Smart Energy monitoring system using ESP32 and Blynk

ABSTRACT:

The Smart Energy Monitoring System is designed to optimize energy usage in residential and industrial environments by providing real-time monitoring and management capabilities. Utilizing the ESP32 microcontroller for efficient data collection and processing, the system tracks energy consumption via sensors and transmits the data to a cloud-based platform, Blynk, for visualization and control. The system enables users to monitor energy usage trends, detect inefficiencies, and make informed decisions to reduce consumption. With features such as remote monitoring, real-time alerts, and energy usage analytics, the Smart Energy Monitoring System aims to promote sustainability and energy efficiency. The integration of ESP32 and Blynk enhances system flexibility, scalability, and ease of use, making it an ideal solution for smart homes and businesses looking to optimize energy management.

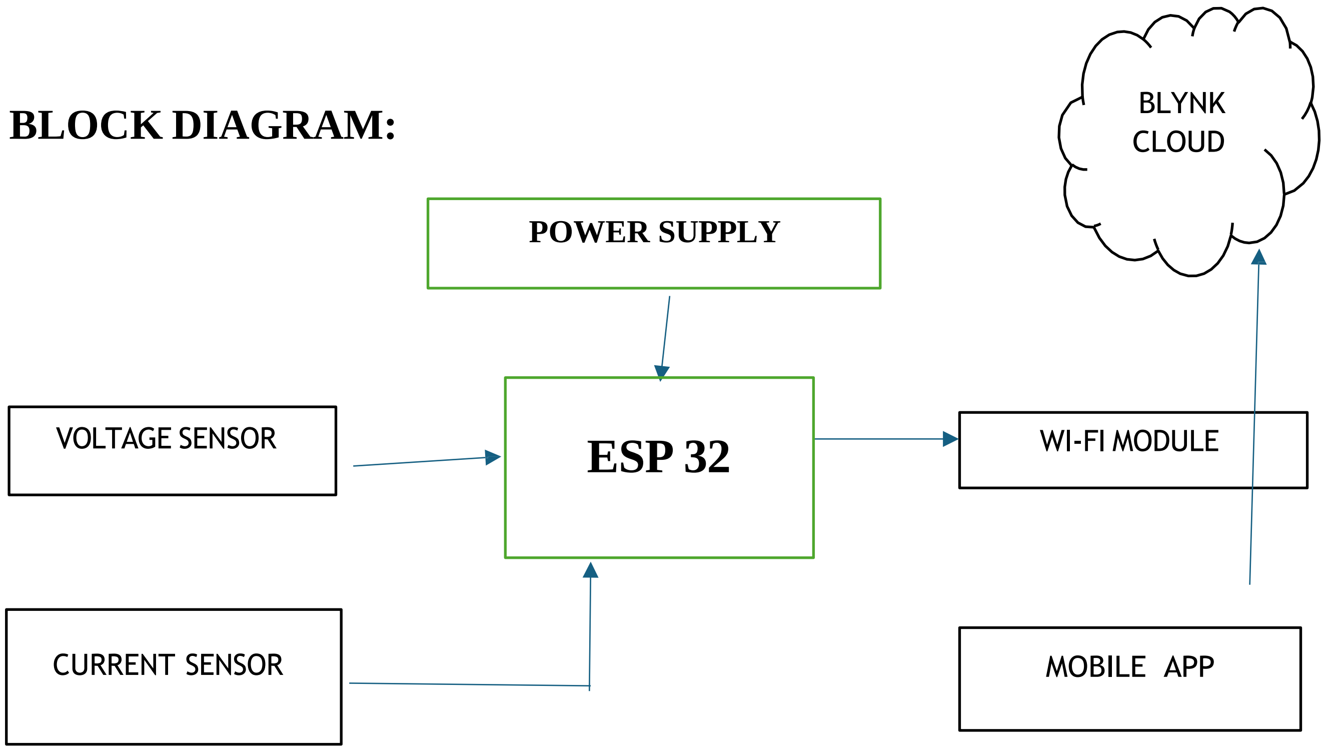
INTRODUCTION:

In this project, we will use the recently updated Blynk 2.0 application and an ESP32 to create an Internet of Things-based Smart Electricity Energy Meter. utilising the old Blynk application, we had previously constructed an IoT DC energy meter, a GSM prepaid energy meter, and an IoT energy meter utilising ESP32. However, with the recent Blynk platform upgrade, we also needed to modernise our energy meter project. Our objective is to streamline and automate the process of monitoring electricity consumption**.**

Manual meter readings have historically been used to track energy usage, which may be cumbersome and time-consuming. By automating distant data collecting, the Internet of Things offers a solution that saves both money and time. Building our own Internet of Things (IoT)-based electricity energy meter is a great opportunity because the idea of a smart energy meter has been quite popular all over the world in recent years.

We will use the revised Blynk 2.0 program and an ESP32 to develop an Internet of Things-based Smart Electricity Energy Meter. We can monitor voltage, current, power, and total energy consumption in kWh by utilising the best current sensor (SCT-013) and voltage sensor (ZMPT101B). The Blynk 2.0 app will receive the readings and show them on a dashboard that can be seen from anywhere. The ESP32's EEPROM will store the energy meter data in the event of a power failure, guaranteeing ongoing readings. Let's get started on our project to automate the monitoring of electricity consumption.

BLOCK DIAGRAM:



The Smart Energy Monitoring System consists of the following components working together:

1. **Energy Meter (Sensors): ZMPT101B voltage sensor and SCT-013-030** current sensor measures key electrical parameters like current, voltage, and power consumption. And track real time power usage.
2. **ESP32 Microcontroller:** This serves as the central controller of the system. It reads data from the energy meter sensors through its ADC (Analog-to-Digital Converter), processes the information (calculates energy usage), and sends it to the Blynk Cloud over Wi-Fi. The ESP32 is chosen for its built-in Wi-Fi, making it ideal for IoT applications.
3. **Wi-Fi Communication:** The ESP32 uses Wi-Fi to transmit the data to the Blynk Cloud platform. This allows real-time data sharing and remote access.
4. **Blynk Cloud:** The cloud platform receives, stores, and processes the energy consumption data sent by the ESP32. It enables easy visualization and monitoring of energy usage trends over time. The Blynk Cloud also handles user commands, such as controlling connected devices remotely.
5. **Blynk App (Mobile/Web Interface**): Users interact with the system through the Blynk app on their mobile device or a web interface. The app displays the real-time energy data in graphical formats like charts and allows users to set alerts, monitor energy usage patterns, and control devices remotely (turn on/off appliances, etc.).

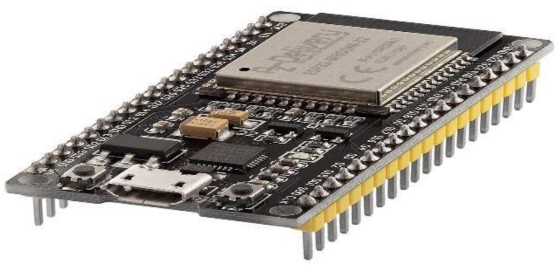
COMPONENTS USED:

HARDWARE REQUIREMENTS:

- ESP32 Microcontroller
- **ZMPT101B** Voltage Sensor – 240 volts
- **SCT-013-030** Non-invasive Current Sensor – 100Amp
- Power Supply -5v
- Resistor 10K and 100ohm
- Connecting Wires
- Wi-Fi Connectivity
- Capacitor 10uF
- Breadboard
- Load (e.g., light bulb)
- Smartphone

1. ESP32 Microcontroller

if Systems created the ESP32 microcontroller, a potent and adaptable chip that is frequently utilised in embedded systems, smart gadgets, and Internet of Things (IoT) applications. an extensive feature set, which includes numerous input/output, strong processing power, and Bluetooth and Wi-Fi connectivity.



The ESP32 is built around the Xtensa LX6 dual-core 32-bit processor:

- Dual-Core Architecture: Allows simultaneous execution of multiple tasks, such as handling network communications and controlling peripherals.
- Clock Speed: Operates at up to 240 MHz, making it capable of running complex applications.

2. Memory

- SRAM: 520 KB on-chip memory ensures fast data processing and temporary data storage.
- Flash Storage: Supports external flash memory (up to 16 MB), used for storing the program code and persistent data.

Wireless Connectivity

The ESP32 integrates both **Wi-Fi** and **Bluetooth**, enabling it to connect to a wide range of networks and devices:

- **Wi-Fi**:
    - Compatible with 802.11 b/g/n (2.4 GHz), making it ideal for IoT applications requiring internet access.
    - Supports features like Access Point (AP) and Station (STA) modes for flexible network setups.
- **Bluetooth**:

- Supports **Bluetooth 4.2** and **Bluetooth Low Energy (BLE)** for efficient short-range communication.
- Useful for wearable devices, sensor data transmission, and device-to-device communication.

Versatile Input/Output (I/O)

The ESP32 offers multiple peripheral interfaces to interact with sensors, displays, motors, and other components:

- **GPIO Pins**: Programmable pins for input/output operations, suitable for reading sensors or driving LEDs.
- **Analog I/O**:
    - **ADC (Analog-to-Digital Converter)**: Converts analog signals into digital data (up to 18 channels).
    - **DAC (Digital-to-Analog Converter)**: Outputs analog signals from digital data (2 channels).
- **PWM (Pulse Width Modulation)**: Generates precise signals for applications like motor speed control and LED dimming.
- **Communication Protocols**:
    - **UART**: Serial communication for debugging or interfacing with other microcontrollers.
    - **SPI and I2C**: For communication with displays, memory chips, and sensors.
    - **I2S**: For audio data streaming.
- **SD Card Support**: Reads and writes data to an SD card for data logging.

