DynaES: Dynamic Energy Scheduling for Energy Harvesting Environmental Sensors

Jianwei Hao, Emmanuel Oni, In Kee Kim, and Lakshmish Ramaswamy
School of Computing, The University of Georgia, Athens, GA, USA
{jhao, mosopefoluwa.oni, inkee.kim, laksmr}@uga.edu

***Abstract*—Low-cost sensors and IoT technologies have facilitated the deployment of environmental sensors to collect and analyze various factors, such as soil properties. Due to the lack of electric power networks in many deployment locations, these sensors rely on energy harvesting (EH) systems that use natural energy sources such as solar power. Specifically, solar-powered EH systems benefit from timely obtaining weather information for efficient future energy scheduling. However, obtaining weather information for EH environmental sensors is always challenging, as they are commonly deployed in harsh environments without network access.**

To address this problem, we present `DynaES`, a novel energy scheduling method for EH sensors without relying on online weather forecasts. `DynaES` comprises two components: a DC power gain estimator that predicts future power gain by individually estimating changes in environmental parameters and ensembling them, and a dynamic energy scheduler that distributes energy to sensors based on priority and adjusts sensing intervals and frequency. We evaluate `DynaES` via simulation-based studies on real-world datasets and compare its performance against state-of-the-art baselines. Evaluation results show that `DynaES` accurately predicts future energy gain with low estimation errors and enables 1.8× – 4× more frequent sensing operations with shorter sensing intervals while achieving longer operation hours without complete battery drains.

***Index Terms*—IoT; Energy Harvesting; Energy Scheduling;**

I. Introduction

With the emergence of low-cost sensors and IoT technologies, various environmental sensors are increasingly developed and deployed to monitor, collect, store, and analyze various environmental factors [1]–[3], such as soil [4], [5] and ecological properties [6]. One of the most critical design considerations when deploying such systems is ensuring that the sensors have a stable and sufficient power supply to remain operational. This becomes particularly challenging for deployment places where electric power networks are not available [6].

Energy harvesting (EH) is a promising approach for powering sensors by converting and storing energy from natural sources [7], [8]. In particular, solar energy is a widely used and convenient power source, and various solar panels are well-operational with computing boards like Raspberry Pi and Arduino, as well as IoT sensors. However, the stability of solar energy supply is subject to weather conditions. For example, deployed sensing systems may face multiple rainy and cloudy days, resulting in an energy shortage and discontinuation of sensor operations. Therefore, as shown in Fig. 1, EH sensing systems are typically equipped with energy storage units (ESUs), such as portable battery packs, that are directly connected to the sensors/computing boards. These units regularly recharge during sunny days to ensure a stable energy supply and continuous operation of the sensors.

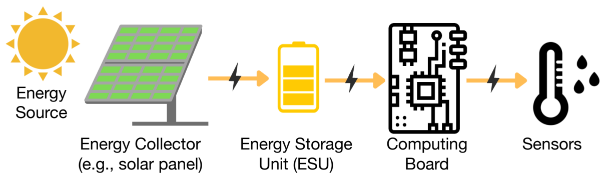

Energy Source
Energy Collector (e.g., solar panel)
Energy Storage Unit (ESU)
Computing Board
Sensors

Fig. 1. Energy harvesting sensing pipeline

The instability of the power supply poses another challenge for solar-powered environmental sensors, as these sensors are often deployed in harsh environments (*e.g.*, wetlands, reservoirs) to collect environmental factors. Unfortunately, these areas often lack reliable network access, making it difficult to obtain timely weather forecasts from well-known weather services [9], [10], thereby hindering the effective management of charged energy levels and power distribution to the sensors. This can result in frequent interruptions and delays in sensor operations to collect target sensing data. Therefore, it is crucial for EH environmental sensors to manage their energy capacity intelligently [11] while enabling seamless sensor operations within a limited power budget [12], [13].

