <ESP8266HTTPBody.h>
13 #include <ESP8266HTTPBody.h>
14 #include <ESP8266HTTPBody.h>
15 #include <ESP8266HTTPBody.h>
16 #include <ESP8266HTTPBody.h>
17 #include <ESP8266HTTPBody.h>
18 #include <ESP8266HTTPBody.h>
19 #include <ESP8266HTTPBody.h>
20 #include <ESP8266HTTPBody.h>
21 #include <ESP8266HTTPBody.h>
22 #include <ESP8266HTTPBody.h>
23 #include <ESP8266HTTPBody.h>
24 #include <ESP8266HTTPBody.h>
25 #include <ESP8266HTTPBody.h>
26 #include <ESP8266HTTPBody.h>
27 #include <ESP8266HTTPBody.h>
28 #include <ESP8266HTTPBody.h>
29 #include <ESP8266HTTPBody.h>
30 #include <ESP8266HTTPBody.h>
31 #include <ESP8266HTTPBody.h>
32 #include <ESP8266HTTPBody.h>
33 #include <ESP8266HTTPBody.h>
34 #include <ESP8266HTTPBody.h>
35 #include <ESP8266HTTPBody.h>
36 #include <ESP8266HTTPBody.h>
37 #include <ESP8266HTTPBody.h>
38 #include <ESP8266HTTPBody.h>
39 #include <ESP8266HTTPBody.h>
40 #include <ESP8266HTTPBody.h>
41 #include <ESP8266HTTPBody.h>
42 #include <ESP8266HTTPBody.h>
43 #include <ESP8266HTTPBody.h>
44 #include <ESP8266HTTPBody.h>
45 #include <ESP8266HTTPBody.h>
46 #include <ESP8266HTTPBody.h>
47 #include <ESP8266HTTPBody.h>
48 #include <ESP8266HTTPBody.h>
49 #include <ESP8266HTTPBody.h>
50 #include <ESP8266HTTPBody.h>
51 #include <ESP8266HTTPBody.h>
52 #include <ESP8266HTTPBody.h>
53 #include <ESP8266HTTPBody.h>
54 #include <ESP8266HTTPBody.h>
55 #include <ESP8266HTTPBody.h>
56 #include <ESP8266HTTPBody.h>
57 #include <ESP8266HTTPBody.h>
58 #include <ESP8266HTTPBody.h>
59 #include <ESP8266HTTPBody.h>
60 #include <ESP8266HTTPBody.h>
61 #include <ESP8266HTTPBody.h>
62 #include <ESP8266HTTPBody.h>
63 #include <ESP8266HTTPBody.h>
64 #include <ESP8266HTTPBody.h>
65 #include <ESP8266HTTPBody.h>
66 #include <ESP8266HTTPBody.h>
67 #include <ESP8266HTTPBody.h>
68 #include <ESP8266HTTPBody.h>
69 #include <ESP8266HTTPBody.h>
70 #include <ESP8266HTTPBody.h>
71 #include <ESP8266HTTPBody.h>
72 #include <ESP8266HTTPBody.h>
73 #include <ESP8266HTTPBody.h>
74 #include <ESP8266HTTPBody.h>
75 #include <ESP8266HTTPBody.h>
76 #include <ESP8266HTTPBody.h>
77 #include <ESP8266HTTPBody.h>
78 #include <ESP8266HTTPBody.h>
79 #include <ESP8266HTTPBody.h>
80 #include <ESP8266HTTPBody.h>
81 #include <ESP8266HTTPBody.h>
82 #include <ESP8266HTTPBody.h>
83 #include <ESP8266HTTPBody.h>
84 #include <ESP8266HTTPBody.h>
85 #include <ESP8266HTTPBody.h>
86 #include <ESP8266HTTPBody.h>
87 #include <ESP8266HTTPBody.h>
88 #include <ESP8266HTTPBody.h>
89 #include <ESP8266HTTPBody.h>
90 #include <ESP8266HTTPBody.h>
91 #include <ESP8266HTTPBody.h>
92 #include <ESP8266HTTPBody.h>
93 #include <ESP8266HTTPBody.h>
94 #include <ESP8266HTTPBody.h>
95 #include <ESP8266HTTPBody.h>
96 #include <ESP8266HTTPBody.h>
97 #include <ESP8266HTTPBody.h>
98 #include <ESP8266HTTPBody.h>
99 #include <ESP8266HTTPBody.h>