Built-in Sensors

The ESP32 includes integrated sensors that extend its functionality:

- **Hall Effect Sensor**: Detects magnetic fields, useful in motor control or security systems.

- **Capacitive Touch Sensors**: Enables touch-sensitive controls, ideal for creating intuitive user interfaces.

Power Management

The ESP32 is designed for both performance and energy efficiency:

- **Low-Power Modes**: Includes deep sleep and light sleep modes, reducing power consumption for battery-operated devices.
- **Wake-Up Features**: Can wake up based on timers, GPIO input, or network activity.

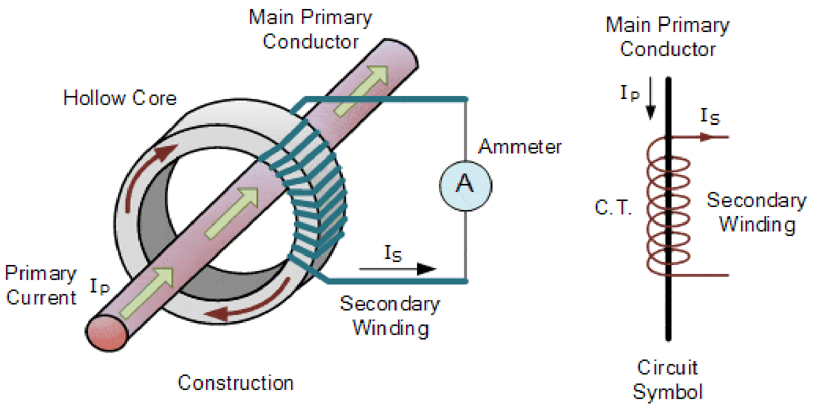
Software and Development

The ESP32 supports a wide range of development tools:

- **Arduino IDE**: A beginner-friendly platform with extensive libraries.
- **ESP-IDF**: Espressif's official development framework, providing fine-grained control.
- **MicroPython**: A lightweight Python interpreter for quick prototyping.
- **PlatformIO**: Advanced integrated development for multi-platform projects.

2. SCT-013 Current Sensor

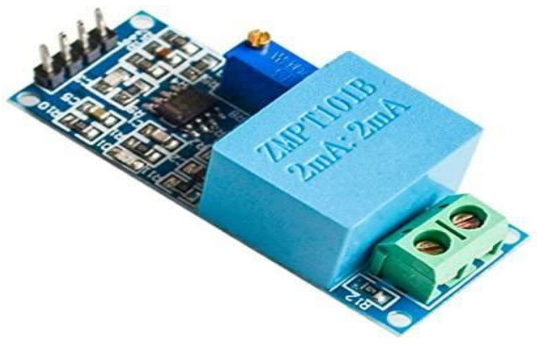
The SCT-013 is a non-invasive clamp meter sensor with a split core that can detect AC current up to 100 amps. This kind of current sensor is used to measure alternating current in a building and is often referred to as a current transformer (CT). The SCT-013 is easy to use because it doesn't require any high-voltage electrical work to attach to either the live or neutral wire.



The sensor has a primary winding, magnetic core, and secondary winding, which is composed of many turns of fine wire enclosed in the transformer casing.

3. ZMPT101B Voltage Sensor

The **ZMPT101B** is a specialized sensor module for measuring AC voltage. It is widely used in applications like energy monitoring, IoT systems, and automation projects due to its high precision and ease of integration.



High-Precision Voltage Transformer:

- Converts high AC voltage into a safe, proportional low-voltage signal.

Adjustable Gain:

- A built-in potentiometer allows you to fine-tune the sensitivity for accurate measurement.

Analog Output:

- Provides a voltage output proportional to the measured AC voltage, compatible with ADC pins of microcontrollers.

Compact Design:

- Lightweight and easy to integrate into embedded systems.

Operating Voltage:

- Powered by a standard **5V DC**, making it compatible with Arduino, ESP32, and other microcontrollers.
- **Input Voltage Range**: 0–250V AC (higher ranges require external circuitry).
- **Output Signal**: Analog voltage proportional to the input AC voltage.
- **Operating Voltage**: 5V DC.
- **Accuracy**: High accuracy due to the precision transformer.
- **Energy Monitoring**: Measure and monitor household or industrial power usage.
- **Home Automation**: Voltage monitoring for appliances and circuits.
- **IoT Systems**: Remote voltage tracking for smart devices
- **Safety Systems**: Detect voltage fluctuations to protect sensitive equipment.
- **Connect the Sensor**: VCC to 5V, GND to ground, and OUT to an ADC pin on the microcontroller.
- **Calibrate the Module**: Adjust the potentiometer to ensure accurate voltage readings.
- **Measure Voltage**: Read the analog signal, process it using a formula to determine the actual AC voltage.

5. Resistors (10K and 100 Ohm)

- **Description:** Passive components used in the circuit for voltage scaling, current limiting, and pull-up/pull-down functions.
- **Role in the System:**
  - **10K Resistor:** used in voltage divider circuits or as pull-up/pull-down resistors for stabilizing signals.
  - **100 Ohm Resistor:** used for current limiting in sensor circuits or to protect the microcontroller.

6. Connecting Wires

- **Description:** Wires used to connect components and establish a complete circuit on the breadboard.
- **Role in the System:**
  - Provide pathways for electrical signals between the ESP32, sensors, and other components**.**

7. Wi-Fi Connectivity

- **Description:** Provided by the ESP32’s built-in Wi-Fi module for connecting to the internet.
- **Role in the System:**
  - Enables communication between the ESP32 and the Blynk app for real-time data monitoring and control.

8. Capacitor (10uF)

- **Description:** A passive component that stabilizes the circuit by filtering out noise and voltage fluctuations.
- **Role in the System:**
    - Smooths out voltage signals, especially from the sensors, to ensure accurate readings.
    - Prevents sudden spikes or drops in power supply from affecting the ESP32 and sensors.

9. Breadboard

- **Description**: A solderless prototyping board for temporary assembly of electronic circuits.
- **Role in the System:**
    - Facilitates easy and flexible connections between components.
    - Allows modifications and troubleshooting during development.

10. Load (e.g., Light Bulb)

- **Description:** The electrical device whose power consumption is monitored.
- **Role in the System:**
    - Demonstrates energy monitoring by acting as a measurable load.
    - Makes it possible to test voltage, current, and power calculations.

11. Smartphone

- **Description:** Acts as a user interface for real-time monitoring and control via the Blynk app.
- **Role in the System:**
    - Displays live data (voltage, current, power, and energy consumption) sent by the ESP32.
    - Allows users to set alerts, control devices, and visualize trends.

SOFTWARE REQUIREMENTS:

- Arduino IDE
- Blynk App
- Blynk Library
- Energy Monitoring Code
- C++ Programming language
- Emon Library

1. Arduino IDE

• **Purpose:**

The Arduino Integrated Development Environment (IDE) is used to write, compile, and upload code to the ESP32 board.

• **Features:**

Easy to use for beginners with support for multiple libraries.

Compatible with ESP32 by installing the ESP32 board manager.

2. . Blynk App

• **Purpose:**

The Blynk app allows users to monitor energy consumption and control appliances remotely.

• **Features:**

Intuitive dashboard creation for monitoring voltage, current, and power.

Supports widgets like graphs, labels, and gauges for real-time data visualization.

• **Setup:**

Create an account on Blynk 2.0. Generate an authentication token for your project to link the ESP32 to the app. Configure widgets like gauges and graphs for real-time monitoring of current, voltage, and power.

3. Blynk Library

• **Purpose:** The Blynk Library enables communication between the ESP32 and the Blynk app via Wi-Fi.

• **Installation:** Open the Arduino IDE. Go to Sketch > Include Library > Manage Libraries, search for "Blynk," and install the latest version.

• **Usage:** Use the **Blynk.begin()** function to connect the ESP32 to the Blynk cloud using the auth token, Wi-Fi SSID, and password.

4. Energy Monitoring Code

• **Purpose:** The code integrates the sensors with the ESP32, performs calculations for power and energy monitoring, and transmits the data to the Blynk app.

• **Features:**

Reads voltage from the ZMPT101B sensor and current from the SCT-013-030 sensor.

Calculates active power, apparent power, and energy consumption. o Sends real-time data to the Blynk app for display

5. C++ Programming Language

• **Purpose:** Arduino code is written in C++, with support for object-oriented programming and library integration.

• **Features:**

Simple syntax for reading analog inputs, controlling GPIOs, and interfacing with sensors.

Allows integration of libraries like Blynk and Emon for efficient energy monitoring

6. Emon Library

• **Purpose:** The Emon Library is used for energy monitoring and power calculation. It simplifies interfacing with sensors like SCT-013-030 and ZMPT101B.

