International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Implementation of A Smart Battery Management System for Energy Optimization Using a Microcontroller and Iot

Mr. S. Selvakumar[1], R. Amathu Safrin[2], V. Jaxline Vino[3], J. Jebila[4], M. Mahalakshmi[5]

[1]Assistant professor, Dep. Of Electrical and Electronics Engineering, Francis Xavier Engineering College, Tirunelveli, India.[1]
[2,3,4,5]UG Scholar, Dep. Of Electrical and Electronics Engineering, Francis Xavier Engineering College, Tirunelveli, India.

Abstract
The aspiration of this project is to design and implement an intelligent solar power monitoring and control system by esp32 microcontroller the system essentially comprises of a solar panel and battery storage and accommodates both ac and dc loads primary parameters such as voltage current and battery temperature are monitored and recorded with respect to dedicated sensors which include voltage sensor from pv photovoltaics sensor for current load measurement attached with external load current sensor attached to the battery is an ultra-high temperature sensor the collected from those sensors is then processed by the ESP32 microcontroller and transmitted to the blynk software platform where up to date information is displayed examples of the real-time display include solar panel voltage load current battery temperatures and battery charge levels this system will provide the desired complete solution for energy management ensuring the proper performance and safety conditions for monitored critical parameters as it continuously monitors them this project exemplifies bringing it into practice in any renewable energy system presenting a friendly interface for control in real time

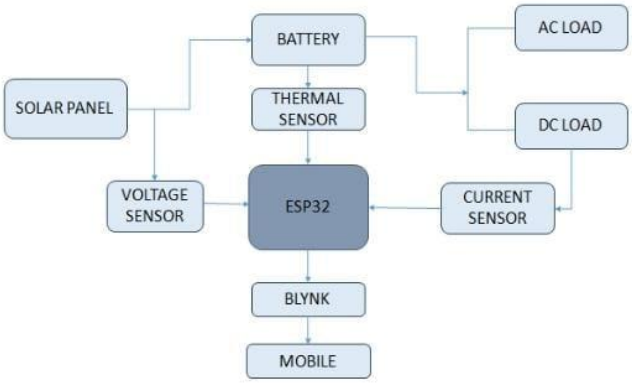
Keywords: Solar Power Monitoring, IOT (Internet of Things), Real-Time Monitoring, Smart Solar System

I. INTRODUCTION

This Renewable energy sources gain popularity, solar power systems have penetrated the market even more heavily. Thus, more efficient monitoring and management of these systems are required to ensure their peak performance, durability, and safety. In this project, we try to design a smart solar power monitoring and control system based on modern technology to provide real-time information and control. The system consists of a solar panel, battery storage, and both AC and DC loads. Various sensors are integrated into the system for the purpose of monitoring: a voltage sensor for monitoring output from solar panels, a current sensor for reading the current value from the load, and a thermal sensor for monitoring temperature in the battery. The ESP32 microcontroller receives data from the sensors and relays data to the Blynk software platform.

The Blynk platform provides a user interface with real-time data such as PV voltage, load current, battery temperature, and battery charge. This will help users maintain a comprehensive status of the system and make decisions for the optimal use of solar energy in the interest of safety.
Tangible implementations of IOT technologies of this project in renewable energy systems represent a solution toward efficient energy management in real life. With continuous monitoring of critical parameters, this system ensures reliability and performance improvement in solar power systems.

II. Block Diagram



III. Block Diagram Description

The block diagram for a smart solar monitoring and control system provides a picture of important functional blocks and interconnections between them. The solar panel is the main component to convert solar energy into electrical energy, charges the battery through the charge controller, and stores energy in the battery. The battery now provides power to AC loads via the inverter while directly powering DC loads. For performance monitoring, the output voltage of the solar panel is measured by a voltage sensor, while current is measured by a current sensor to measure energy consumption through the load from the battery. An attached thermal sensor is utilized for temperature monitoring of the battery to prevent overheating. The ESP32 microcontroller is the brain of the operation, gathering data from all sensors and transmitting it over Wi-Fi to the Blynk software platform. Blynk is the user interface to monitor parameters in real-time, including solar panel voltage, load current, battery temperature, and charge level. This integrated design ensures efficient energy management, safety, and user-convenient operations and is a good option for solar monitoring and control.