100 #include <ESP8266HTTPBody.h>

```

Fig 8: CODING PART OF NODE MCU

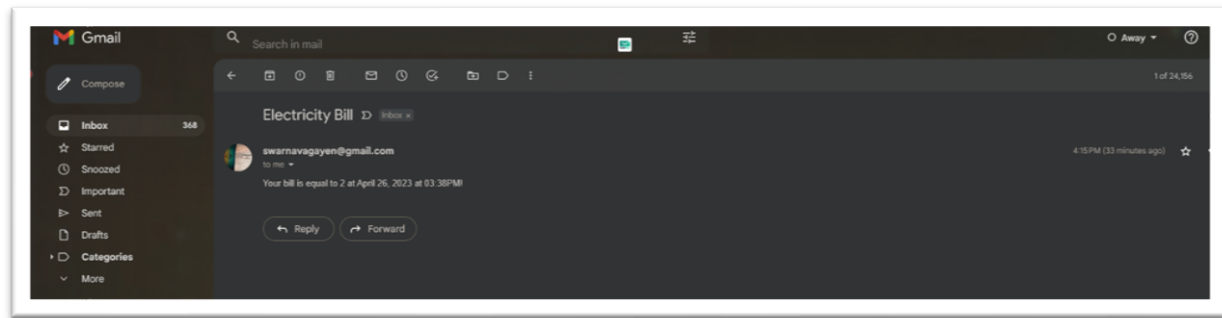


Fig 10: BILL SENT TO E-MAIL

DISCUSSION-

The project “Internet of Things based Smart Energy Meter “is an interesting project that involves the use of various electronic components such as the ACS712 current sensor, ESP8266 Wi-Fi module, and Arduino Uno. The project aims to measure the AC power consumption of a device using the ACS712 sensor, and then sends the measured data to a remote server through the ESP8266 Wi-Fi module. This project has many real-life applications in industries and homes, where it is necessary to monitor the power consumption of various devices for maintenance and energy management purposes. In this discussion, we have explain the various aspects of this project in detail, including its working principle, components used, circuit diagram, code, and applications.

Working principle: The working principle of the AC power measurement project is based on the use of the ACS712 current sensor, which measures the current flowing through the AC line. The ACS712 sensor is a Hall effect-based device that can measure the current flowing through a conductor without breaking the circuit. It can measure the current up to 5A, which makes it ideal for measuring the power consumption of small appliances. The ACS712 sensor outputs an analog voltage signal proportional to the current flowing through the conductor. The output voltage is then fed to the Arduino Uno, which converts the analog signal into a digital signal using its built-in Analog-to-Digital Converter (ADC). The ADC resolution of the Arduino Uno is 10 bits, which means that it can measure the voltage upto 1023 levels. The digital value obtained from the ADC is then converted into the corresponding current value using the formula: $\text{Current} = (\text{ADC value} / 1023) * 5V / \text{ACS712 sensitivity}$ Where ACS712 sensitivity is

0.185V/A.

The smart energy meter system project utilizing the ESP8266, Arduino Uno, LCD display 16x2, ACS712, ZMPT101B, and integration with Adafruit IO provides an effective solution for monitoring and managing energy consumption. In this discussion, we will explore the significance of the project, its benefits, limitations, and potential areas of improvement.

The smart energy meter system addresses the growing need for efficient energy management in residential and commercial settings. By leveraging the capabilities of the ESP8266 and Arduino Uno microcontrollers, the system enables real-time monitoring of energy consumption, allowing users to make informed decisions about their energy usage. The integration with Adafruit IO further enhances the functionality by providing remote access to energy data, enabling users to track and analyze their consumption patterns.

