WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

HYBRID SOLAR-WIND MPPT SYSTEM WITH PREDICTIVE LOAD FORECASTING

Dr. A. D. Bijwe[1], Manish Pillewan[2], Rohan Hiwase[3], Anshul Kashte[4]

Department of Electronics & Communication Engineering
Priyadarshini College of Engineering, Nagpur, India

ABSTRACT :

Solar and wind power are both uneven sources, and a turbine follows the wind speed and the way it's loaded, in the main; the panel and the turbine each go, through a power-conditioning circuit suited to that source, and an ESP32 microcontroller reads the electrical amounts and looks after the battery, the load and a small show, at least in practice. A panel is generating more or less resting on the sunlight reaching it and on its own temperature; in the next few pages a hybrid solar-wind system is suggested that uses maximum power point tracking (MPPT) and adds a load-forecasting stage on top. A machine-learning regressor is trained on logged load data so that it can estimate demand over a short window ahead; the controller can then use that estimate to prefer renewable supply, store surplus energy in the battery, and cut non-essential loads when energy is tight. We didn't have confirmed experimental logs or trained-model scores while writing this, so no quantitative gains are claimed here; those figures have to come from controlled tests on the finished prototype, and the sections below are covering the architecture, how measurements are taken, the model we have in mind, the build, and the test plan.

Index Terms, *hybrid renewable energy, solar energy, wind energy, maximum power point tracking, ESP32, load forecasting, battery storage, Random Forest, in many respects.*

I. INTRODUCTION

More people now want small power systems that are running their loads with less help from the standard grid, broadly speaking; what's more, that has pushed renewable sources into the spotlight. Solar photovoltaic (PV) and wind generation make a natural pair, since in many climates one tends to create when the other doesn't. Neither is steady on its own, though, at least in principle; indeed, PV output moves with irradiance, cell temperature and the point on its curve where the panel is operated. Better yet than that, wind output rests on the turbine, its generator and the wind itself; that's why, sizing a hybrid wind-PV framework means looking at the generation profile and the demand side together [1].

MPPT is the usual way of keeping a PV panel near the most power it can (in practical terms) give as conditions change, all things considered; the available schemes differ in what they need to sense, how fast they respond, how complicated they're, on the whole. And how well they're running [2]. Pairing MPPT with a battery and a supervisory controller is giving a applied way of matching what the sources yield to what the load asks for.

The prototype we suggest is employing a token 12 V, 20 W solar panel and a 12 V wind turbine, together with converter stages, voltage and current sensors, an ESP32, a battery and a load-management function. A forecasting model is added to support the controller's decisions, or better yet than reacting only to the load as it's right now, the controller can look at the projected demand and plan how to split renewable power and stored energy between the load and the battery.

We concentrate on the functional architecture, the measurement and data flow, the forecasting method, the setup and a test plan, at least for now. The forecast is treated as an estimate, not a promise. As, well, any claim about performance has to rest on measurements from the gathered hardware, at least empirically.

II. SYSTEM ARCHITECTURE

The design is split into the generation, power conditioning, measurement, storage, load control and forecasting, to a significant degree; and thus, solar and wind have diverse electrical behaviour, so each is getting its own conditioning stage; in turn, fig. 1 is showing how power and data move through it, in actual application. What's more, the two only are meeting at a prevalent DC bus after their voltage and current have been regulated.