This work directly tackles the energy management problem of EH environmental sensors, especially when deployed in harsh environments without network access. We present `DynaES` that accurately estimates the ESU's battery level (DC power gain) and dynamic energy scheduling based on sensor priority. We first develop a DC power gain estimator for `DynaES` to predict the available energy amount during the next operation period. With a thorough analysis of the National Solar Radiation Database (NSRDB) [14] and Photovoltaic Data Acquisition (PVDAQ) Public Datasets [15], we reveal that three parameters – global horizontal irradiance (GHI) [16], temperature, and solar zenith angle (SZA) [17] – are most critical to determine the DC power gain. The power gain estimator forecasts changes in these three parameters for the upcoming sensors' operation period (*e.g.*, one or two weeks) and utilizes a random forest model with the three estimated parameters to determine the final power gain. As this power gain estimator requires in-situ deployment with the various sensors, we carefully determine the best predictor for each parameter based on the accuracy, overhead, and power consumption in a computing board (Raspberry Pi). Specifically, we use a combination of lightweight deep learning and statistical time-series models for estimating these parameters.

Moreover, the scheduling component in DynaES performs dynamic power distribution based on each sensor's priority (significance). The sensing priority is determined based on domain-specific characteristics (*e.g.*, ideal sensing frequency for high data quality) and real-world metrics (*e.g.*, power drain). The scheduler then dynamically adjusts the intervals and frequency of sensing operations in order to ensure high energy efficiency and preserve sensing data quality.

We created a trace-based simulator to evaluate the effectiveness of DynaES, which utilizes input data created from two publicly available solar radiation datasets to estimate the DC power gain and simulate dynamic power scheduling for different EH sensors. To ensure a realistic simulation study, we collected data on the real power consumption of the DC power gain estimator and environmental sensors, including pH, electrical conductivity (EC), oxidation-reduction potential (ORP), and temperature sensors. We used this data as the simulation parameters for evaluating DynaES.

Our evaluation of DynaES focused on measuring the accuracy of the estimated DC power gain, monitoring changes in the charged battery level, and analyzing the frequency and intervals of sensing operations to determine whether DynaES can ensure sustainable sensor operations and data quality. According to our evaluation results, DynaES was able to accurately estimate the DC power gain with a RMSE of less than 100 within a week. We further compared the scheduling performance of DynaES with state-of-the-art baselines. The results revealed that DynaES enabled 1.8× to 4× more sensing operations, with shorter sensing intervals, without experiencing any outage caused by a complete battery drain.

The contributions of this work are as follows:

1. Accurate DC power gain prediction. The DC power gain estimator in DynaES can accurately predict ESU's DC power gain by combining lightweight predictors, which are carefully determined by their accuracy, overhead, and power consumption. This prediction mechanism enables solar-powered EH sensors to perform stable operations without external support from online weather forecasting services.

2. Dynamic energy scheduling algorithm. DynaES' scheduling algorithm shows effective utilization of harvested energy for sensors. This algorithm dynamically adjusts the intervals and frequency of sensing operations, ensuring high energy efficiency and preserving sensing data quality.

3. A thorough evaluation with a realistic simulation study. We perform a thorough simulation study to evaluate EH environmental sensors' operations. Our simulation study uses real-world public datasets as inputs, and the trustworthiness of the simulation is enhanced by incorporating actual measurement data of predictors and sensors. We evaluate various performances of DynaES, including power gain prediction, operation times, and sensing frequency compared to baselines.

The rest of this paper is organized as follows: Section II describes the detailed design of DynaES. Section III discusses our evaluations of DynaES, focused on its power gain prediction accuracy and energy scheduling efficiency. Section IV provides a discussion regarding this study. Section V describes related work. Finally, Section VI concludes this paper.