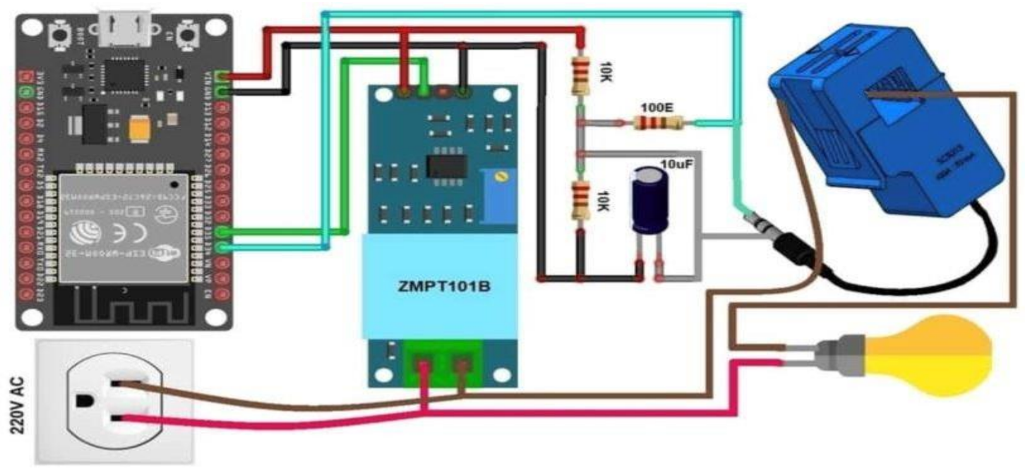
• **Installation:** In Arduino IDE, go to Library Manager, search for "EmonLib," and install it.

• **Features:**

Provides functions to measure AC voltage and current. o Includes calibration tools to ensure accurate readings. o Can calculate real power, apparent power, and power factor.

Circuit Diagram & Hardware Setup

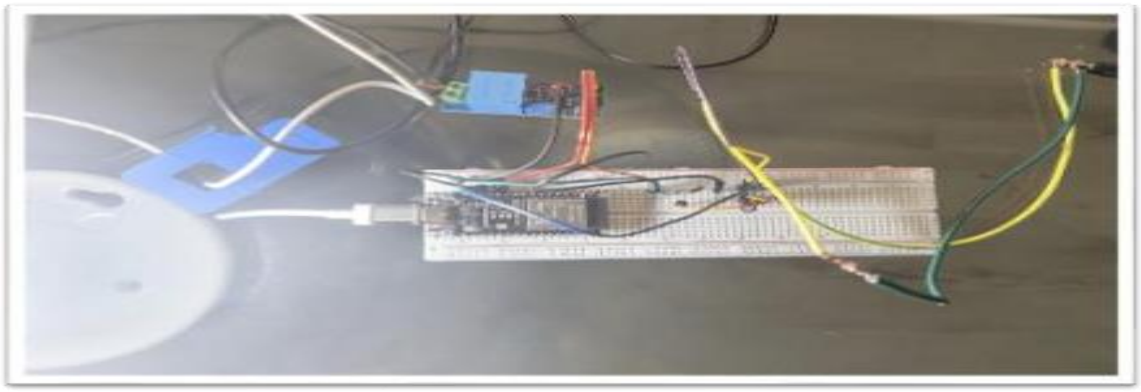
Below is the circuit diagram for the ESP32-based Internet of Things electricity energy meter. The Fritzing software was used to construct the design, and the connection diagram is straightforward.



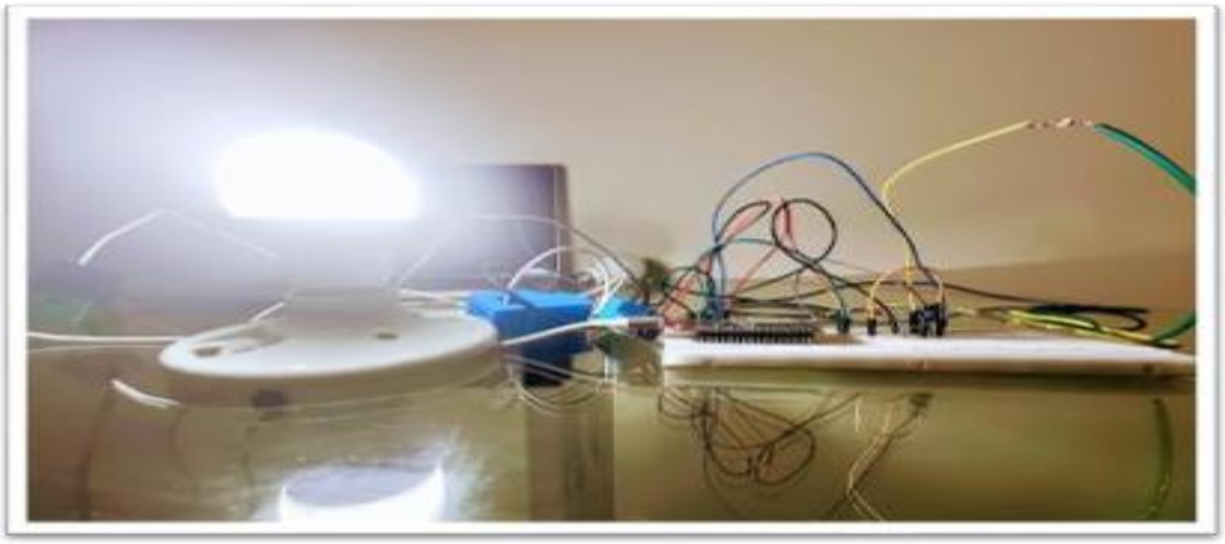
The pin connections for the smart energy monitoring system are carefully designed to ensure proper functionality and accurate data acquisition. The ZMPT101B voltage sensor is connected to the 5V power supply via its **VCC** pin, with the **GND** pin connected to the system ground. Its **Analog Output** pin is linked to **GPIO3** on the ESP32, enabling it to send voltage readings.

The SCT-013 current sensor's **VCC** and **GND** pins are connected to the 5V supply and ground, respectively, while its **Analog Output** pin is connected to **GPIO4** on the ESP32 for current measurement.

10k and 100-ohm resistors are placed in series along the sensor input lines to condition signals and limit current, protecting the ESP32 from voltage spikes. A 10uF capacitor is connected in parallel to the resistors to filter noise and stabilize the input signals, ensuring reliable data transmission and system accuracy. These connections integrate seamlessly to facilitate precise monitoring of energy parameters.



The AC wires need to be connected to the Voltage Sensor's input AC Terminal to measure the voltage and current. Only one live or neutral wire needs to be placed within the clip portion for the Current Sensor.

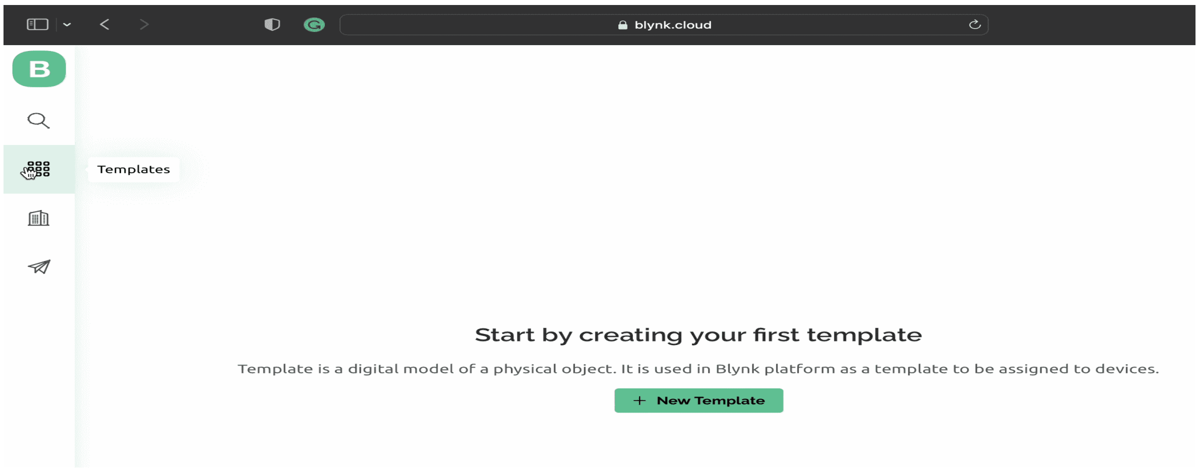


Setting Up Blynk 2.0 Web and Mobile Dashboard

Blynk is a smartphone application that can be used on both Android and iOS smartphones to operate any Internet of Things application. It lets you design your own graphical user interface for Internet of Things applications. Here, the Blynk Web Dashboard and mobile application will both show the IoT energy meter data.

Visit **blynk.io** and sign up using the email ID.

