International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

AI Assisted Renewable Energy Performance Monitoring and Predictive Analytics Using ESP32

Vikas Maruti Pednekar[1], Prof. Swati Deshmukh[2], Dr. Ashwini Bade[3], Dr. Prabhat Pallav[4]

1 PG Student, Department of Electronics and Telecommunication Engineering
2 PG Coordinator, Department of Electronics and Telecommunication Engineering
3 Assistant Professor, Department of Electronics and Telecommunication Engineering
4 Head of Department, Department of Electronics and Telecommunication Engineering
Siddhant College of Engineering, Sudhambare, Talegaon - Chakan Highway, Sudumbre, Maharashtra 410501, India

---

Abstract - Photovoltaic output changes with irradiance, panel temperature, shading, orientation, and electrical loading. This paper presents an ESP32-based platform that measures voltage, current, temperature, and light intensity from a 12 V solar panel and transfers time-stamped records to a personal computer through USB serial communication. A Python pipeline validates and stores the measurements, computes power and energy, creates leakage-safe time-series features, and compares several regression algorithms for short-horizon energy prediction. A Flask dashboard integrates live monitoring, historical plots, data-quality warnings, model metrics, and forecasts. The proposed evaluation uses chronological splits, fixed preprocessing, and common metrics so that persistence, linear, tree, ensemble, and kernel models can be compared fairly. The work emphasizes calibrated measurement and traceable validation because prediction quality cannot be separated from sensor accuracy and dataset integrity.

Key Words: photovoltaic monitoring, ESP32, USB serial, predictive analytics, machine learning, Flask dashboard.

1. INTRODUCTION

Small photovoltaic installations are often assessed with occasional voltage readings or energy totals. Such observations do not explain why production changes because electrical output depends on several interacting conditions. Irradiance determines the available solar resource, cell temperature changes conversion efficiency, partial shading alters the current-voltage characteristic, and the connected load determines the operating point. A useful monitoring system therefore needs synchronized electrical and environmental measurements rather than a single isolated quantity.

Low-cost microcontrollers make continuous acquisition practical, but measurement quality remains the foundation of the system. The analog-to-digital converter in an ESP32 has finite resolution, attenuation-dependent behavior, reference variation, and an input range that must never be exceeded. Voltage dividers, current sensors, temperature sensors, and light sensors introduce their own offsets and scale errors. Without documented calibration, an apparently accurate machine-learning model can merely learn systematic sensor bias.

The proposed project connects a 12 V solar panel to protected sensing circuits and an ESP32. The controller converts raw samples into engineering units, checks their validity, assigns sequence numbers, and sends records to a PC through USB serial communication. The PC stores both raw and calibrated values. This separation preserves evidence for later recalibration and prevents a firmware change from silently changing the meaning of the dataset.

Machine learning is applied after measurement validity has been established. The prediction task is formulated as short-horizon power or interval-energy regression. Persistence, linear regression, decision trees, random forests, gradient boosting, and support vector regression are compared under identical chronological splits. A Flask dashboard presents measurements and forecasts while exposing stale, missing, saturated, or disconnected sensor states.

1.1 Contributions

The work defines an end-to-end architecture from solar sensing to a browser dashboard. It specifies measurement records, serial framing, storage, feature generation, model comparison, and operational warnings as parts of one traceable pipeline.

The paper also defines a comparison protocol that protects temporal order. Preprocessing and hyperparameter selection use only training periods, while later observations remain reserved for validation and testing. This structure reduces optimistic error estimates caused by future-data leakage.

Finally, the design keeps measured values, predictions, and data-quality states distinct. A model cannot overwrite an invalid reading, and the dashboard does not draw a continuous line through an unknown interval. This behavior makes the interface useful for engineering review rather than only for visual demonstration.

2. RELATED WORK

Photovoltaic monitoring standards describe performance quantities such as energy yield, reference yield, final yield, and performance ratio. These quantities become meaningful only when electrical production and environmental conditions share a consistent time base. NREL's PV performance guidance and PVWatts model demonstrate how irradiance, temperature, system rating, and losses contribute to expected energy production [2]-[4].

Embedded monitoring studies commonly use low-cost controllers to sample voltage, current, temperature, and irradiance proxies. The ESP32 is attractive because it provides analog inputs, timers, serial connectivity, and sufficient processing for filtering and protocol handling. Espressif's calibration driver compensates part of the reference-voltage variation, but system-level calibration against traceable instruments is still required [1].

Solar forecasting methods range from persistence and physical models to statistical and machine-learning approaches. Persistence provides a strong short-horizon baseline under slowly changing conditions. Linear models offer transparent coefficients, while tree-based methods capture nonlinear interactions without an explicit physical equation. Support vector regression can model smooth nonlinear relationships after feature scaling. Comparative studies must control preprocessing, horizon, and test periods because changes in these choices can dominate the reported ranking.