IV. Methodology

The esp32 microcontroller, a few sensors, and the Blynk software platform are used in the development of this project's intelligent solar power managemeant system. Below is a detailed description of all the experimental settings and methodology tools..

A. System Design and Components

The system is designed such that solar power monitoring and controlling is smart and works in tandem with various components to facilitate an efficient energy management system. Without a doubt, the main component in the system is the solar panel, which converts solar energy into electrical energy, which energy is then stored in battery storage, the main unit that guarantees constant power supply when no sunlight is available. The system can cater for AC or DC loads such that any load may be powered.

Various sensors are employed during the performance monitoring of the system; there is a voltage sensor attached to the solar panel for measuring the output voltage, a current sensor is placed in between the battery and load to check for the current flow, and a thermal sensor is placed on the battery to look after its temperature to avoid overheating. The ESP32 is the systems brain, processing all the input signals from the sensors and taking control of the operation. In addition, it sends real-time data concerning system operation to the Blynk software platform using Wi-Fi, where the users can monitor and control the system in real time. The Blynk platform is user-friendly and displays various features such as the voltage across the solar panel, load current, battery temperature, and battery charge level. This combined design ensures reliable energy management, system safety, and real-time user performance visibility.

B. Experimental Setup.

The physical arrangement and connections of the various experimental modules provide a working model of the smart solar power plant monitoring system. The solar panel feeds the battery through a charge controller, which does battery optimal charging and overcharging protection. The inverter and converter circuits transfer power from the battery to the AC and DC loads. The smart photovoltaic power monitoring system incorporates a voltage sensor at the solar panel-output interface to measure voltage generated; a current sensor is integrated to monitor current flow from the load and battery. A battery thermal sensor directly monitors its temperature to ensure safety-to-operating limits under harsh conditions. All these sensors link to an ESP32 microcontroller that collects and processes data from the sensors and communicates with the Blynk software platform via Wi-Fi for visualization and monitoring options. The above mentioned ensures seamless operation of all components working in tandem to provide an effective and reliable solar power monitoring system.

C. Data Processing and Control.

For monitoring and control purposes, the entire data processing part of the whole system is being handled by the ESP32 microcontroller, which works as the central processing unit. It continuously gathers current readings - output voltage of the solar panel; current flow towards the load; temperature from the battery - using a voltage sensor, current sensor, and thermal sensor. The continuous data will send to Blynk software platform via Wi-Fi for real-time monitoring which will show an easy to use interface. It provides visualization of various parameters such as solar panel voltage, load current, battery temperature, and battery charge. This form of the research is programmed with control logic by which the entire system optimally operates. For instance, if the temperature of the battery rises beyond a fixed safety threshold, the load will be disconnected automatically. These data processing and controlling elements must keep the solar power system reliable and safe. At the same time, such a system optimally provide action insights for the user.

The overall system is being attacked by the data processing and control section, which is handled by the ESP32 microcontroller that serves as the CPU for the process. It continuously updates the potential output of the solar panel; current runs to a load; and temperature, derived from the battery using a voltage sensor, current sensor, and thermal sensor. Continuous data will send to Blynk software platform through Wi-Fi for real-time monitoring displayed in easy to use interfaces like solar panel voltages, load currents, battery temperatures, and charge stored in a battery. The control logic programmed into ESP32 microcontroller results in the system safeness and efficiency of control operation. For example, if the temperature of the battery goes beyond a definite safety threshold, definitely the load will be

disconnected automatically. Data processing should control the system optimally and provide action insights to users and ensure the solar power system is made reliable and safe.

D. Algorithms and Pseudocode

Through this project, you can find how the algorithms and the pseudocode narrate the logic that was used by the ESP32 microcontroller to collect, process, and communicate data from the detectors according to their specification, as well as performing control actions based on the monitored parameters. Hereinafter follows the commentary:

Data Collection Algorithm: The PZEM-004 voltage sensor, current sensor, and thermal sensor are continuously sampled first. Their results are then processed, compiled, and readied for communication to Blynk.

