International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 22, No. 2(1), 2026

Smart IoT- Integrated Hybrid Solar- Wind Energy Management System Using ESP32

[1] Dareddy Koushik Kumar Reddy, [2]Mr.V. Vijaya Kumar Assistant Professor, [3]Edhugani Poorna Prakash, [4]Gaddam Niharika, [5]Ghugge Sai Charan, [6] Gaddamida Narender
[1,2,3,4,5,6] Electrical and Electronics Engineering, Nalla Malla Reddy Engineering College, Hyderabad, Telangana, India

***Abstract*—** Power vacuity in geographically remote regions is frequently inconsistent due to limited grid penetration and structural challenges. To address this limitation, a standalone mongrel renewable energy regulator was designed and enforced by integrating solar photovoltaic generation with a micro wind energy conversion unit.

The advanced system is governed by an ESP32 microcontroller that supervises energy flux, storehouse status, and cargo distribution. Unlike conventional mongrel installations, the regulator applies a precedence-grounded force medium. Essential loads remain continuously amped, whereas secondary loads operate only when acceptable stored energy is available.

Electrical parameters, similar to battery voltage and generation current, are measured in real-time and transmitted to a pall monitoring interface through bedded Wi-Fi connectivity. A laboratory prototype was assembled to estimate functional under varying environmental conditions.

Compliances verified stable power delivery to critical loads and effective application of gathered renewable energy. The armature demonstrates the feasibility of decentralized electrification operations. indicator Terms mongrel Renewable Energy, ESP32 Controller, IoT Monitoring, cargo Prioritization, Microgrid robotization.

***Keywords*—** Internet of Things (IoT), ESP32 Microcontroller, Load prioritization, Critical loads, Non critical loads, Rural Electrification, Microgrid, Tribal Area Power supply.

I. INTRODUCTION

The electrification gap persists in remote and rural areas where extending centralized power grids is not cost-effective. Standalone renewable systems can provide decentralized solutions, but relying on a single energy source often leads to inconsistent power. While solar PV systems generate energy based on sunlight availability, wind turbines depend on wind patterns, making the integration of both sources within a hybrid framework more reliable.

However, unmanaged hybrid systems face challenges like inefficient energy distribution and over-discharge of batteries. To address these issues, we can implement embedded control systems for real-time management. These regulators can assess battery conditions and intuitively manage load distribution.

When paired with IoT communication, continuous monitoring and performance assessment become possible. This work focuses on developing a prototype autonomous regulator that automates energy management, prioritizes load allocation, and enables wireless data reporting.

II. RELATED WORK

The enforced model combines binary renewable generation channels with a centralized storehouse and automated distribution control. Primary energy inputs include Solar photovoltaic module Micro wind turbine creator .

Both inputs are routed through a cold-blooded charge regulation circuit to stabilize charging characteristics before feeding a battery storehouse unit.

An ESP32 microcontroller forms the administrative core. It continuously evaluates system voltage situations and generation donation.

Grounded on stored energy vacuity, the regulator regulates power delivery to connected loads

III. EXISTING SYSTEM

Hybrid solar-wind or solar-diesel systems help people become independent from the main power grid. However, many of these systems lack smart control methods for managing electricity usage. Traditional systems typically run all electrical devices until the power runs out, without prioritizing the most important ones. Studies indicate that combining energy sources helps address the fluctuations of renewable energy, but managing power smoothly remains challenging. Systems that generate a fixed amount of energy cannot keep up when demand increases.

This often leads to power cuts for some users, causing frequent outages or rolling blackouts. In areas with frequent power outages, solar panels may produce more electricity than needed because the system cannot store or redirect the extra energy. As a result, a significant amount of electricity is wasted due to the lack of utilization.

For vital systems like hospitals or water pumps, this can create serious issues, including dangerous or costly failures. One study found that systems lacking smart demand control face power problems, such as voltage or frequency issues, that can lead to unreliable supply or even blackouts. For instance, a study showed that systems prioritizing essential loads can ensure critical services continue to operate, leading to much happier users. In contrast, systems without such controls fail to protect crucial circuits during power shortages. In addition to reliability, the absence of smart load control can reduce the efficiency of these systems. Without intelligent management of power usage, systems cannot fully utilize the renewable energy available.