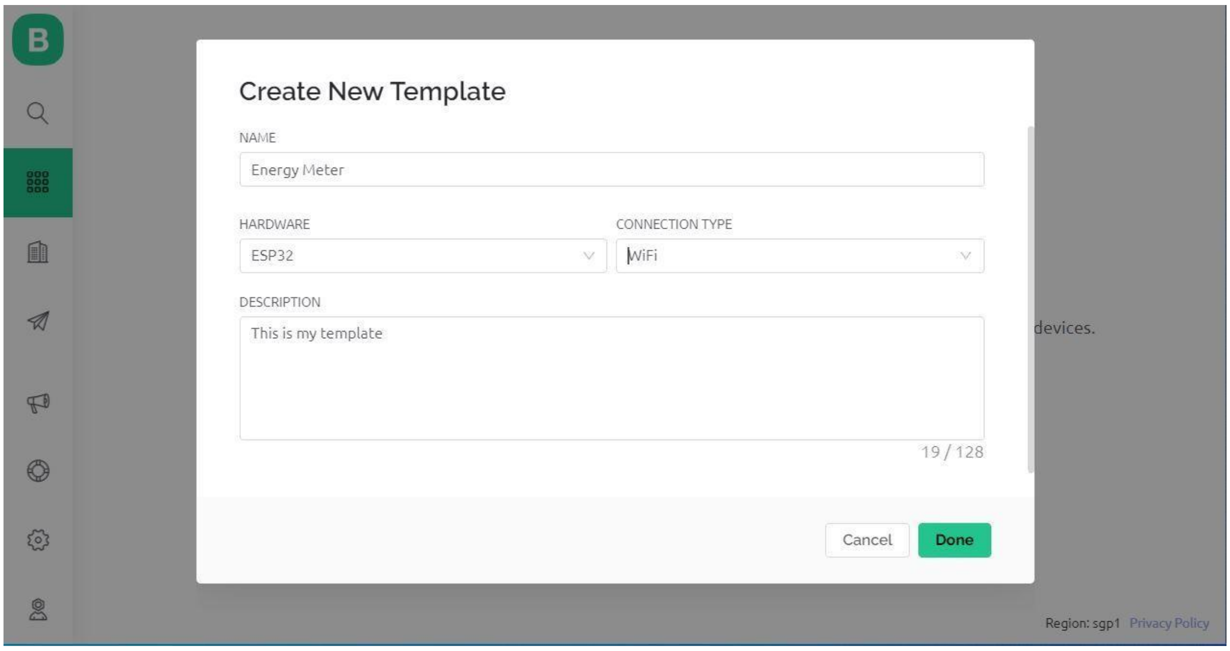
First, we created a New Template.



Start by creating your first template

Template is a digital model of a physical object. It is used in Blynk platform as a template to be assigned to devices.

Assigned the name, Hardware & Connection Type.



Create New Template

NAME: Energy Meter

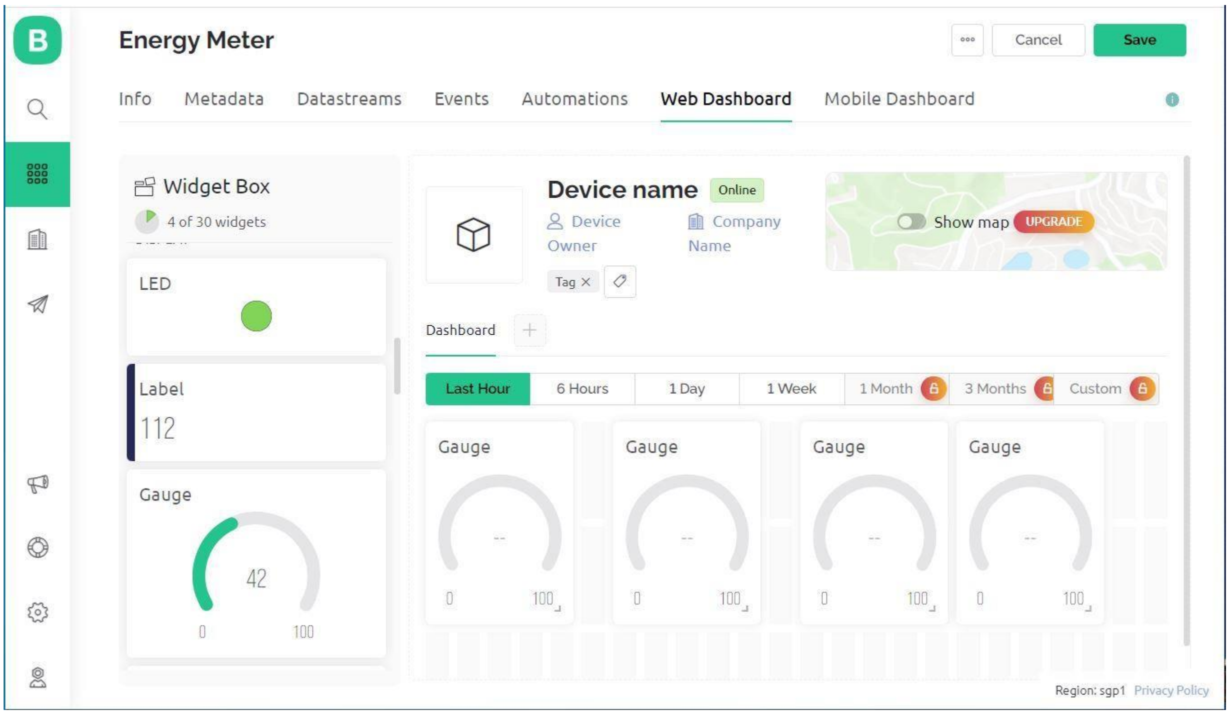
HARDWARE: ESP32

CONNECTION TYPE: WiFi

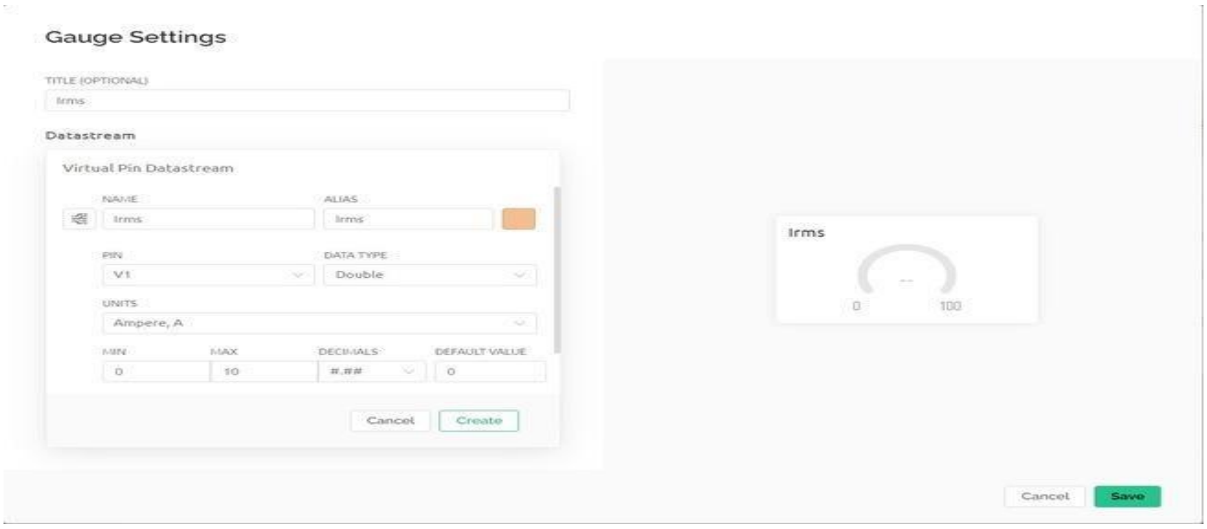
DESCRIPTION: This is my template

19 / 128

From the Web Dashboard, Created a 4 widgets Gauge.



The purpose of the four widgets is to show the Vrms, Irms, Power, and KWh values. Adjust the settings according to the picture below.


Gauge Settings

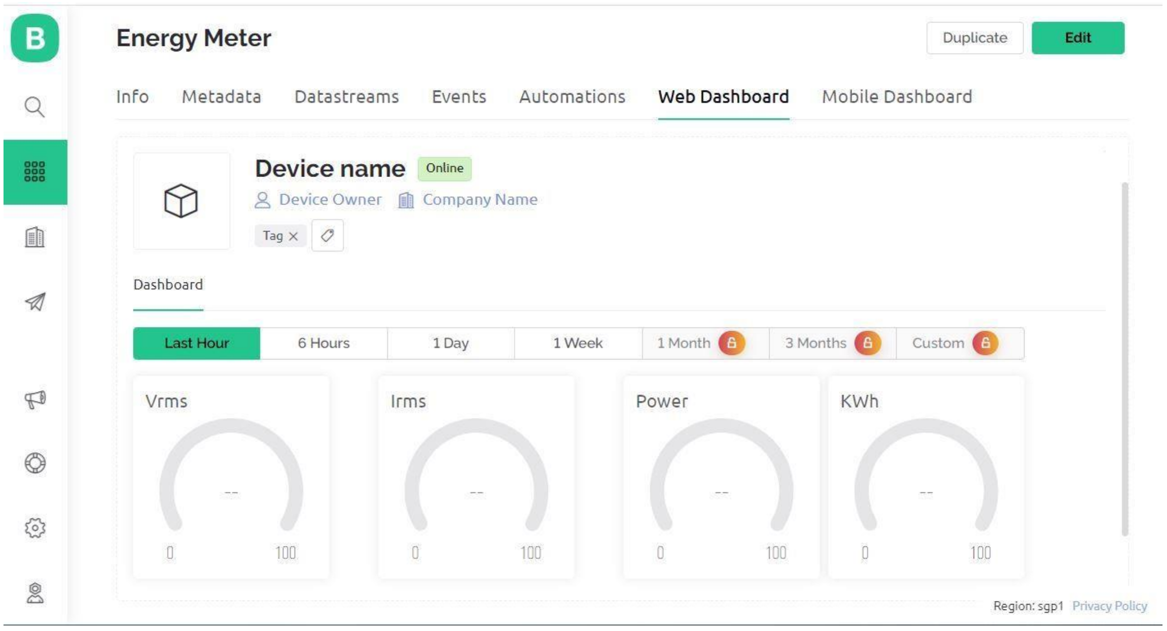
TITLE (OPTIONAL): Irms

Datastream

Virtual Pin Datastream

NAME: Irms
ALIAS: Irms
PIN: V1
DATA TYPE: Double
UNITS: Ampere, A
MIN: 0
MAX: 10
DECIMALS: #.##
DEFAULT VALUE: 0