Time-series validation differs from random cross-validation. A random split allows neighboring observations from the same weather episode to appear in both training and test data, which makes the task easier than real forecasting. Chronological folds preserve operational order and provide a more credible estimate of how a model will behave on future periods [5].

Source	Focus	Use in this work
Espressif [1]	ADC calibration	Raw-to-voltage conversion
NREL [2]-[4]	PV performance	Energy and yield definitions
Scikit-learn [5]	Time-series validation	Chronological model comparison
Pallets [6]	Flask architecture	Dashboard organization

Table -1: Technical foundations used in the proposed system

3. SYSTEM ARCHITECTURE

The architecture contains four layers: the photovoltaic source and sensors, ESP32 acquisition, PC analytics, and the Flask dashboard. Each layer has a defined input, output, timing expectation, and failure state. This boundary-based design prevents a failure in one layer from being misinterpreted as a change in solar performance.

The 12 V panel connects to a measurement interface rather than directly to an ESP32 input. A resistive divider scales panel voltage below the permitted ADC range and includes protection appropriate to the prototype. Current can be measured with a Hall-effect sensor or shunt amplifier selected for the expected range. A contact temperature sensor measures panel-back temperature, while a calibrated light sensor or characterized irradiance proxy provides incident-light information.

The ESP32 samples channels at a stable interval, rejects impossible values, and reports saturation explicitly. Every record includes a timestamp or host-relative time, sequence number, raw counts, converted quantities, unit identifiers, calibration version, and quality bits. The controller calculates instantaneous electrical power as the product of synchronized voltage and current readings, but the PC retains the constituent measurements so that the calculation can be audited.

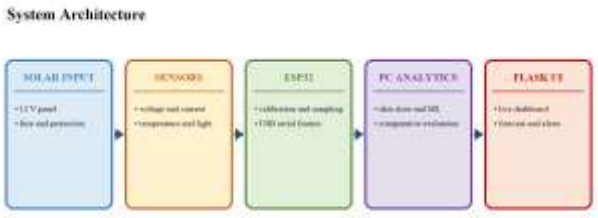


Fig -1: Overall monitoring and prediction architecture

3.1 Measurement Channels

Channel	Range	Stored values	Primary check
Voltage	0-18 V	ADC, V	Divider calibration
Current	Project rated	ADC, A	Zero and gain
Temperature	Ambient to hot panel	Raw, deg C	Reference comparison
Light	Dark to full sun	Raw, relative or W/m²	Linearity and saturation

Table -2: Proposed measurement record

Channel calibration uses paired readings from the prototype and a reference instrument over several operating points. A straight-line correction is acceptable only when residuals show no clear curvature. Temperature dependence and zero drift are recorded separately because a single gain factor cannot correct every error source.

Power is calculated only from voltage and current samples that belong to the same acquisition cycle. Interval energy uses numerical integration over valid power samples. When a time gap exceeds the permitted limit, the interval remains incomplete rather than being filled with an assumed value.

4. DATA ACQUISITION AND COMMUNICATION

Firmware starts with a known configuration and publishes its version, calibration identifiers, sample period, and reset cause. The acquisition task reads the sensors, applies bounded filtering, converts units, creates quality flags, and emits one record. Filtering reduces random noise but does not conceal step changes or disconnected sensors. Raw values remain available for diagnosis.

USB serial communication provides a simple wired link for the laboratory prototype. A line-oriented record can be encoded as comma-separated values or compact JSON. Each line begins with a recognizable version token and ends with a checksum when corruption detection is required. The host rejects malformed lines, duplicate sequence numbers, out-of-order records, and values outside configured limits.

The PC acquisition service opens the selected serial port, parses complete records, writes them to a durable file or database, and forwards recent values to the web application. Automatic reconnection is bounded and logged. If the device disappears, the dashboard changes the acquisition state to disconnected and retains the last valid timestamp without presenting the old value as live data.

Storage uses an append-only raw table and a derived analysis table. The raw table preserves received fields. The analysis table contains aligned intervals, engineered features, targets, and experiment identifiers. This separation supports repeatable processing and allows the same raw capture to be evaluated with revised feature definitions.

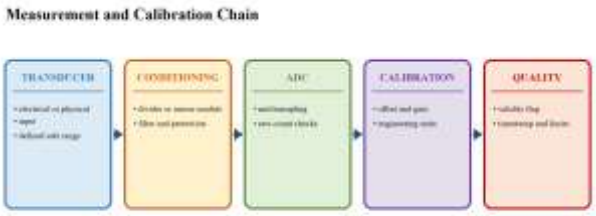


Fig -2: Measurement and calibration path

4.1 Data Quality States

Each observation carries explicit states such as valid, range error, ADC saturation, sensor unavailable, stale timestamp, communication error, or calibration unknown. Derived power is valid only when both electrical channels are valid and time-aligned. The forecasting pipeline excludes invalid targets and uses missing-value indicators where appropriate for predictor variables.