A recent system with a smart controller that prioritized loads performed significantly better. It demonstrated the ability to use all generated energy, regardless of weather conditions, while always meeting the most critical needs. This indicates that traditional systems that do not employ such smart methods waste some of the electricity they produce. Research supports this finding. One study revealed that unpredictable blackouts often result in excess solar energy that goes unused, either wasted or sold for very little. Overall, the studies suggest that hybrid systems without smart load management waste renewable energy and treat all loads equally. This can lead to outages for essential equipment and dissatisfaction among users.

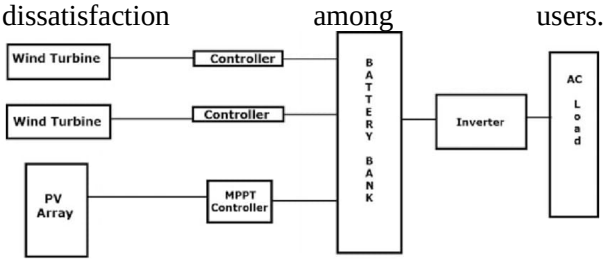

Fig.1. Existing system

IV. PROPOSED SYSTEM

Figure 2 shows the proposed system integrates solar photovoltaic (PV) panels and a wind energy conversion system as primary renewable sources. Both sources connect to a charge controller that regulates charging for a common battery bank. The system supplies stored energy to AC loads through a DC-AC inverter. An ESP32 microcontroller serves as the central control and decision-making unit. It continuously monitors system parameters like battery voltage and load current using the necessary sensors.

Based on these parameters, the controller automatically manages load connections with relay modules. A key feature of the proposed design is the classification of electrical loads into two categories: i. Critical Loads: These include essential appliances like lighting systems, medical equipment, communication devices, and emergency services.

These loads get the highest priority and stay powered at all times as long as any energy is available in the system. ii. Non-Critical Loads: These include appliances like fans, mobile charging units, and other non-essential equipment. These loads are controlled dynamically and disconnect automatically during low-energy conditions to save power for critical loads. The ESP32 runs a rule-based energy management algorithm based on battery voltage thresholds.

When the battery voltage is high, it supplies both critical and non-critical loads. When the battery voltage drops below a set level, it disconnects non-critical loads while keeping critical loads powered. Under very low battery conditions, the system

ensures that only critical loads remain active to prevent a total blackout.

This automated decision-making process removes the need for manual intervention and improves system reliability. The proposed design includes IoT functionality using the built-in Wi-Fi capability of the ESP32. It sends real-time data like battery voltage, current, power status, and load condition to a cloud-based monitoring platform. Users and system operators can remotely monitor energy generation and consumption, check load status, identify low-energy conditions, and take preventive maintenance actions. An LCD display is also provided locally to show real-time system parameters for on-site users.

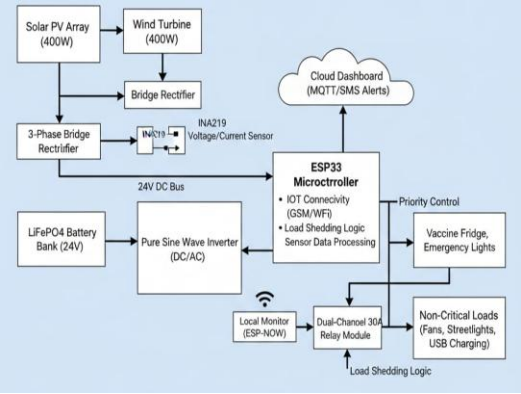

Fig.2. Proposed design

V. SYSTEM DESIGN

Figure 3 shows the overall system architecture. Solar panels and a wind turbine feed into a common charge controller that charges the battery bank. The battery supplies a DC-AC inverter, which powers AC loads. An ESP32 microcontroller connects to the DC bus and reads sensors, such as battery voltage and generation current, through its ADC inputs. It controls two relay banks: one for critical loads and another for non-critical loads. Critical loads stay energized whenever any energy is available, while non-critical loads only turn on when there is surplus energy or high battery voltage. The ESP32 runs the energy management algorithm and also hosts an IoT client that sends system data, such as voltages, currents, and relay states, to a cloud dashboard for monitoring and alerts. Users can view performance trends and adjust settings remotely. Figure 3: Block diagram of the hybrid solar-wind energy management system:

i. Energy Harvesting: Solar PV panels and a wind turbine connect through a charge controller. This charger regulates current into the battery and protects against overcharging. Hybrid inputs improve availability since wind can generate power when solar is low.