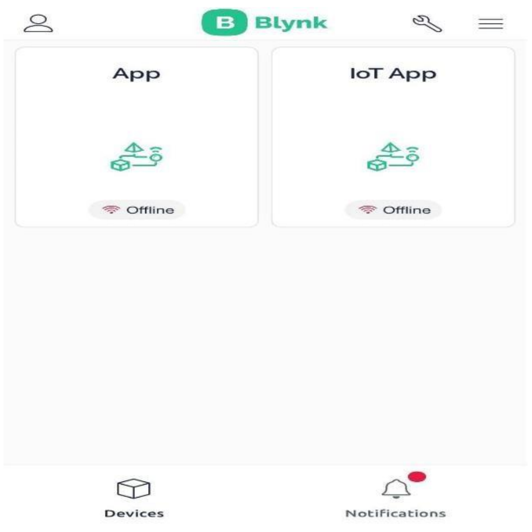
At last, the Web Dashboard appears as follows and is prepared to accept the data from the ESP32 Smart Energy Meter.

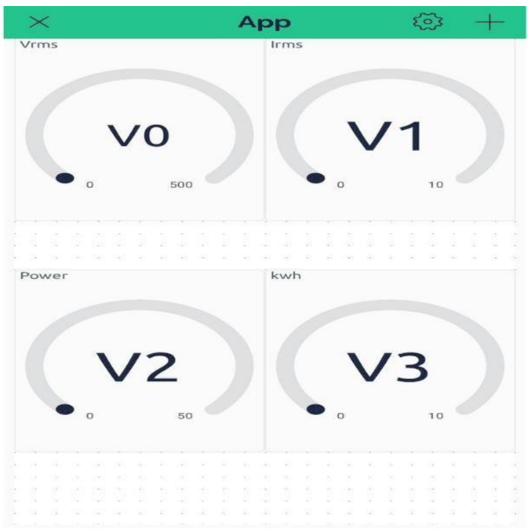


We configured in the mobile app dashboard in addition to the web dashboard.

The Blynk app is available for download and installation from the Google Play Store. The App Store offers downloads for iOS users.

After the installation is finished, launch the application and register with your email address and password. Next, configure the application similarly to Web Dashboard.





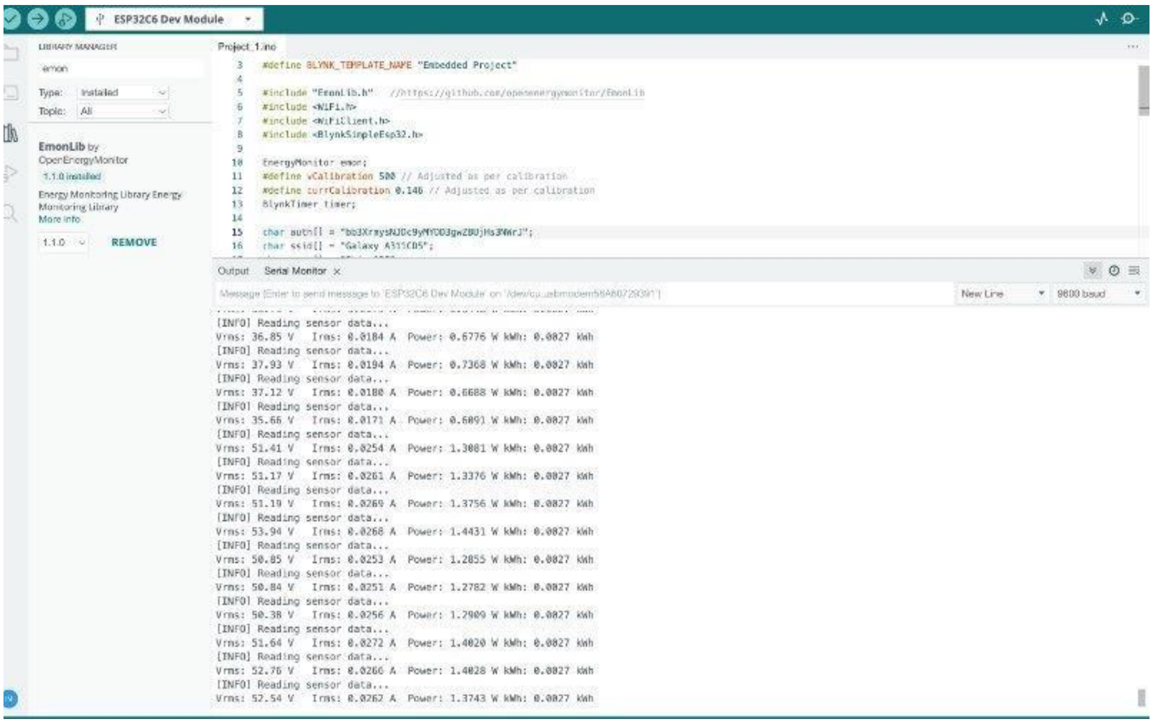
Required Library Installation:

Emon Lib Library Electricity energy meters use the Emon lib Library. Emon Lib, or Continuous Monitoring of Electricity Energy, repeats a series of voltage and current measurements every five to ten seconds.

The sketch is notified that the measurements are available and should be read and processed by EemonLib, which continually monitors the voltage and all of the current input channels the background and computes a true average quantity for each. Download **EmonLib Library**

The **Emon (Energy Monitor)** library is an essential tool for measuring and analyzing electrical parameters such as voltage, current, power, and energy. It is widely used in energy monitoring systems due to its simplicity and accuracy. Developed as part of the OpenEnergyMonitor project, the library provides functions to process analog sensor data and calculate real-time energy metrics.

It supports sensors like the ZMPT101B voltage sensor and the SCT-013 current sensor, enabling the calculation of root mean square (RMS) values, apparent power, real power, and power factor. By handling complex signal processing tasks like offset removal and phase error correction, the Emon library simplifies energy monitoring in projects like smart energy systems.


ESP32C6 Dev Module
LIBRARY MANAGER
emon
Type: Installed
Topic: All
EmonLib by OpenEnergyMonitor
1.1.0 installed
Energy Monitoring Library Energy Monitoring Library
More info
1.1.0
REMOVE
Project_1.ino
Output
Serial Monitor
Message (Enter to send message to 'ESP32C6 Dev Module' on '/dev/cu.usbmodem58A6072939 1')
New Line
9600 baud