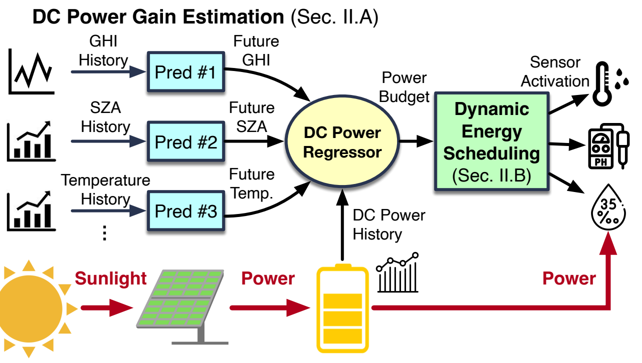


Fig. 2. The overall design of DynaES

II. The Design of DynaES

This section describes the detailed design of DynaES. Fig. 2 shows the overall design of DynaES that consists of two main components: ESU's DC power gain estimator (Section II-A) and dynamic energy scheduler (Section II-B) for various sensors. As stated in the introduction section, EH environmental sensors often have constraints, such as unreliable network access, which prevents them from obtaining timely weather information for future operational intervals. Consequently, DynaES should be capable of knowing the future power gain from solar energy. To achieve this, the **DC power gain estimator** of DynaES predicts future power gain by utilizing various parameters obtained by solar panels and equipped sensors in the deployed system.

After obtaining the predicted power gain, the **dynamic energy scheduler** in DynaES performs energy scheduling and distribution for various sensors based on their priority. The priority of each sensor is determined by its sensing objectives, with input from domain experts. The goal of the scheduler is to enable frequent operations of the more important sensors, thereby ensuring that the EH environmental sensors achieve their sensing goals efficiently while maintaining stable operations by avoiding a complete battery drain.

A. DC Power Gain Estimation

To design the DC power estimator, we start by investigating the relationship between solar/environmental parameters and DC power gain through solar panels. Unfortunately, there is no single dataset/DB containing solar energy-related factors and DC power gain. However, we found two datasets separately containing solar and environmental factors (NSRDB [14]) and DC power gain via photovoltaic solar energy (PVDAQ [15]).

Data preparation. NSRDB, maintained by the National Renewable Energy Laboratory (NREL), contains time-series data of meteorological, environmental, and solar irradiance from various locations across the U.S. The DB provides access

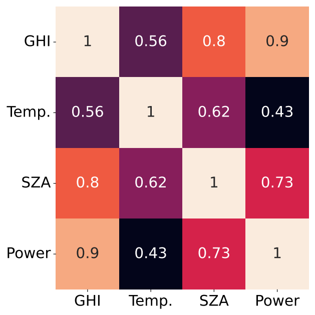


Fig. 3. Correlation coefficient of environmental/solar-irradiation parameters with DC power gain

to 19 critical environmental and solar-energy-related parameters, including temperature, wind speed/direction, SZA, GHI, and other essential information. While NSRDB has various time-series data regarding environmental parameters and solar irradiance, it lacks information connecting DC power gain leveraging solar irradiance and other factors. Therefore, we decided to merge NSRDB with PVDAQ DB.

PVDAQ is also collected and maintained by NREL. The datasets contain large-scale time-series data of PV (Photovoltaic) system metadata (*e.g.*, PV system models and dimensions of solar panels) and performance data (*e.g.*, DC power gain) from various experimental and public PV sites across the U.S. To merge these two separate datasets, we first found data chunks collected in the same locations and same measurement periods. One challenge in the data integration was that PVDAQ often had missing data points (data in specific months or days), so we had to focus on generating the longest continuous data from two DBs. Specifically, we could create integrated datasets from 2015 to 2018 containing all parameters in both DBs.

Estimating future changes in GHI and environmental parameters. With the integrated datasets, we performed correlation analysis to find the most significant environmental parameters to determine the DC power gain. Fig. 3 shows the correlation coefficient of various parameters (in NSRDB) with determining DC power gain (in PVDAQ), and we observed that GHI, temperature, and SZA are strongly or moderately correlated with DC power gain.

GHI refers to the amount of solar irradiance received by a solar panel on a horizontal surface [16]. GHI is directly linked to the amount of energy the solar panel can generate. Additionally, the efficiency of a solar panel in energy generation is directly impacted by ambient temperature. SZA is the angle between the sun and the zenith and can be an important factor in determining the amount of sunlight that a solar panel receives at different times of day and year [17]. When SZA is low, the sun is closer to the zenith, and the panel receives more direct sunlight, resulting in increased power generation. Conversely, when the SZA is high, the sun is farther from the zenith, and the panel receives less direct sunlight, resulting in reduced power generation. Therefore, these three parameters can be useful in predicting future DC gain. Please note that while we analyzed the coefficient of all 19 parameters, Fig. 3 only reports the three most important parameters.