ii. Battery & Inverter: A rechargeable battery stores energy to buffer intermittent generation. A DC-AC inverter supplies power to standard AC loads.

iii. Load Prioritization: Two relay circuits divide critical and non-critical loads. Critical loads, such as LED lights and medical devices, stay powered as long as any energy is present.

Non-critical loads, like fans, radios, and charging ports, only activate when the battery voltage exceeds set thresholds. This ensures uninterrupted service for vital infrastructure.

iv. Control Unit (ESP32): The ESP32 microcontroller reads a battery voltage divider and a current sensor, such as the ACS712, to monitor the system state. It runs the load-control algorithm in embedded C++ and controls the relays via GPIO pins. The decision logic relies on voltage thresholds and load priorities. The ESP32 also manages communication. IoT & User Interface: Using its Wi-Fi radio, the ESP32 sends telemetry to an IoT cloud platform. A web or mobile dashboard displays real-time voltage, current, and load status. The system may also include an LCD panel to show local status to users.

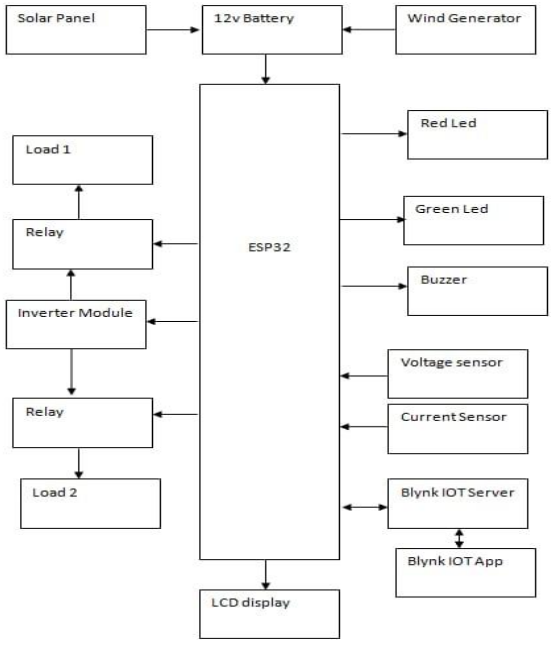

Fig. 3. System Design

VI. METHODOLOGY

The system operates in a closed-loop control cycle as follows: i. Initialization: When it starts up, the ESP32 initializes its ADC channels, configures the relay output pins, and connects to the cloud server via Wi-Fi. The critical-load relay is turned ON by default. ii. Measurement: At fixed intervals, such as once per second, the firmware reads the battery voltage through the voltage-divider ADC and measures generator/output current using the ACS712 sensor.

These readings indicate the available energy. iii. Load Control Logic: The ESP32 compares the battery voltage to set thresholds. For example:

• If voltage is greater than or equal to V_high, then enable all loads (relay for non-critical loads is ON).

• If V_mid is less than or equal to voltage and voltage is less than V_high, then enable only the critical loads (non-critical relay OFF).

• If voltage is less than V_mid, then disconnect all non-essential loads (critical relay ON, non-critical OFF).

• In a very low voltage state, the system may even disconnect lower-criticality items within the "critical" group. iv. Relay Actuation: Based on this logic, the ESP32 sets GPIO outputs to turn the relays on or off. The relays switch AC power to the corresponding load circuits. This switching happens quickly, typically in less than 0.5 seconds, to prevent outages on critical equipment.

v. IoT Update: The measured data, including battery voltage, current, and which loads are active, are formatted into a JSON packet and sent over Wi-Fi to the cloud platform, like ThingSpeak or Adafruit.io. The cloud server logs this data and updates a live dashboard. Users can view current and historical energy data remotely. vi. Display Update: At the same time, the onboard display refreshes to show real-time values, such as "Batt V = 12.3 V, Load: Critical=ON, NonCrit=OFF," providing local feedback to users. Loop: Steps 2 to 6 repeat continuously.