One of the key benefits of this project is its affordability and accessibility. The use of readily available components such as the ESP8266, Arduino Uno, and standard sensors makes it feasible for DIY enthusiasts and hobbyists to implement their own energy monitoring system. The LCD display provides a user-friendly interface for displaying energy data, allowing users to easily understand and track their consumption.

The integration with Adafruit IO adds a layer of convenience and flexibility to the system. By connecting to the Adafruit IO platform, users can access their energy data remotely through a web interface or mobile application. This allows for seamless monitoring, data visualization, and analysis, enabling users to identify trends, set energy-saving goals, and make informed decisions to optimize their energy usage.

However, there are certain limitations to consider. The accuracy of the energy measurements depends on the calibration of the ACS712 current sensor and ZMPT101B voltage sensor. Proper calibration procedures should be followed to ensure accurate readings. Additionally, noise and interference in the electrical system can affect the accuracy of the sensor readings. Implementing noise filtering techniques and proper grounding practices can help mitigate these issues.

Furthermore, the system's scalability and expandability should be considered. While the current design

focuses on monitoring energy consumption, future enhancements could include features such as load control, demand response, or integration with smart home automation systems. Planning for future expansion and modularity during the initial design phase will facilitate seamless integration of additional functionalities.

In terms of user experience, the project could benefit from additional features such as data logging, real-time notifications, and energy usage analytics. Incorporating these features would provide users with a more comprehensive understanding of their energy consumption patterns and empower them to make more informed decisions.

To enhance the security of the system, additional measures could be implemented, such as secure communication protocols (e.g., SSL/TLS) for data transmission between the microcontroller and Adafruit IO. User authentication and access control mechanisms can also be integrated into the system to ensure data privacy and prevent unauthorized access.

Overall, the smart energy meter system project demonstrates the potential for utilizing affordable components and IoT platforms to enable energy monitoring and management. While it provides a solid foundation for energy monitoring, there are opportunities for further development and customization to meet specific user requirements and industry standards.

By continually improving the accuracy, scalability, and user experience of the system, it can contribute to more efficient energy utilization, cost savings, and environmental sustain

8. CONCLUSION AND SCOPE OF FUTURE WORK

Conclusion

In conclusion, the smart energy meter system project utilizing the ESP8266, Arduino Uno, LCD display 16x2, ACS712, ZMPT101B, and integration with Adafruit IO offers an accessible and affordable solution for monitoring and managing energy consumption. By providing real-time data, users can make informed decisions about their energy usage, promoting energy efficiency and cost savings.

The project's integration with Adafruit IO expands the functionality by allowing users to remotely access and analyze their energy data. This empowers users to track consumption patterns, set energy-saving goals, and optimize their energy usage for a more sustainable future.

While the project showcases several benefits, it's important to consider areas of improvement. Ensuring accurate sensor calibration, addressing noise and interference issues, and planning for scalability and expandability are crucial for a reliable and adaptable system. Additionally, incorporating features such as data logging, real-time notifications, and energy usage analytics can enhance the user experience and provide deeper insights into energy consumption patterns.

By prioritizing security measures such as secure communication protocols and user authentication, the system can safeguard sensitive data and ensure privacy. This is essential in an increasingly connected world where data protection is of paramount importance.

The smart energy meter system project demonstrates the potential of affordable components and IoT platforms in enabling energy monitoring and management. With continuous refinement and customization to meet user needs and industry standards, this project has the potential to contribute to more efficient energy utilization, cost savings, and a greener environment.

By embracing the principles of sustainability and empowering users with knowledge and control over their energy consumption, the smart energy meter system project paves the way for a more conscious and responsible approach to energy management. Together, we can make a positive impact on our energy usage and contribute to a more sustainable future for generations to come.