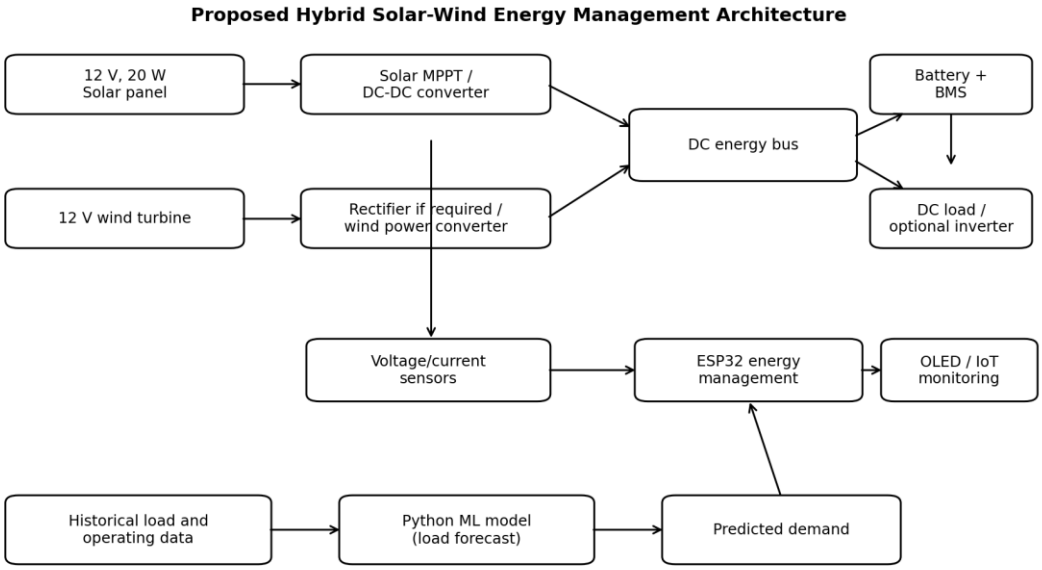


Fig. 1. As, well, functional architecture of the suggested hybrid solar-wind energy management system.

A. Solar and Wind Power Stages

The solar side is employing a small 12 V, 20 W panel feeding an MPPT or DC-DC converter, in real-world scenarios; the converter is holding the panel at a appropriate operating point and gives the DC bus or the battery charger a compatible voltage, all things considered. Real panel output can, after all, sit above or below the nameplate figure hinging on conditions and its rated limits, which is often why voltage and current need to be logged during the testing.

A token 12 V turbine is, as it stands, being employed by the wind side; indeed, have is stilled to verify what kind of generator it has and what its output waveform looks like, all things considered. If it turns out to be AC, a rectifier and some filtering will (at least in theory) be needed ahead of the DC conversion, to a significant degree, and what's more, a solar MPPT module shouldn't be supposed to work with a wind input until it has been inspected. The converter and control strategy on this side have to suit the turbine.

B. Sensors and Embedded Controller

Voltage and current sensors sit at the chosen points of the circuit and measure the sources, the battery and the load, to a significant degree. Before any reading is trusted, the sensor polarity, range, calibration and sampling interval should be checked, along with whether the sensor's output voltage is safe for the ESP32 input. Power at any instant follows from multiplying the two readings; these measurements are taken in by the ESP32, employs the energy-management rules we set, and can pass status to an OLED show or to an optional monitoring dashboard.

C. Battery Storage and Load

The extra energy goes into the battery, when generation is higher than demand. Warnings raised in software can't stand in for protection built into the hardware. An inverter is optional, and if one is employed it has to be matched to the battery, converter, wiring, fuse and BMS current ratings. When the sources fall short, the battery differs; in turn, for a low-power prototype a DC load is the sensible choice. The cell chemistry and the series/parallel layout of the pack must be matched by the charger and the battery-management system (BMS).

D. Data and Control Flow

The measurements are being gathered by the ESP32 and hands them to the control logic, so historical load records are carried out on a computer, where the forecasting model is trained, in the main; indeed, even as the framework is running, the forecast has to reach the controller over a named link, to take one instance serial or Wi-Fi, hinging on how the firmware is written, in the main. A reading that's missing, invalid or unlikely should be logged and kept out of the prediction; it shouldn't be quietly replaced with a normal-looking value.

III. MACHINE-LEARNING METHOD

Load forecasting is treated as a supervised regression problem; every training example is pairing a set of attributes known at prediction time with a target, which is the demand over a fixed future interval, at least for now. Random Forest is our candidate model, including it averages many decision trees and can, to some extent, represent nonlinear relationships between the inputs [3], to a significant degree; the choice of model and the precision we report must come from testing; in turn, on top of that, the name of the algorithm is proving nothing by itself.

A. Input Features and Prediction Target

The traits we have in mind are the latest load power, earlier (lagged) load readings, the time of day, solar power, wind power and whatever battery measurements are available, in real-world scenarios; only magnitudes the project really records should be employed. The target can, in practice, be the average load power over the next fixed interval, such as the following measurement window, in real-world scenarios, and the interval and the units are needing to be written down, so that training and live prediction are using exactly the same definition.

The feature vector can be written as:

$$x_t = [Pload(t-1), Pload(t-2), Psolar(t), Pwind(t), Vbat(t), time(t)] \quad (1)$$

This is only an example, and the final feature set should follow whatever the prototype actually logs, at least in practice, and battery state of charge, for one, shouldn't be listed as an gathered attribute unless it's measured directly or estimated reliably, in many respects.

B. Data Preparation and Training

Every load reading should carry a timestamp and use uniform units. The dataset is then verified for missing values, duplicate entries, sensor faults and outliers caused by communication errors, in actual application. Targets are built from later load readings, according to the forecast horizon we pick, in actual application. Time should be followed by the split between training and test data, so that nothing from the future leaks into the training. Testing on a separate session says more about how well the model generalises, if a few test sessions exist.