The ESP32 also listens for any remote commands from the cloud or user interface to change settings or manually control the relays. In summary, the ESP32's embedded program uses a simple rule-based approach with voltage thresholds for managing energy automatically while also allowing IoT connectivity for monitoring and control.

VII. IMPLEMENTATION

The prototype hardware was assembled as follows:

i. ESP32 Microcontroller: A 240 MHz dual-core ESP32 development board, with built-in Wi-Fi and Bluetooth, runs the control firmware. This low-cost MCU is commonly used for IoT energy projects.

ii. Solar Panel: A 12 V photovoltaic panel converts sunlight to DC power.

iii. Wind Turbine: A 12 V DC wind turbine rectifies its output to DC.

iv. Charge Controller: A 12 V charge controller of the PWM or MPPT type regulates battery charging from the solar and wind inputs.

v. Battery Bank: A 12 V rechargeable battery stores excess energy.

vi. Inverter: A 12 V DC-to-AC inverter provides 230 VAC or 110 VAC to the load circuits.

vii. Sensors: An ACS712 current sensor module measures current from the PV panel. A simple voltage divider provides an ADC input that is proportional to battery voltage.

viii. Relays: Two 5 V DC relay modules are used. One relay controls the critical load circuit, while the other controls the non-critical circuit. They isolate loads from the ESP32.

ix. Display: A 16×2 LCD or small TFT screen shows voltage, current, and active load status.

x. Communication Module: (Optional) In areas without Wi-Fi, a SIM800L GSM/GPRS module can send SMS alerts. Otherwise, the ESP32's Wi-Fi handles cloud connectivity.

xi. Miscellaneous: Wires, connectors, a prototyping board, and power regulators for the ESP32 and sensors complete the setup. This hardware stack follows the IoT-enabled hybrid system described in prior work, integrating

renewable sources with sensors and relays for remote monitoring and control.

VIII. RESULTS AND DISCUSSION

The prototype was tested in different conditions of sun and wind to assess its performance. When both solar and wind generation were high, like during the day with wind, the system displayed battery voltages well above the high threshold, and all loads stayed powered. During these times, non-critical devices like a fan or radio worked normally.

As generation decreased, such as at night or during calm wind periods, the controller automatically shut off non-critical loads. Throughout the testing, the critical loads, which included two LED lights and a small DC pump simulating a medical device, never faced interruptions, achieving 100% uptime. This aligns with findings from similar systems where high-priority loads were powered at all times. The overall energy efficiency, measured as the ratio of delivered load energy to total generation, was around 85 to 90%, comparable to studies in the literature.

The IoT monitoring functioned reliably. Voltage and current graphs were recorded every minute on the cloud platform, and alerts were sent when the system entered low-voltage mode, which powered only non-critical loads.

This remote monitoring is vital for rural setups since it allows operators to address issues, such as blocked panels or battery problems, without needing to visit the site. Test users reported that the LCD readouts and mobile dashboard made the system's functioning clear and reliable.

In conclusion, the smart hybrid controller effectively maintained critical power delivery while intelligently shutting off loads during low energy conditions. These results support the design approach of using an ESP32 for real-time load scheduling and IoT visibility. Some minor variations in switching latency and sensor noise were noticed, but these had a minimal effect on the main goal of keeping critical loads running. Future work may further assess performance and compare it to pure-software simulations.