Scope of future work

- 1) The smart energy meter system project holds great potential for future expansion and development. Here are some areas of future scope that can be explored:
- 2) **Advanced Analytics and Visualization:** Enhance the project by implementing advanced data analytics and visualization techniques. This could involve generating meaningful insights, identifying consumption patterns, and providing visual representations of energy data through graphs, charts, and dashboards. This will enable users to gain deeper insights into their energy consumption habits and make more informed decisions.
- 3) **Machine Learning and Predictive Analytics:** Incorporate machine learning algorithms to analyze historical energy data and predict future consumption patterns. By leveraging predictive analytics, the system can provide users with personalized recommendations on optimizing energy usage, reducing peak demand, and achieving energy savings.
- 4) **Demand Response Integration:** Integrate the smart energy meter system with demand response programs. This allows the system to respond to signals from utility providers during peak demand periods, automatically adjusting energy usage or engaging in load shedding strategies to alleviate strain on the grid and potentially benefit from incentive programs.
- 5) **Smart Home Integration:** Expand the project to integrate with other smart home devices and systems. This would enable the system to interact with appliances, lighting, heating, and cooling systems, allowing for automated energy management based on user preferences and real-time energy data. Additionally, integrating with voice assistants like Amazon Alexa or Google Assistant can provide hands-free control and energy optimization.
- 6) **Energy Monitoring for Renewable Sources:** Extend the project to monitor and analyze energy generation from renewable sources such as solar panels or wind turbines. This would provide users with a comprehensive view of their energy ecosystem, including both consumption and generation, and facilitate better management of renewable energy resources.

- 7) **Mobile Application Development:** Develop a dedicated mobile application to provide users with remote access to real-time energy data, notifications, and control over connected devices. The mobile app can also offer features like energy-saving tips, personalized recommendations, and energy usage goal tracking.
- 8) **Integration with Smart Grid Infrastructure:** Explore the integration of the smart energy meter system with the existing smart grid infrastructure. This would enable bidirectional communication between the energy meter and utility providers, facilitating real-time energy usage monitoring, load balancing, and improved grid management.
- 9) **Energy Efficiency Benchmarking:** Introduce benchmarking capabilities that allow users to compare their energy consumption against similar households or industry standards. This can provide insights into energy performance and encourage healthy competition for energy efficiency improvements.
- 10) **Blockchain Integration:** Investigate the integration of blockchain technology to enhance the security, transparency, and reliability of energy data. Blockchain can ensure the immutability and integrity of energy transactions, enable peer-to-peer energy trading, and facilitate incentive programs for energy conservation.
- 11) **Community Engagement and Social Impact:** Foster community engagement by creating a platform for users to share energy-saving tips, participate in energy challenges, and collaborate on sustainability initiatives. This can create a sense of community and promote collective efforts towards a greener future.
- 12) **Integration with Renewable Energy Certificates (RECs):** Explore the possibility of integrating the smart energy meter system with the issuance and tracking of Renewable Energy Certificates. This would allow users to quantify and validate the renewable energy they consume, contributing to sustainability goals and supporting the development of renewable energy projects.
- 13) **Real-time Pricing Integration:** Integrate the project with real-time pricing information from

utility providers. This would enable users to monitor energy prices and adjust their consumption patterns accordingly, maximizing cost savings by shifting usage to off-peak hours or when renewable energy generation is high.

- 14) **Energy Consumption Comparison:** Develop a feature that allows users to compare their energy consumption with similar households or buildings in their area. This benchmarking functionality can provide insights into potential energy-saving opportunities and encourage healthy competition for energy efficiency improvements.
- 15) **Energy Management for Multiple Locations:** Extend the project's capabilities to support energy management for multiple locations, such as homes, offices, or commercial buildings. This would involve integrating multiple energy meters and providing centralized monitoring and control through a single interface.
- 16) **Integration with Smart Appliances:** Collaborate with appliance manufacturers to integrate the project with smart appliances that can communicate and respond to energy consumption data. This enables automated energy optimization based on real-time information and user preferences.
- 17) **Gamification and Rewards System:** Implement a gamification element to encourage users to actively participate in energy conservation. By earning points or rewards based on energy-saving achievements, users can be motivated to adopt sustainable practices and compete for recognition.