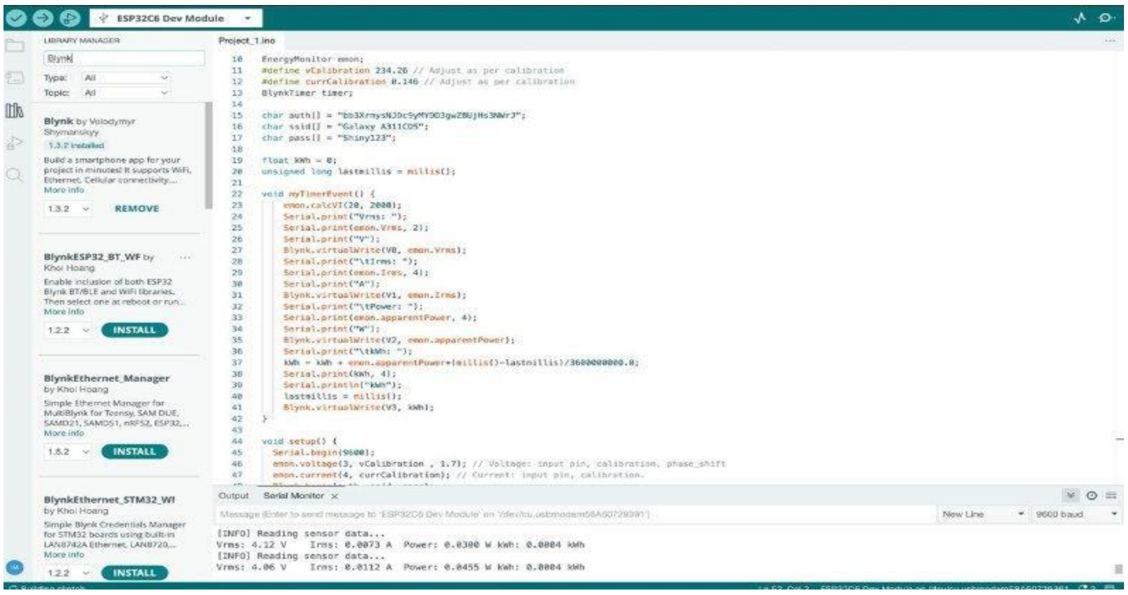
```
#define BLYNK_TEMPLATE_NAME "Embedded Project"

#include "EmonLib.h"   //https://github.com/openenergymonitor/EmonLib
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

EnergyMonitor emon;
#define vCalibration 500 // Adjusted as per calibration
#define currCalibration 0.146 // Adjusted as per calibration
BlynkTimer timer;

char auth[] = "bb3XrmysNJDc9yMYDD3gwZBUjHs3NWrJ";
char ssid[] = "Galaxy A315CD5";
```

```
[INFO] Reading sensor data...
Vrms: 36.85 V   Irms: 0.0184 A  Power: 0.6776 W kWh: 0.0027 kWh
[INFO] Reading sensor data...
Vrms: 37.93 V   Irms: 0.0194 A  Power: 0.7368 W kWh: 0.0027 kWh
[INFO] Reading sensor data...
Vrms: 37.12 V   Irms: 0.0180 A  Power: 0.6688 W kWh: 0.0027 kWh
[INFO] Reading sensor data...
Vrms: 35.66 V   Irms: 0.0171 A  Power: 0.6091 W kWh: 0.0027 kWh
[INFO] Reading sensor data...
Vrms: 51.41 V   Irms: 0.0254 A  Power: 1.3081 W kWh: 0.0027 kWh
[INFO] Reading sensor data...
Vrms: 51.17 V   Irms: 0.0261 A  Power: 1.3376 W kWh: 0.0027 kWh
[INFO] Reading sensor data...
Vrms: 51.19 V   Irms: 0.0269 A  Power: 1.3756 W kWh: 0.0027 kWh
[INFO] Reading sensor data...
Vrms: 53.94 V   Irms: 0.0268 A  Power: 1.4431 W kWh: 0.0027 kWh
[INFO] Reading sensor data...
Vrms: 50.85 V   Irms: 0.0253 A  Power: 1.2855 W kWh: 0.0027 kWh
[INFO] Reading sensor data...
Vrms: 50.84 V   Irms: 0.0251 A  Power: 1.2782 W kWh: 0.0027 kWh
[INFO] Reading sensor data...
Vrms: 50.38 V   Irms: 0.0256 A  Power: 1.2909 W kWh: 0.0027 kWh
[INFO] Reading sensor data...
Vrms: 51.64 V   Irms: 0.0272 A  Power: 1.4020 W kWh: 0.0027 kWh
[INFO] Reading sensor data...
Vrms: 52.76 V   Irms: 0.0266 A  Power: 1.4028 W kWh: 0.0027 kWh
[INFO] Reading sensor data...
Vrms: 52.54 V   Irms: 0.0262 A  Power: 1.3743 W kWh: 0.0027 kWh
```

Library of Blynk

The most widely used Internet of Things platform is Blynk, which allows you to manage your deployed items at scale, create apps to control them, and link any gear to the cloud. More than 400 hardware models, such as Arduino, ESP8266, and ESP32, can be connected to the Blynk Cloud using the Blynk Library. Download **Blynk Library**

It allows seamless integration with sensors and microcontrollers, enabling users to visualize real-time data and send commands from a smartphone or computer. With Blynk, developers can use virtual pins to exchange data between the hardware and the app. It supports Wi-Fi, Bluetooth, and GSM/3G communication protocols, making it versatile for a range of IoT applications. In energy monitoring systems, Blynk is used to display sensor data (e.g., voltage, current, and power) and control connected devices like loads remotely, offering a user-friendly way to manage and optimize energy **consumption.**



Blynk by Volodymyr Shymanskyy
1.3.2 installed
Build a smartphone app for your project in minutes! It supports WiFi, Ethernet, Cellular connectivity...
More info
1.3.2

BlynkESP32_BT_WF by Khoi Hoang
Enable inclusion of both ESP32 Blynk BT/BLE and WiFi libraries. Then select one at reboot or run...
More info
1.2.2

BlynkEthernet_Manager by Khoi Hoang
Simple Ethernet Manager for MultiBlynk for Teensy, SAM DUE, SAMD21, SAMD51, nRF52, ESP32,...
More info
1.6.2

BlynkEthernet_STM32_WI by Khoi Hoang
Simple Blynk Credentials Manager for STM32 boards using built-in LAN8742A Ethernet, LAN8720,...
More info
1.2.2

```
EnergyMonitor emon;
#define vCalibration 234.26 // Adjust as per calibration
#define currCalibration 0.146 // Adjust as per calibration
BlynkTimer timer;

char auth[] = "bb3XrmysNJDcSyMYDO3gwZBUjHs3NWrJ";
char ssid[] = "Galaxy A311CD5";
char pass[] = "Shiny123";

float kWh = 0;
unsigned long lastmillis = millis();

void myTimerEvent() {
    emon.calcVI(20, 2000);
    Serial.print("Vrms: ");
    Serial.print(emon.Vrms, 2);
    Serial.print("V");
    Blynk.virtualWrite(V0, emon.Vrms);
    Serial.print("\tIrms: ");
    Serial.print(emon.Irms, 4);
    Serial.print("A");
    Blynk.virtualWrite(V1, emon.Irms);
    Serial.print("\tPower: ");
    Serial.print(emon.apparentPower, 4);
    Serial.print("W");
    Blynk.virtualWrite(V2, emon.apparentPower);
    Serial.print("\tkWh: ");
    kWh = kWh + emon.apparentPower*(millis()-lastmillis)/3600000000.0;
    Serial.print(kWh, 4);
    Serial.println("kWh");
    lastmillis = millis();
    Blynk.virtualWrite(V3, kWh);
}

void setup() {
  Serial.begin(9600);
  emon.voltage(3, vCalibration , 1.7); // Voltage: input pin, calibration, phase_shift
  emon.current(4, currCalibration); // Current: input pin, calibration.
```

Message (Enter to send message to 'ESP32C6 Dev Module' on '/dev/cu.usbmodem56A60728391')

```
[INFO] Reading sensor data...
Vrms: 4.12 V    Irms: 0.0073 A  Power: 0.0300 W kWh: 0.0004 kWh
[INFO] Reading sensor data...
Vrms: 4.06 V    Irms: 0.0112 A  Power: 0.0455 W kWh: 0.0004 kWh
```

CHALLENGES FACED:

1. A significant challenge encountered during the project was the procurement of incorrect components. While the required products were ordered, mismatches in specifications or compatibility issues arose, such as receiving components with differing pin configurations or unsupported protocols. This setback caused delays as the incorrect components had to be returned or replaced, leading to additional costs and adjustments to the project timeline.

2. Use of Non-Invasive Current Sensor:

While working with the non-invasive current sensor, the lack of a female jack for its auxiliary output presented a significant challenge. To address this, bare wires were improvised to serve as the ground and line connections. However, this temporary solution introduced potential issues such as unstable connections and increased risk of noise interference in the signal. Despite these limitations, the approach allowed the project to proceed until the appropriate component could be sourced.

3. Calibration of ZMPT Voltage Sensor:

Calibrating the ZMPT voltage sensor presented a multifaceted challenge. Fine-tuning the sensor's potentiometer to ensure accurate voltage measurements required meticulous adjustments and repeated testing. Factors such as noise interference and variations in the power supply further complicated the process. Additionally, developing custom code for the microcontroller to process and interpret the sensor's output accurately was essential. This involved implementing algorithms to filter noise, scale the analog input to real-world voltage values, and ensure the readings were both reliable and precise.

PROTOCOLS USED:

1. GPIO (General-Purpose Input/Output):

Purpose: Used to control and read external components such as buttons, LEDs, or sensors.

How it works: GPIO pins on the ESP32 can be configured as either inputs or outputs. When configured as input, they read the state of a sensor (like a button press or voltage level). When configured as output, they control external devices (like turning an LED on or off).

Benefit: GPIO pins provide flexibility and allow for interactive control and sensing of physical components connected to the microcontroller.

2. Wi-Fi Protocol:

Purpose: Facilitates wireless communication between the ESP32 and the internet, enabling remote monitoring and control of the energy meter.

How it works: The ESP32 comes with built-in Wi-Fi, allowing it to connect to a local Wi-Fi network. This network connection enables it to send data to cloud platforms like Blynk or other servers over the internet.

Benefit: Wi-Fi connectivity allows you to access and control the energy meter remotely, view real-time data, and store historical consumption data.

3. Blynk :

Purpose: Used for communication between the ESP32 and the Blynk mobile application.

How it works: Blynk allows you to create a custom interface on a mobile app to control IoT devices. The ESP32 sends data (e.g., voltage, current, power) to the Blynk app using the Blynk library. The app then displays this data in real-time on various widgets (such as value displays or graphs).

Benefit: Blynk makes it easy to visualize data from the ESP32 on a smartphone, providing an intuitive and user-friendly interface for monitoring energy consumption.

4. ADC (Analog-to-Digital Converter):

Purpose: Converts the analog signals from sensors (e.g., *the current and voltage sensors)* into digital values that the ESP32 can process.

How it works: The ESP32 has built-in ADCs that convert the analog output from sensors like the SCT-013 current sensor and ZMPT101B voltage sensor into a digital signal. This digital signal is then used by the microcontroller for further calculations (e.g., calculating power or energy).

Benefit: ADCs enable the ESP32 to read and process real-world analog signals, such as voltage and current, to monitor energy usage accurately.