TABLE I
ACCURACY, OVERHEAD, AND POWER CONSUMPTION MEASUREMENT RESULTS OF THREE PREDICTION MODELS ON RASBERRY PI. THE BOLD VALUES ARE THE BEST RESULTS WE OBTAIN IN THE MEASUREMENT.

	Accuracy	Overhead (Time in Sec.)		Power Consumption (mAh)	
	(RMSE)	Inference	Training	Inference	Training
HWES	150.8	**0.04**	–	**0.01**	–
NP	179.0	1.56	91.68	0.25	18.62
LSTM	**117.9**	6.45	138.41	0.99	25.52

To estimate the future DC power gain, it is critical to know the future changes of GHI, SZA, and temperature. Moreover, the estimation of future parameter changes needs to be performed on a computing board deployed together with EH sensors. Therefore, we should carefully decide to use proper predictors for these parameters based on their performance and power consumption (as the predictors should not consume a significant amount of power in the sensing system). For the predictor selection, we tested three time-series predictors from deep-learning and statistical learning approaches: long short-term memory (LSTM) [18], NeuralProphet (NP) [19], and Holt-Winters exponential smoothing (HWES) [20] methods.

LSTM is a type of recurrent neural network (RNN) designed for sequence or time-series predictions. In each inference task, LSTM generates both a prediction and hidden states. Specifically, the hidden states are passed back into the model for the next prediction during each inference. These hidden states store valuable information about the sequence of data, such as long-term and short-term trends in time series, which can assist in making accurate inferences [18]. LSTM can be a suitable choice for predicting future parameter changes since the seasonal period of each parameter could extend up to 3 months (the average length of a meteorological season). In such cases, the long-term memory of LSTM can be useful. Additionally, even within a specific season, environmental parameters can have fluctuations. Therefore, the short-term memory of LSTM can be beneficial in such cases.

NP is also designed for time-series predictions with supporting automatic detection of trends and seasonality in the given time-series dataset. NP is known to be an efficient approach in various time-series prediction problems [19]. And finally, HWES is a lightweight and widely used time series model focusing on capturing trends and seasonality in its forecasting. We chose HWES as it is lightweight (potentially consuming less power), and the parameters (*e.g.*, GHI and temperature) we want to predict can have strong seasonality.

To determine the proper predictor, we examined three candidates with GHI prediction, as GHI is the most important contributor to determining DC power gain. This test was performed on Rasberry Pi 4B, a widely used general-purpose computing board. We measured the prediction accuracy (RMSE), overhead in the training model and performing inference tasks, and actual power consumption. Specifically,

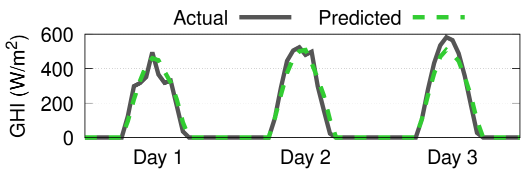
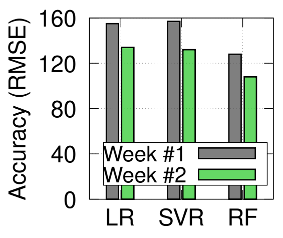
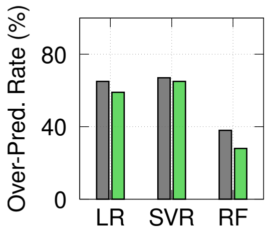


Fig. 4. 3-day GHI prediction results by LSTM (solid line: actual GHI values, dotted line: predicted GHI values)



(a) Overall Accuracy (RMSE)



(b) Over-prediction Rate (%)

Fig. 5. Accuracy evaluation results of three DC regressors

we used `INA-219`[1], a voltage, current, and power measurement chip, for measuring the power consumption of the models on the computing board.

The measurement results are reported in Table I. LSTM produced the most accurate prediction results but required higher computation time in training and inference than the other two models. LSTM's power consumption measurements on Raspberry PI were 0.99mAh (inference) and 25.52mAh (training). On the other hand, HWES showed a significantly lower overhead than the two other models, and its power consumption was also negligible. But its prediction accuracy was worse than LSTM (but better than NP).