Quality monitoring also detects flat-lined sensors, implausible rates of change, and inconsistent combinations. For example, significant power during a dark light reading may indicate a light-sensor fault or a timing mismatch. These checks create warnings for investigation; they do not automatically rewrite the measured record.

5. PREDICTIVE ANALYTICS

The target can be defined as average power during the next interval or energy produced during a fixed horizon. The forecast horizon, sampling interval, and aggregation rule remain constant across all experiments. Target timestamps are shifted before the dataset is split so that future outcomes cannot enter current features.

Candidate predictors include present voltage, current, temperature, and light intensity; hour-of-day encodings; recent power lags; rolling mean and variability; recent energy; sensor-quality indicators; and optional panel configuration. The system will also retrieve online weather data from an API such as OpenWeather, including cloud cover, ambient temperature, humidity, and wind speed, and align it with sensor timestamps to improve solar-yield prediction. Feature calculations use only values available at the forecast issue time. Rolling windows remain left-aligned, and normalization parameters are fitted only on training data.

Persistence predicts that near-future production will match the most recent valid observation or recent mean. It establishes the minimum useful benchmark. Linear regression measures the value of a transparent additive relationship. A decision tree represents thresholds and interactions but can overfit. Random forests average many trees, while gradient boosting corrects residual errors sequentially. Support vector regression uses scaled predictors and a selected kernel to represent smooth nonlinear behavior.



Fig -3: Leakage-safe machine-learning pipeline

5.1 Validation Protocol

Observations are ordered by time and divided into expanding training folds followed by later validation periods. The final test period remains untouched until model and feature choices are fixed. All algorithms receive the same eligible rows, target definition, and folds. Hyperparameter tuning occurs inside the training portion of each fold.

Mean absolute error reports typical absolute deviation in the target unit. Root mean squared error gives greater weight to large misses. The coefficient of determination summarizes variance explained relative to a mean predictor but is interpreted with MAE and RMSE. Normalized errors may be added when experiments span different panel ratings or seasons.

Residuals are reviewed against time of day, light level, temperature, and rate of irradiance change. Aggregate metrics can hide systematic underprediction near peak output or unstable behavior during cloud transitions. Prediction latency, model size, training time, and dashboard update time are recorded because a small accuracy improvement may not justify a less maintainable system.

Model	Preprocessing	Main strength	Risk
Persistence	Recent valid value	Strong baseline	Misses rapid change
Linear	Scaling optional	Interpretable	Limited nonlinearity
Decision tree	Minimal	Threshold effects	High variance
Random forest	Minimal	Robust ensemble	Larger model
Gradient boosting	Minimal	Tabular accuracy	Tuning sensitivity
SVR	Scaling required	Smooth nonlinear fit	Cost on large data

Table -3: Comparative model evaluation plan

6. FLASK DASHBOARD

The Flask application separates serial acquisition, storage, analytics, and presentation. The acquisition process can continue when no browser is open. Routes or blueprints organize live status, history, experiments, and configuration. The application reads recent records from storage rather than accessing the serial port directly, which avoids competing readers and simplifies testing [6].

The live page displays voltage, current, power, temperature, light intensity, cumulative energy, last-sample age, and acquisition state. Historical plots share a common time axis so that changes in output can be compared with light and temperature. Forecast plots distinguish measured and predicted values and display the forecast issue time and horizon.

The model-comparison page lists the dataset version, folds, feature set, hyperparameters, MAE, RMSE, R^2, latency, and residual diagnostics. Results are not labelled as final when calibration is incomplete or the test period is too small. Export functions provide CSV data, experiment metadata, and figures for review.

Input validation and safe configuration are required even for a local prototype. The application restricts file paths and serial settings to permitted values, escapes displayed content, and does not expose a development server beyond the intended host. A production deployment uses a supported WSGI server and appropriate access controls.



Fig -4: PC software organization

7. EXPERIMENTAL PLAN

Hardware validation begins with safe-range tests using a controllable DC source before the solar panel is connected. Voltage and current channels are compared against reference instruments at several points. Temperature is checked in stable ambient and elevated conditions. The light channel is tested from dark to bright illumination while saturation and repeatability are recorded.

An outdoor acquisition campaign covers clear, variable-cloud, partial-shading, and temperature-change conditions. Configuration, panel orientation, load, date, time zone, calibration version, firmware version, and operator notes accompany each session. Records from setup changes are separated from stable operating data.