A Random Forest regressor is giving its estimate by combining the outputs of many trees, on the whole, and settings such as the number of trees and the maximum depth should be chosen, through the validation and recorded so others can (at least in most settings) reproduce the finding, at least for now. A model trained on synthetic or randomly made data is, in effect, fine for inspecting that the software works end to end; indeed, in plain words the same, it doesn't show that the model can forecast a real load.

C. Check Metrics

Mean Absolute Error (MAE) is the average size of the forecast error:

$$MAE = (1/N) \Sigma_{i=1}^{N} |y_i - \hat{y}_i| \quad (2)$$

Root Mean Squared Error (RMSE) penalises large deviations more heavily:

$$RMSE = \sqrt{[(1/N) \Sigma_{i=1}^{N} (y_i - \hat{y}_i)^2]} \quad (3)$$

In these expressions y_i is the measured load power, $\hat{y}_i$ is the forecasted value and N is the number of test samples, on the whole; indeed, MAE and RMSE are also in watts, when the target is load power in watts, broadly speaking. If the coefficient of determination R^2 is given, it should be read together with the error metrics and its limits kept in mind; before they have been computed on a held-out test set, none of these scores should be quoted.

IV. IMPLEMENTATION

A. Hardware Configuration

Table the primary parts are listed chosen for the prototype. The exact current ratings, converter limits and sensor models still have to be verified against the datasheets of the modules we buy, at least empirically.

TABLE I. PRINCIPAL PROTOTYPE COMPONENTS

Component	Function in the framework
12 V, 20 W solar panel	Solar power source
12 V wind turbine	Wind power source; rated wattage to be validated
Solar MPPT module / DC-DC converter	Controls the solar operating point and conditions the voltage
Wind rectifier/converter, if required	Conditions the turbine output for the DC bus
ESP32	Sensor acquisition, control decisions and communication
Voltage and current sensors	Measure source, battery and load amounts
3S, 2000 mAh battery and compatible BMS	Energy storage and cell protection
N-channel MOSFETs, Schottky diodes, relay module	Switching, blocking and load control as designed
DHT11 and OLED show	Optional environmental sensing and local show
Optional 12 V-to-220 V inverter	AC output, only if it suits the battery and load rating

B. Battery Calculations

Take a 3-series lithium-ion pack built from cells rated 3.7 V nominal and 2.0 Ah, on the whole. Its token voltage is:

$$V_{nom} = 3 \times 3.7 = 11.1 \text{ V} \quad (4)$$

Cells in series add up their voltages. Though, the capacity stays at about 2.0 Ah. The small energy then works out to:

$$E_{nom} = V_{nom} \times C = 11.1 \times 2.0 = 22.2 \text{ Wh} \quad (5)$$

The fully charged pack is reading 12.6 V. These numbers come from the stated nominal cell ratings and aren't measured battery behaviour with a maximum charging voltage of 4.2 V per cell, in real-world scenarios; and thus, the cell datasheet and the BMS in use must be followed by the real operating and

charging limits, at least in principle.

C. Firmware and Software

The ESP32 firmware can be written in C/C++ with the Arduino framework; it would read the sensors, check the readings, refresh the show, use the energy-management rules and, if the model runs on a computer, exchange data with the forecasting application, at least for now. The final setup is needing to state the serial or Wi-Fi protocol, the sampling interval and where the model runs, on the whole. And more, scikit-learn can train and gauge the Random Forest regressor [4], [5], and python with Pandas and NumPy can organise the logged data on the computer side.

The control rules should be tied to limits that have, broadly speaking, actually been checked. A token 200 W inverter shouldn't be anticipated to deliver its full rating from a 3S, 2000 mAh pack, because the DC current required can be very large and rests on the pack, the BMS, the converter and the losses, at least for now. A properly rated charger and protection circuit is, to a large extent, being needed by the battery; in fact, failing that the whole power path has been inspected, a low-power DC demonstration load is the safer option, in real-world scenarios.

V. EXPERIMENTAL Check AND RESULTS

The check should keep three things apart, like rated component specifications, values found from them, and measurements taken from the gathered framework, in real-world scenarios; the forecasting model should be tested on data that wasn't employed to fit it, to a significant degree. Bluntly, a controlled test should log solar and wind voltage and current, battery voltage and current, load power, ambient conditions, and the input and output power of each converter, on the whole.

A. Prototype Test Procedure

More correctly, employed, check every connection, the sensor ranges, the fuses, the battery polarity and the converter output limits, before power is, all things considered. Under diverse light conditions, then record solar measurements, and wind measurements, under the recorded turbine conditions, at least in practice. After that, observe how the combined system reacts when the load changes, when the battery is charging and when source availability varies. Across representative conditions, for the forecasting test, collect timestamped load data, train the model on an earlier period and gauge it on a later period it hasn't seen, in many respects. Repeating the runs is recommended.