As both LSTM and HWES models have strengths and weaknesses, we decided to use different predictors for estimating different parameters based on their significance. Specifically, LSTM is used for GHI predictions due to its strong correlation coefficient (0.9) with DC power gain. `DynaES` can leverage the accurate prediction results powered by LSTM. For example, Fig. 4 shows 3-day GHI prediction results by LSTM, and the figure shows that LSTM could accurately predict the GHI changes. Moreover, LSTM's power consumptions were only 0.99mAh (inference) and 25.52mAh (training), which can be easily accommodated by the typical ESU's capacity of 20,000 – 30,000mAh, given that training and inference will be performed periodically (*e.g.*, once a week). For the temperature and SZA, we decided to use HWES as it has reasonable prediction accuracy and negligible power consumption.

DC power gain regression. The next step is to design a DC power gain regressor that takes three predicted parameters from the previous step as inputs and estimates the future gains of DC power in ESU. To achieve more accurate prediction, this DC power regressor also leverages the history of DC power gains. To incorporate the predicted parameters and the DC power gain history, we considered various regressors, including linear regression (LR) [20], support vector regression (SVR) [20], and random forest (RF) [21] models.

LR is a commonly used approach for various regression problems due to its intuitive nature. We also considered the SVR because it can model both linear and non-linear relationships between the parameters and DC power via the kernel. Finally, we tested RF, which not only captures linear and non-linear relationships (like SVR) but is also known for its robustness to outliers and over-fitting. To compare the accuracy of these three regressors, we randomly selected two weeks of data from our dataset and assessed the performance of each regressor on this data.

Fig. 5 shows the accuracy evaluation results of three DC power regressors. Specifically, Fig. 5a reports the RMSE results of all the predictions. Overall, RF showed significantly higher accuracy than the other two regressors. For example, RF's RMSE results were 128 (week #1) and 108 (week #2), but the other two regressors had around a RMSE of 155 (week #1) and 133 (week #2).

Additionally, it is also important for the DC regressor in `DynaES` to have minimal over-prediction cases. Over-prediction occurs when the predicted DC power gain is higher than the actual DC power gain, which can lead to several scheduling issues. *i.e.*, overly allocating energy to sensors, resulting in more frequent sensing operations than future energy gain. Fig. 5b shows the over-prediction rates of the three regressors. Consistent with previous results, the RF-based regressor showed the lowest over-prediction rates, with 38% (week #1) and 28% (week #2), respectively. In contrast, LR had over-prediction rates of 65% and 59%, and SVR showed over-prediction rates of 67% and 65% for weeks #1 and #2, respectively.

Our further analysis revealed that the better accuracy of RF is mainly due to its robustness to outliers. For example, DC power gain prediction relies on various parameters with seasonality but can also be impacted by several intra-seasonal variations (*e.g.*, temperature changes within a specific season). RF is an ensemble model that trains each constituent tree on a random subset of the features, indicating that an outlier variable would not affect the prediction outcome of all the trees. Additionally, because the final prediction outcome is an average of the prediction outcomes of each tree, the outlier variable would have a reduced impact on the final prediction outcome in RF compared to other models.

With accuracy and over-prediction rate evaluations, we decided to use RF for our DC power regressor. The overall prediction procedure is illustrated in Fig. 6. In summary, the power gain estimation process in `DynaES` consists of two steps. The first step (step #1 in Fig. 6) takes past time-series data of GHI, temperature, and SZA and predicts future changes in these parameters. In this step, a more accurate LSTM is used to predict future GHI, while power-efficient HWES models are used to predict other parameters. The DC regressor (step #2 in Fig. 6) then utilizes RF to predict future (*e.g.*, 1-week)