Software tests inject truncated records, invalid numbers, missing fields, sequence gaps, duplicate lines, and disconnect events. The expected result includes parser rejection or a specific quality state, a log entry, and recovery without corrupting previously stored data. Dashboard tests verify that stale values are labelled and that selected time windows return consistent aggregates.

Model experiments start only after the dataset passes completeness and plausibility checks. Each run records the code version, dataset checksum, target definition, fold boundaries, feature list, preprocessing parameters, algorithm settings, and metric values. Repeating a run with the same inputs should reproduce the same ranked results within numerical tolerance.



Fig -5: Experimental validation workflow

7.1 Acceptance Criteria

The measurement subsystem is accepted when reference comparisons remain within the documented tolerance over the intended range and no unsafe ADC input condition occurs. The communication subsystem is accepted when ordered records survive sustained acquisition and recover from a cable reconnection without silent duplication.

The analytics subsystem is accepted when chronological evaluation completes without leakage, every model uses identical test observations, and reported metrics can be reproduced from saved predictions. The dashboard is accepted when it displays current status, preserves invalid states, and links every forecast to a model and dataset version.

8. DISCUSSION

The proposed system joins embedded measurement and predictive software, but its credibility depends more on traceability than on the number of algorithms. A complex model cannot recover information that a saturated or poorly calibrated sensor never captured. For this reason, raw values, calibration versions, quality states, and experiment metadata remain first-class outputs.

Light intensity from a low-cost sensor is not automatically equivalent to plane-of-array irradiance. Its use as a predictor can still be useful after characterization, but energy-yield claims should distinguish a relative light proxy from a calibrated pyranometer measurement. Likewise, a 12 V laboratory panel represents the electrical behavior of a small prototype and does not establish performance for a full-scale installation.

Model ranking may change with season, horizon, aggregation interval, and available features. The project therefore reports results under stated conditions rather than declaring a universally best algorithm. Persistence may be difficult to beat at very short horizons, while ensembles may provide value when nonlinear temperature and irradiance interactions are represented in a sufficiently varied dataset.

The Flask dashboard supports interpretation by showing measurements, validity, and forecasts together. This design reduces the risk that a user treats a forecast as a direct sensor value. It also makes model failure visible when inputs fall outside the training range or when the last valid record becomes stale.

9. CONCLUSIONS

This paper presented an AI-assisted renewable-energy performance monitoring architecture based on a 12 V solar panel, calibrated sensors, an ESP32, USB serial communication, Python analytics, and a Flask dashboard. The design measures electrical and environmental conditions, preserves raw and calibrated records, and computes power and energy only from time-aligned valid samples.

The predictive workflow compares persistence, linear, tree, ensemble, and kernel regression methods using fixed features, chronological folds, and common metrics. The evaluation intentionally separates model development from the final test period and records all configuration needed for reproduction. Experimental accuracy will be reported only after hardware calibration and representative data collection are complete.

Future work can extend the acquisition period across seasons, add calibrated irradiance and weather inputs, estimate forecast intervals, detect model drift, and evaluate transfer between panels. These additions should retain the same traceability principles so that changes in hardware, data, and models remain distinguishable.

ACKNOWLEDGEMENT

The authors acknowledge the support of the Department of Electronics and Telecommunication Engineering, CAYMET's Siddhant College of Engineering. Student, guide, and laboratory details may be inserted before submission.

REFERENCES

1. Espressif Systems, ESP32 Series Datasheet and ESP-IDF ADC Calibration Driver Documentation, 2026.

2. A. P. Dobos, PVWatts Version 5 Manual, National Renewable Energy Laboratory, NREL/TP-6A20-62641, 2014.

3. B. Marion et al., Performance Parameters for Grid-Connected PV Systems, National Renewable Energy Laboratory, 2005.

4. National Renewable Energy Laboratory, PVWatts Calculator and Photovoltaic Data Tools, accessed 2026.

5. Scikit-learn Developers, TimeSeriesSplit, Model Selection, and Regression Metrics Documentation, 2026.

6. Pallets Projects, Flask Documentation 3.1, Application Factories, Blueprints, Security, and Deployment, 2026.

7. L. Breiman, Random Forests, Machine Learning, vol. 45, pp. 5-32, 2001.

8. J. H. Friedman, Greedy Function Approximation: A Gradient Boosting Machine, Annals of Statistics, vol. 29, no. 5, 2001.

9. C. Cortes and V. Vapnik, Support-Vector Networks, Machine Learning, vol. 20, pp. 273-297, 1995.

10. T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, 2nd ed., Springer, 2009.

11. IEC 61724-1, Photovoltaic System Performance - Part 1: Monitoring, current edition.

12. IEC 62548, Photovoltaic Arrays - Design Requirements, current edition.

13. Python Software Foundation, Python 3 Documentation, 2026.

14. PySerial Project, pySerial API Documentation, 2026.