B. Known Specifications and Evaluation Status

TABLE II. KNOWN SPECIFICATIONS AND EVALUATION STATUS

Parameter	Value or status
Solar panel rating	12 V, 20 W nominal (recommended component)
Wind turbine rating	12 V; rated power not offered
Battery small voltage	11.1 V (found for 3S, 3.7 V cells)
Battery capacity	2000 mAh (stated component)
Battery token energy	22.2 Wh (found)
3S full-charge voltage	12.6 V, supposing 4.2 V per cell
MPPT performance / efficiency	Requires measured input-output data
Load forecast MAE / RMSE / R^2	Requires measured held-out test data
Actual operating time	Not established; needs load testing

C. Illustrative Energy and Efficiency Calculations

A rough runtime is following from dividing the token battery energy by the load power for a constant load, at least in practice. Taking a 5 W load as an example:

$$t_i d_{eal} = 22.2 \text{ Wh} / 5 \text{ W} = 4.44 \text{ h} \quad (6)$$

This is an ideal figure, not a measured runtime. When you try it, the usable capacity, converter losses, the battery cut-off voltage, temperature and a changing load all are decreasing it, broadly speaking.

Converter efficiency should be worked out from input and output power measured at the same moment:

$$\eta = (P_{out} / P_{in}) \times 100\% \quad (7)$$

No measured power records, forecast error values or checked runtime results were available when this draft was prepared; this section therefore knownly specifies and the suggested way of finding them; it doesn't report experimental success, all things considered; quantitative results should be added only once testing is done.

VI. DISCUSSION

The design brings together source-specific power conditioning, battery storage, ESP32-based monitoring and a short-term load forecast, at least for now; the MPPT stages deal with the shifting operating points of the renewable sources. And the energy-management logic is matching what's available to what the load and battery need, in many respects; in fact, a forecast can betterly inform. And yet, it doesn't raise efficiency on its own, so plus, the controller has to make good use of it; what's more, the prediction has to be precise enough to be worth employing.

How well the framework does will rest on the turbine's true power rating, the behaviour of both converter stages, sensor calibration, battery limits, the load profile, and how much good historical data there's. As, well, the predictive approach should be checked with a uncomplicated baseline controller under the analogous source and load conditions. A model fitted on a small or synthetic dataset may, at least in theory, not carry over to real conditions, to a significant degree. Okay then the forecast should be checked on chronological holdout data.

The prototype should be handled as an experimental low-power system; in turn, battery charging, overcurrent protection, short-circuit protection, wiring and inverter sizing all are needing independent verification, to a significant degree; the final report should keep found ratings separate from measured values. As, well, it shouldn't conclude that a fault or safety problem exists just, because a sensor gave one odd reading, at least empirically.

VII. CONCLUSION

The paper recapped the suggested architecture and the evaluation plan for a hybrid solar-wind MPPT system with predictive load forecasting. The design is employing a 12 V, 20 W solar panel, a 12 V wind turbine, conditioning stages suited to each source, an ESP32 controller, voltage and current sensing, a protected battery and a load-management function. Future work should prove the framework with repeatable tests and with load data the model hasn't seen before any quantitative refinement is claimed. The token voltage and energy of the 3S, 2000 mAh battery can be found from its specification. A Random Forest regression model is considered for anticipating future demand from recorded load and operating data. But the real charging behaviour, runtime, converter efficiency and forecast accuracy must all be measured.

REFERENCES :

[1] W. D. Kellogg, M. H. Nehrir, G. Venkataramanan, and V. Gerez, “Generation unit sizing and cost analysis for stand-alone wind, photovoltaic, and hybrid wind/PV systems,” IEEE Transactions on Energy Conversion, vol. 13, no. 1, pp. 70-75, Spoil. 1998, doi: 10.1109/60.658206.

[2] T. Esram and P. L. Chapman, “Comparison of photovoltaic array maximum power point tracking techniques,” IEEE Transactions on Energy Conversion, vol. 22, no. 2, pp. 439-449, Jun. 2007, doi: 10.1109/TEC.2006.874230.

[3] L. Breiman, “Random forests,” Machine Learning, vol. 45, pp. 5-32, 2001, doi: 10.1023/A:1010933404324.

[4] F. Pedregosa et al. “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[5] Scikit-learn developers, “RandomForestRegressor,” scikit-learn documentation. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[6] Espressif Systems, “ESP32 Series Datasheet.” [Online]. Available: https://documentation.espressif.com/esp32_datasheet_en.html

[7] N. Mohan, T. M. Undeland, and W. P. Robbins, Power Electronics: Converters, Applications, and Design, 3rd ed. Hoboken, NJ, USA: Wiley, 2003.