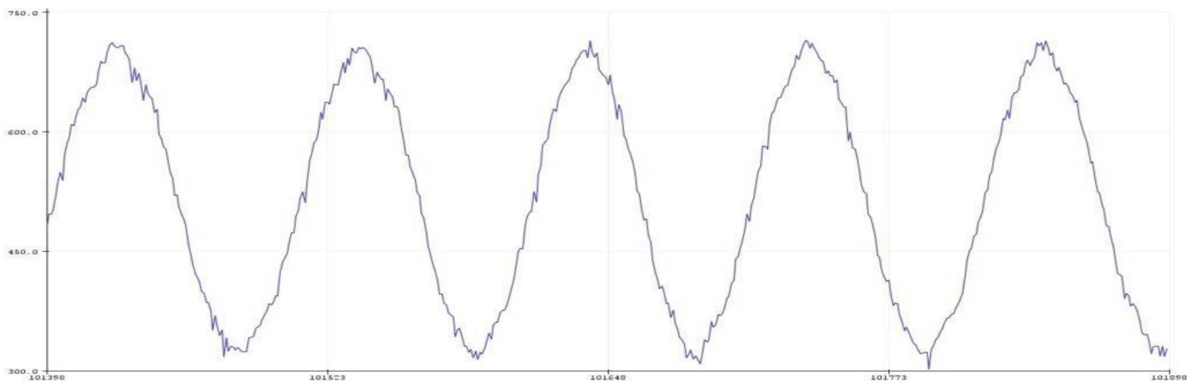
ZMPT101B Voltage Sensor Calibration

Since the ZMPT101B Voltage Sensor is not pre-calibrated, it must first be calibrated. An Arduino UNO/Nano Board can be used to calibrate the sensor. The Arduino UNO/Nano may be a viable option for calibration because of its flawless linear ADC pin. The sensor can be calibrated using the Arduino's analogue pin A0.

Now upload the following code to the Arduino Board.

```
void setup()

{

Serial.begin(9600);s

}



void loop()

{

Serial.println (analog Read(A0));

delay(100);

}
```

Once the code has been uploaded, launch the Serial Plotter. Rotate the potentiometer to calibrate the sensor if the sine wave is not displayed by the serial plotter. That indicate it is a proper calibration after it displays a sine wave.



IoT Smart Energy Meter Source Code/Program

This code is an Arduino sketch that uses an ESP32 board to construct an Internet of Things energy meter. The code makes use of a number of libraries, including Wi-Fi. EmonLib WIFI Client and BlynkSimpleEsp32

Paste the code below into your Arduino IDE after copying it.
Modify the lines that follow in this code. These lines contain the Blynk Authentication Token, WiFi SSID, and password. Put your credentials in their place.

Blynk and WiFi credentials

const char auth[] = "******************************";**

const char ssid[] = "******************************";**

const char pass[] = "******************************";**

Once the necessary line has been changed, you can upload the code to your ESP32 Board. Choose the ESP32 Development Module and the COM Port for it. Once the code has been uploaded, click the upload button.

```
#define BLYNK_PRINT Serial

#include "EmonLib.h"   //https://github.com/openenergymonitor/EmonLib

#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>



EnergyMonitor emon;

#define vCalibration 106.8

#define currCalibration 0.52

BlynkTimer timer;
```

```
char auth[] = "*******";


char ssid[] = "*******";

char pass[] = "*********3";


float kWh = 0;

unsigned long lastmillis = millis();


void myTimerEvent() {

   emon.calcVI(20, 2000);

   Serial.print("Vrms: ");

   Serial.print(emon.Vrms, 2);

   Serial.print("V");

   Blynk.virtualWrite(V0, emon.Vrms);

   Serial.print("\tIrms: ");

   Serial.print(emon.Irms, 4);

   Serial.print("A");

   Blynk.virtualWrite(V1, emon.Irms);

   Serial.print("\tPower: ");

   Serial.print(emon.apparentPower, 4);

   Serial.print("W");

   Blynk.virtualWrite(V2, emon.apparentPower);
```

```
    Serial.print("\tkWh: ");
    kWh = kWh + emon.apparentPower*(millis()-lastmillis)/3600000000.0;
    Serial.print(kWh, 4);
    Serial.println("kWh");
    lastmillis = millis();
    Blynk.virtualWrite(V3, kWh);
}

void setup() {
  Serial.begin(9600);
  emon.voltage(3, vCalibration, 1.7); // Voltage: input pin, calibration, phase_shift
  emon.current(4, currCalibration); // Current: input pin, calibration.
  Blynk.begin(auth, ssid, pass);
  timer.setInterval(5000L, myTimerEvent);
}

void loop() {
  Blynk.run();
  timer.run();
}
```

Code Explanation:

Key Components

1. **Libraries**

- EmonLib.h: Enables energy monitoring capabilities such as measuring voltage, current, and calculating power.
- **WiFi.h & BlynkSimpleEsp32..h**: Allow the ESP32 to connect to a Wi-Fi network and communicate with the Blynk IoT platform.

2. **Global Variables**
    - emon: An object of EnergyMonitor from EmonLib, responsible for energy calculations.
    - vCalibration & currCalibration: Calibration factors for voltage and current sensors.
    - kWh: Tracks cumulative energy consumption.
    - lastmillis: Stores the timestamp of the last measurement for energy calculation.

3. **Blynk Credentials**
    - auth, ssid, pass: Required for authenticating and connecting the ESP32 to the Blynk platform.

4. **BlynkTimer**
    - timer: Manages periodic function calls for data measurement and reporting.

myTimerEvent Function:

This function is executed every 5 seconds:

- **Voltage and Current Measurement**
    - emon.calcVI(20, 2000): Samples voltage and current 20 times within 2 seconds for accurate readings.
    - emon.Vrms and emon.Irms: RMS values for voltage and current are calculated.
- **Power and Energy Calculation**
    - emon.apparentPower: Apparent power (in watts) is calculated as **Vrms×Irms.**
    - Energy (in kWh) is calculated using: **kWh+=apparentPower×elapsed time (ms)/3,600,000,000** This converts power (watts) into energy over time.
- **Data Transmission to Blynk**
    - Virtual pins V0, V1, V2, and V3 are updated with voltage, current, power, and energy data, respectively, for display on the Blynk app.

2. setup Function

Executed once at startup:

- **Serial Communication**
    - Initializes serial communication at 9600 baud for debugging.
- **Sensor Configuration**
    - emon.voltage(3, vCalibration, 1.7): Configures the voltage sensor using GPIO pin 3, calibration factor, and phase shift.

    - emon.current(4, currCalibration): Configures the current sensor using GPIO pin 4 and calibration factor.
- **Blynk Initialization**
    - Connects the ESP32 to the Wi-Fi network and Blynk server.

- **Timer Setup**
    - Sets a timer to call myTimerEvent every 5000 milliseconds (5 seconds).

    3. loop Function

    Executed continuously:
- **Blynk and Timer Execution**
    - Blynk.run(): Maintains connection with the Blynk server.
    - timer.run(): Ensures periodic execution of myTimerEvent.

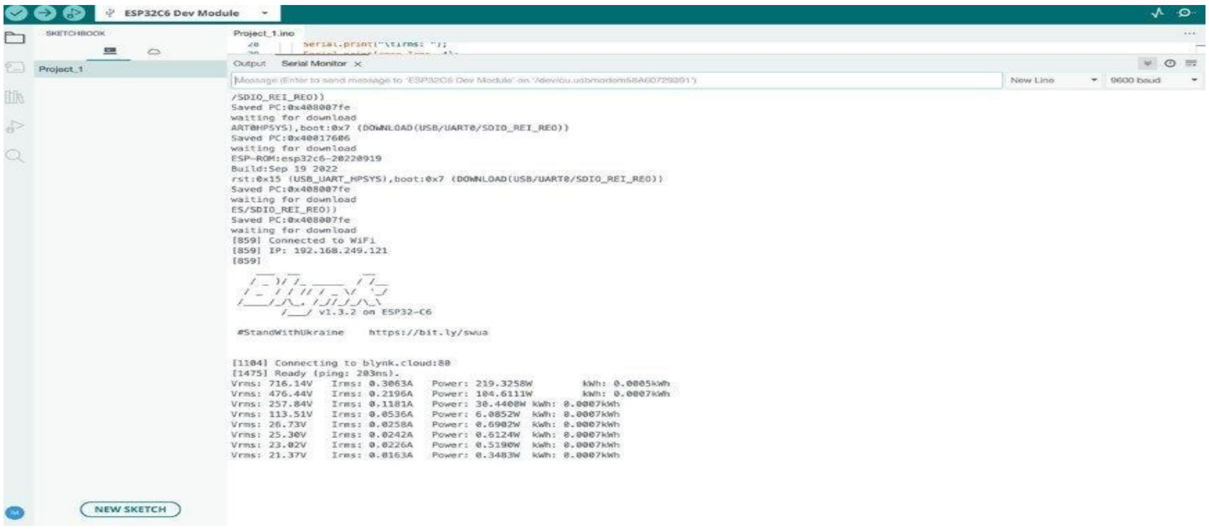
OUTPUTS

ESP32 IoT Energy Meter Data Testing on Blynk :

Using the provided SSID and password, the ESP32 Board we attempted to connect to the wifi network.