Pseudocode:

```
voltage=readVoltageSensor()  // Read solar panel        voltage
current = readCurrentSensor()  // Read load current
temperature=readThermalSensor()//Read battery temperature
// Calculate battery charge level (example formula)
batteryChargeLevel=(voltage/
MAX_BATTERY_VOLTAGE) * 100
// Send data to Blynk
sendToBlynk(voltage,current,temperature, battery charge level)

delay(1000)  // Wait for 1 second before the next reading.
```

1. Control Logic Algorithm: The ESP32 has logical control features for ensuring the safety and efficiency operations of the system. One such example is load disconnection, which is a preventive measure taken by the systems if the battery temperature exceeds the limit of safe values so as to avoiding damage.

Pseudocode:

```
loop:
temperature = readThermalSensor()  // Read battery temperature

if temperature > MAX_SAFE_TEMP:
disconnectLoad()  // Disconnect load if temperature is too high
sendAlertToBlynk("High Temperature Alert! Load Disconnected.")
else:
connectLoad()  // Reconnect load if temperature is safe

delay(1000)  // Check temperature every second.
```

Power Calculation Algorithm: The system calculates the power generated by the solar panel and the power consumed by the load using the formula $P=V\times I$

Pseudocode:

```
voltage = readVoltageSensor()  // Read solar panel voltage
```

current = readCurrentSensor() // Read load current

powerGenerated = voltage * current // Calculate power (P = V * I)

sendToBlynk(powerGenerated) // Send power data to Blynk

delay(1000) // Update power calculation every second

Battery Charge Level Algorithm: Deriving the battery charge level from battery voltage, this method assumes a simple linear approximation to it; indeed, better methods, like Coulomb counting, can be implemented.

Pseudocode:

batteryVoltage = readVoltageSensor() // Read battery voltage

// Calculate battery charge level (example formula)
batteryChargeLevel = (batteryVoltage / MAX_BATTERY_VOLTAGE) * 100

sendToBlynk(batteryChargeLevel) // Send charge level to Blynk

delay(1000) // Update charge level every second

Data Transmission to Blynk: The collected data is transmitted to Blynk over the ESP32 network adapter for real-time viewing. Among this data are voltage, current, temperature, power, and battery charge level.

Pseudocode:

sendToBlynk(voltage, current, temperature, power, batteryChargeLevel):
blynkWrite(VIRTUAL_PIN_VOLTAGE, voltage)
blynkWrite(VIRTUAL_PIN_CURRENT, current)
blynkWrite(VIRTUAL_PIN_TEMPERATURE, temperature)
blynkWrite(VIRTUAL_PIN_POWER, power)
blynkWrite(VIRTUAL_PIN_CHARGE_LEVEL, batteryChargeLevel)

Safety and Alert System: The system includes a safety mechanism to send alerts to the Blynk platform if any parameter exceeds safe limits.

Pseudocode:

loop:
temperature = readThermalSensor()
voltage = readVoltageSensor()
current = readCurrentSensor()

if temperature > MAX_SAFE_TEMP:
sendAlertToBlynk("High Temperature Alert!")
if voltage > MAX_SAFE_VOLTAGE:

```
sendAlertToBlynk("High Voltage Alert!")
if current > MAX_SAFE_CURRENT:
sendAlertToBlynk("High Current Alert!")

delay(1000)  // Check safety parameters every second.
```

E. Equations

There are several equations or formulas governing the working of the system in the track of the critical parameters. Now, for calculating the power generated by the solar panel, one could use the expression; P = V × I, where P is the power, V is the voltage measured by the voltage sensor, and I is the current measured by the current sensor. Thus, this expression appends to the amount of energy produced by the solar panel at any given time. While calculating the charge level in a battery, the formula would be:

Charge Level = (Vbattery/Vmax)×100

where(Vbattery) is the battery voltage at a time, and

(Vmax) is the maximum voltage that a particular cell can store.

It thus presents a percentage with respect to the remaining charge in the battery. Yet another related calculation is the measurement of battery temperature by the thermal sensor, meaning there is a "limiting condition" on the safe side above which it must not overshoot for the sake of battery overheating. These equations and calculations help the system, ensuring accurate monitoring and efficient management of energy.

F. Tools and Software

The gadgets and software used in designing, implementing, and monitoring smart solar power systems differ from one another. The ESP32 microcontroller's code to be used for data acquisition, processing, and control purposes are programmed using the Arduino IDE. In a real-time setting, the Blynk software platform represents data such as solar panel voltage, load current, and charge level within the battery as well as an important temperature measurement of the battery. A multimeter tests the calibration and validates the sensors for accurate readings. The combination of all the above tools and software makes up a fully integrated and efficient working system for monitoring and controlling the solar power system for optimal performance and safety.