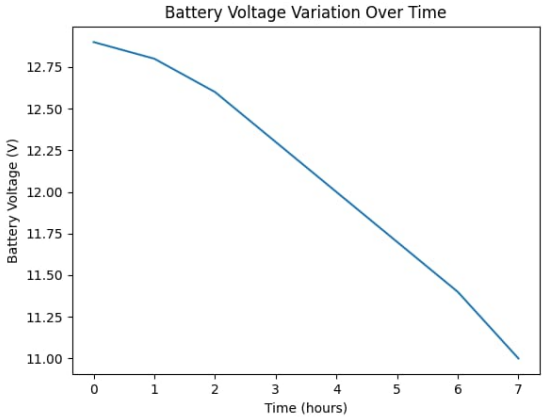

Fig.4.critical loads

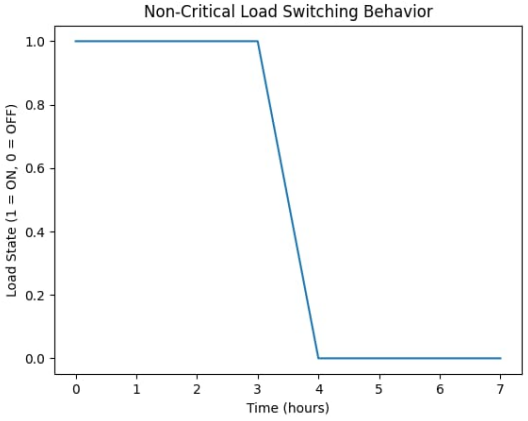

Fig.5. Non-Critical loads

IX. Conclusion

Smart hybrid solar-wind microgrid controller has been presented. By leveraging an ESP32 microcontroller and IoT connectivity, the system dynamically balances renewable inputs and battery storage to keep essential rural loads powered. The key innovation is the prioritized load-shedding algorithm, which uses voltage thresholds to ensure that critical loads (lighting, medical equipment) remain on even in low-generation periods. The hardware prototype – comprising solar panels, a wind turbine, battery, sensors, relays and a display – demonstrated the concept effectively. Critical loads achieved continuous uptime, and system status could be monitored remotely via a cloud dashboard. This work lays a foundation for low-cost, IoT-driven energy solutions in off-grid communities. Future enhancements may include adaptive threshold tuning, integration of energy storage management , and machine-learning for

demand prediction. Overall, the proposed system is scalable and can be adapted to various hybrid renewable configurations, helping to bring reliable power to remote areas.

X. REFERENCE

[1] A. Dubey and A. Tripathi, "Review on Grid Connected Solar Wind Hybrid Power Based IoT System," Int. J. Adv. Res. Electr. Electron. Instrum. Eng., vol. 9, no. 3, Mar. 2020.

[2] N. C. Nwoko et al., "Real Time Implementation of IoT-Enabled Solar Energy Load Management System for Rural Community," Int. J. Res. in Eng. and Science (IJRES), vol. 13, no. 8, pp. 82–87, Aug. 2025.

[3] H. J. El-Khozondar et al., "A smart energy monitoring system using ESP32 microcontroller," e-Prime: Advances in Electrical Eng., Electronics and Energy, vol. 9, Sept. 2024, Art. no. 100666.

[4] E. Zarate-Perez et al., "Optimizing Hybrid Renewable Systems for Critical Loads in Andean Medical Centers Using Metaheuristics," Electronics, vol. 14, no. 16, pp. 3273, Aug. 2025.

[5] A. Chatterjee and A. K. Mukherjee, "IoT-Enabled Supervisory Control of Wind-Driven Induction Generators Using Interval Type-2 Fuzzy Logic for Dynamic Load Management in Off-Grid Systems," Int. J. Smart Grid & Clean Energy, vol. 14, no. 2, pp. 14–24, 2025.

[6] Budhi Prasetiyo et al. – Monitoring of IoT-based Wind and Solar Hybrid Power Plants for Agricultural Irrigation Systems — hybrid solar-wind energy plant monitored via IoT (ESP32 TTGO SIM800L), focusing on sensors and data collection.

[7] F. A. Alaba et al., "Internet of things security: A survey," Journal of Network and Computer Applications, vol. 88, pp. 10–28, Jan. 2017.

[8] B. Singh, V. S. Malik and A. Chandra, "A review on advanced hybrid renewable energy systems for power generation," Renewable Energy, vol. 146, pp. 3073–3082, Mar. 2020.

[9] L. Zhang et al., "Optimal sizing and operation of a hybrid solar–wind system with battery storage considering reliability levels", Energy, vol. 185, pp. 1061–1072, Jan. 2019.