This program uses an ESP32 microcontroller to monitor energy usage and send the data to the Blynk platform. It utilizes the EmonLib library to measure voltage, current, apparent power, and cumulative energy consumption in kilowatt-hours (kWh). The myTimerEvent function, triggered every 5 seconds, calculates these values and sends them to the Blynk app via virtual pins. The setup function configures the sensors and Wi-Fi connection, while the loop function ensures continuous communication with Blynk. This setup is useful for remote home energy monitoring and IoT-based energy tracking.

SERIAL MONITOR READINGS



```
/SDIO_REI_REO))
Saved PC:0x408007fe
waiting for download
ART0HPSYS),boot:0x7 (DOWNLOAD(USB/UART0/SDIO_REI_REO))
Saved PC:0x40017606
waiting for download
ESP-ROM:esp32c6-20220919
Build:Sep 19 2022
rst:0x15 (USB_UART_HPSYS),boot:0x7 (DOWNLOAD(USB/UART0/SDIO_REI_REO))
Saved PC:0x408007fe
waiting for download
ES/SDIO_REI_REO))
Saved PC:0x408007fe
waiting for download
[859] Connected to WiFi
[859] IP: 192.168.249.121
[859]
    ___  __          __
   / _ )/ /_ _____  / /__
  / _  / / // / _ \/  '_/
 /____/_/\_, /_//_/_/\_\
        /___/ v1.3.2 on ESP32-C6

 #StandWithUkraine    https://bit.ly/swua


[1104] Connecting to blynk.cloud:80
[1475] Ready (ping: 203ms).
Vrms: 716.14V   Irms: 0.3063A   Power: 219.3258W        kWh: 0.0005kWh
Vrms: 476.44V   Irms: 0.2196A   Power: 104.6111W        kWh: 0.0007kWh
Vrms: 257.84V   Irms: 0.1181A   Power: 30.4408W kWh: 0.0007kWh
Vrms: 113.51V   Irms: 0.0536A   Power: 6.0852W  kWh: 0.0007kWh
Vrms: 26.73V    Irms: 0.0258A   Power: 0.6902W  kWh: 0.0007kWh
Vrms: 25.30V    Irms: 0.0242A   Power: 0.6124W  kWh: 0.0007kWh
Vrms: 23.02V    Irms: 0.0226A   Power: 0.5190W  kWh: 0.0007kWh
Vrms: 21.37V    Irms: 0.0163A   Power: 0.3483W  kWh: 0.0007kWh
```

Fig :Serial Monitor Output readings

The Serial Monitor displays the following real-time data:

1. **Vrms (Voltage RMS):** The Root Mean Square (RMS) voltage, which is a measure of the average voltage over time. It provides a more accurate representation of AC voltage than the average value.

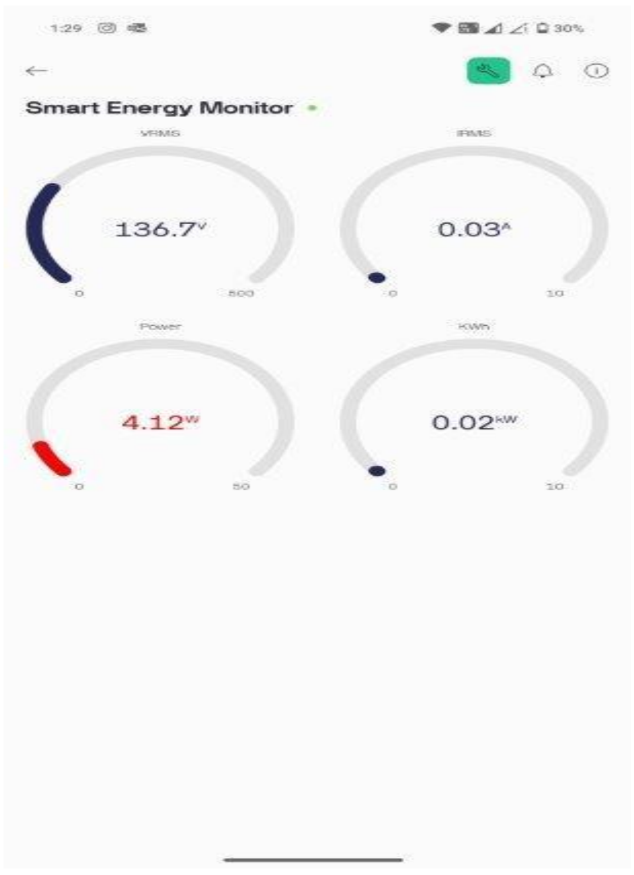
2. **Irms (Current RMS):** The RMS value of the current, calculated in the same way as voltage, representing the average current consumption.

3. **Power:** The apparent power (in watts) calculated as the product of the RMS voltage and current. This represents the total energy consumption at any given moment.

4. **kWh:** The cumulative energy consumption in kilowatt-hours, calculated over time. It is updated every 5 seconds based on the apparent power.

Observation : We observed that the data is printed every 5 seconds, showing updates for voltage, current, power, and accumulated energy consumption, helping with real-time monitoring and debugging as shown above in the picture.

Blynk App Output



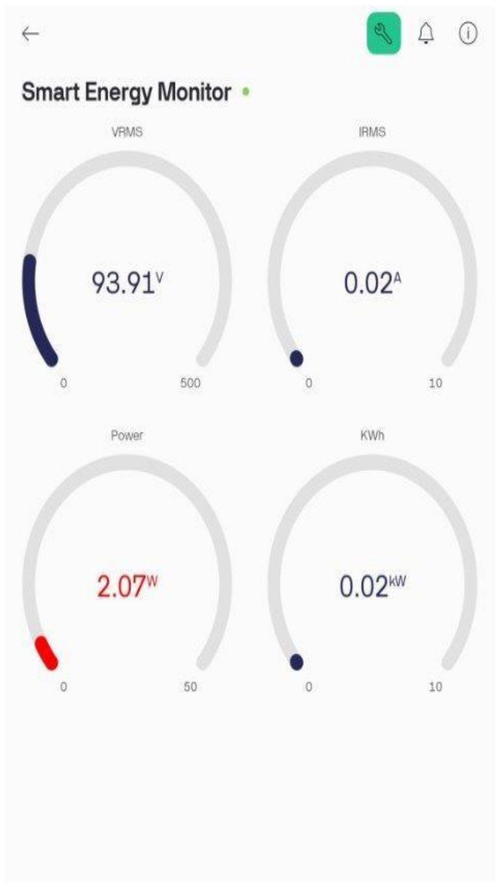


Fig : output readings data from Blynk app

Blynk App Output

In the Blynk app, the following data is displayed on the user interface:

1. **Vrms (Voltage):** Displayed on virtual pin V0, showing the RMS voltage in volts.
2. **Irms (Current):** Displayed on virtual pin V1, showing the RMS current in amperes.
3. **Power:** Displayed on virtual pin V2, showing the apparent power in watts.
4. **kWh (Energy Consumption):** Displayed on virtual pin V3, showing the cumulative energy consumption in kilowatt-hours.

Observation : we observed that the values are updated every 5 seconds, providing a live visualization of energy usage directly on the Blynk app, making it easy for users to monitor and track their energy consumption remotely.

REFERENCES:

1. **P**. M R and B. Bhowmik, "**Development of IoT-Based Smart Home Application with Energy Management,"** *2023 International Conference on Sustainable Communication Networks and Application (ICSCNA)*, Theni, India, 2023, pp. 367-373, doi: 10.1109/ICSCNA58489.2023.10370276.
2. N. Ashokkumar, V. Arun, S. Prabhu, V. Kalaimagal, D. Srinivasan and B. Shanthi, **"Design and Implementation of IoT based Energy Efficient Smart Metering System for Domestic Applications,"** *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, 2023, pp. 1208-1212, doi: 10.1109/ICACCS57279.2023.10113012**.**
3. Y. Siregar and E. Tandry, **"Monitoring System Design And Controlling Smart Home Features With The Blynk App,"** *2023 7th International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM)*, Medan, Indonesia, 2023, pp. 139-143, doi: 10.1109/ELTICOM61905.2023.10443083.
4. A. Othman and N. H. Zakaria, **"Energy Meter based Wireless Monitoring System using Blynk Application via smartphone,"** *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, Kota Kinabalu, Malaysia, 2020, pp. 1-5, doi: 10.1109/IICAIET49801.2020.9257827.
5. S. Jafar Tahir, P. Bihani and P. M. Tiwari, "Energy Metering and Controlling using Android and IoT based Smart System," *2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT)*, Kannur, India, 2022, pp. 931-936, doi: 10.1109/ICICICT54557.2022.9917727.

CONCLUSION:

- The IoT-based smart electricity energy meter developed in this project effectively combines modern IoT technology with traditional energy monitoring techniques to provide a smart and efficient solution. Using the ESP32 microcontroller, SCT-013 current sensor, and ZMPT101B voltage sensor, the system accurately measures electrical parameters such as voltage, current, power, and total energy consumption (kWh).
- The integration with the Blynk application enables real-time data visualization and remote monitoring, offering users convenience and insights into their energy usage. This system eliminates the need for manual meter readings, reducing errors and enhancing efficiency. Additionally, the project demonstrates a cost-effective and scalable design, making it suitable for both residential and commercial applications.