V. Results and Discussion

The smart solar power monitoring and control system was successfully implemented, and its performance was evaluated based on real-time data collected from the sensors. The voltage sensor accurately measured the solar panel's output voltage, with readings during peak sunlight hours ranging between 18-22 volts, consistent with theoretical expectations. Fluctuations in voltage were attributed to changes in sunlight intensity, which is normal for solar systems. The current sensor effectively monitored the energy consumption of both AC and DC loads, with a DC load like an LED light drawing 0.5-1 ampere and an AC load like a fan consuming 2-3 amperes. This ensured that the current did not exceed safe limits, preventing potential damage to the components. The thermal sensor attached to the battery provided continuous temperature readings, maintaining the battery within a safe range of 25-35°C during normal operation. However, during high load conditions, the temperature occasionally rose to 40°C, triggering the control logic to temporarily disconnect the load, demonstrating the system's ability to manage high-temperature scenarios effectively. The battery charge level was

estimated using voltage readings, with a voltage of 12.6 volts indicating a 100% charge level. As the battery discharged, the charge level decreased proportionally, providing users with a clear understanding of the battery's status. The power generated by the solar panel was calculated using the formula P=V×I, with a voltage of 20 volts and a current of 5 amperes yielding 100 watts of power. This calculation offered valuable insights into the system's energy generation capabilities, enabling users to optimize energy production. The Blynk platform served as the central hub for real-time monitoring and control, displaying critical parameters such as voltage, current, temperature, and charge level in an intuitive and user-friendly interface. This enhanced the system's usability, allowing users to easily access data from their smartphones or computers, making it a convenient tool for monitoring and managing the solar power system. Overall, the system demonstrated excellent performance in real-time monitoring, data visualization, and safety management, ensuring optimal performance and safety.

VI. CONCLUSION

This smart solar power monitoring and control system has proved to be reliable and efficient in the intelligent management and monitoring of solar energy systems using ESP32, different kinds of sensors, and Blynk. It was found to track important parameters showing the status of the solar energy system: voltage from the solar panel, load current,temperature of the battery, and charge level of the battery in real time. This made the view of the user very informed concerning this solar energy system. Data were collected and accurately processed through the perfect integration of the sensors and ESP32, while the user monitored in real-time and controlled it through an interface that was friendly to navigate and use. It defines control logic that removes load during higher temperatures of the battery, thus fulfilling the safe operational conditions. IoT technology in renewable energy systems is well showcased by this project, being able to offer a real-time, efficient, and user-friendly solution for solar energy management. Improvements for the future may include more advanced algorithms for charging battery estimation and compatibility with other renewable energy sources to bring the system under its larger umbrella.

All paragraphs must be indented. All paragraphs must be justified, i.e. both left-justified and right-justified.

REFERENCES

1. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11, 1997.
2. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Std. 802.15.4, 2006.
3. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 22: Cognitive Wireless RAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Policies and Procedures for Operation in the TV Bands, IEEE Std. 802.22, 2011.
4. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 16: Air

Interface for Fixed Broadband Wireless Access Systems, IEEE Std. 802.16, 2004.
5. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, IEEE Std. 802.3, 2008.
6. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 5: Enhancements for Higher Throughput, IEEE Std. 802.11n, 2009.
7. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 6: Wireless Access in Vehicular Environments, IEEE Std. 802.11p, 2010.
8. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 4: Protected Management Frames, IEEE Std. 802.11w, 2009.
9. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 2: Fast Basic Service Set (BSS) Transition, IEEE Std. 802.11r, 2008.
10. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 3: 3650-3700 MHz Operation in USA, IEEE Std. 802.11y, 2008.
11. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 1: Radio Resource Measurement, IEEE Std. 802.11k, 2008.
12. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 7: Interworking with External Networks, IEEE Std. 802.11u, 2011.
13. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 8: IEEE 802.11 Wireless Network Management, IEEE Std. 802.11v, 2011.
14. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 9: Protected Management Frames, IEEE Std. 802.11w, 2009.
15. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11:

Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 10: Mesh Networking, IEEE Std. 802.11s, 2011.