- 18) **Energy Audit and Recommendations:** Develop an energy audit feature that analyzes energy consumption patterns and provides personalized recommendations for energy-saving improvements. This could include suggestions for appliance upgrades, insulation enhancements, or behavior modifications.
- 19) **Integration with Electric Vehicle (EV) Charging:** Explore the integration of the smart energy meter system with EV charging infrastructure. This would enable users to monitor and manage the charging of their electric vehicles, optimizing energy usage and potentially leveraging renewable energy sources for charging.

- 20) Energy Data Sharing and Research: Consider the anonymized and aggregated sharing of energy data for research purposes. With appropriate consent and privacy measures, the collected data can contribute to energy research, grid planning, and policy-making initiatives.
- 21) Continuous System Optimization: Implement a feedback loop to continuously optimize the performance of the smart energy meter system. Regular monitoring, maintenance, and firmware updates can ensure the system remains accurate, reliable, and compatible with evolving technologies and industry standards.
- 22) By exploring these future scope areas, the smart energy meter system project can continue to evolve, providing even more sophisticated energy monitoring, management, and sustainability features. The integration of emerging technologies, collaboration with industry partners, and engagement with users will be essential to drive innovation and maximize the project's impact on energy efficiency and environmental stewardship.

REFERENCE

H. T. Nguyen, S. S. Sreejith, and K. S. Kim, “Smart home energy management system with Arduino and labview: A review,” *Journal of Energy Storage*, vol. 31, p. 101869, Mar. 2020, doi: 10.1016/j.est.2020.101869.

N. Wang, Y. Chen, X. Chen, and X. Yan, “A Smart Home Energy Management System Based on Wireless Sensor Network and Artificial Neural Network,” in *2019 IEEE International Conference on Computational Electromagnetics (ICCEM)*, 2019, pp. 107–110, doi: 10.1109/ICCEM47739.2019.8955629.

N. Arunkumar and A. Chandrasekar, “Smart Energy Management System for Home Automation,” *International Journal of Power Electronics and Drive System*, vol. 9, no. 2, pp. 808–819, Jun. 2018, doi: 10.11591/ijpeds.v9.i2.pp808-819.

S. Chaurasiya, P. Raj, and R. Sahu, “IoT Based Smart Home Energy Management System: An Overview,” in *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*, 2020, pp. 273–278, doi: 10.1109/SPIN48953.2020.9071688.

“ESP8266 Datasheet,” AI-Thinker, 2016. “ACS712 Datasheet,” Allegro MicroSystems, 2017.

“Arduino Uno Rev3 Datasheet,” Arduino, 2012.

“LLC Datasheet,” Sparkfun Electronics, 2016.

Gupta, R., & Mukhopadhyay, S. (2015). Non-Invasive Sensing Techniques for Monitoring Electrical Energy Consumption—A Review. *IEEE Sensors Journal*, 15(5), 2745-2755. doi: 10.1109/jsen.2014.238235

Chinniah, K., Kanagaraj, G., Kaliappan, M., & Kandasamy, N. (2016). Power consumption monitoring using Arduino. 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), 1-5. doi: 10.1109/iccpct.2016.7530273

Cheng, M., Liu, Y., Li, Y., Li, Q., & Guo, S. (2018). Design of Home Energy Management System Based on IoT. 2018 IEEE 4th International Conference on Computer and Communications (ICCC), 1022-1026. doi: 10.1109/compcomm.2018.8667237

ACS712 datasheet. Allegro MicroSystems. <https://www.allegromicro.com/-/media/files/datasheets/acs712-datasheet.ashx>

ESP8266 datasheet. Espressif Systems. https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

Arduino Uno website. <https://www.arduino.cc/en/Main/arduinoBoardUno>

Logic Level Converter website. Adafruit Industries. <https://learn.adafruit.com/adafruit-txb0108-logic-level-converter/overview>

Open Energy Monitor project. <https://openenergymonitor.org/>