[1]`https://www.ti.com/product/INA219`

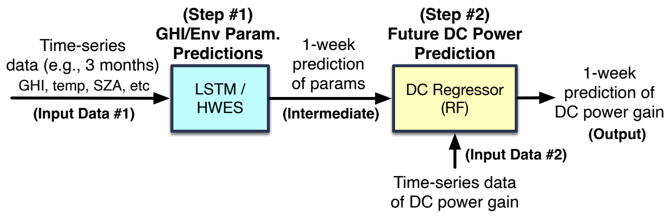

(Step #1)
GHI/Env Param. Predictions
Time-series data (e.g., 3 months)
GHI, temp, SZA, etc
(Input Data #1)
LSTM / HWES
1-week prediction of params
(Intermediate)
(Step #2)
Future DC Power Prediction
DC Regressor (RF)
1-week prediction of DC power gain
(Output)
(Input Data #2)
Time-series data of DC power gain

Fig. 6. Overall prediction procedure to estimate DC power gain

power gain by taking predicted parameters (step #1) along with the history of DC power gains. The output from the DC regression will be predicted power gain for the next week (hourly data for seven days, in total 24×7 predictions), which is the energy budget for the dynamic energy scheduler in the following subsection to facilitate further operations.

B. Dynamic Energy Scheduling

Given the prediction for future DC power gain (Section II-A), the dynamic energy scheduler in DynaES creates scheduling plans for each sensor. Our scheduling algorithm is described in Algorithm 1. The scheduler takes the following inputs: DC power gain prediction (E_{gain}), sensor profile (SP), battery (ESU) profile (B), and scheduling time window (T). Specifically, T refers to the duration for which we intend to schedule the power distribution to sensors in advance. And the scheduler's output is the scheduling plan for sensors during T.

The scheduler initiates the power distribution process by computing the total (E_{tot}) and accumulated energy (E_{accum}) for T (lines 1–5). T is a user-defined parameter (*e.g.*. one or two weeks). Following this, the scheduler evaluates whether E_{tot} is sufficient to meet the minimum energy requirement ($B[min\_energy\_required]$) for non-sensing activities, such as power consumed by computing boards or other equipment (line 6). If the energy is inadequate, then the EH sensor cannot carry out any sensing operations during T (line 7).

The scheduler then assigns energy distribution priority (E_{pr}) to each sensor ($s, \forall s \in SP$) based on each sensor's priority level ($s[priority]$) in line 9 and allocates energy for each sensor accordingly (E^s_{alloc}) as described in line 10. Subsequently, the scheduler calculates the frequency (line 11) and interval (line 12) for each sensor during T. Also, the shortest sensing period is calculated as the sensing time (line 12). For the sensors that have the shortest sensing period are selected to be scheduled (line 14). Next, the scheduler calculates the required energy (E_{req}) to execute all sensing operations (line 15) and searches the time that has enough energy in E_{accum} to determine the termination time (T_{end}) of sensing operations (lines 16-21). Finally, the scheduler generates the schedule plan, which includes the calculated sensing interval T_{end} and all sensors ($S^s_{min\_period}$) for the sensing operations.

Algorithm 1: DynaES Energy Scheduling Algorithm

```
Input: DC power (energy) gain prediction: $E_{gain}$,
       sensor profile: $SP$, battery profile: $B$,
       scheduling time window: $T$
Output: sensor schedule plan: $sched$
1  initialize hourly accumulated energy
     $E_{accum}[0] = E_{gain}[0] + B[current\_level]$
2  calculate total energy amount over $T$
     $E_{tot} = B[currrent\_level] + sum(E)$
3  for $t \in [1, T]$ do
4      $E_{accum}[t] = E_{accum}[t-1] + E_{gain}[t]$
5  end
6  if $E_{tot} \leq B[min\_energy\_required]$ then
7      sleep($T$)
8  end
9  calculate energy priority ratio
     $E_{pr} = E_{tot} \div \sum_{s \in SP} s[priority]$
10 calculate energy allocation to each sensor
     $E^s_{alloc} = E_{pr} \times s[priority]$
11 calculate number of sensing operations for each sensor
     $S^s_{ops} = E^s_{alloc} \div s[consumption]$
12 calculate the sensing period for each sensor
     $S^s_{period} = T \div S^s_{ops}$
13 choose the shortest sensing period
     $T\_min = min(S^s_{period})$
14 choose sensors with the shortest sensing period
     $S^s_{min\_period} = \{s : S^s_{period} == T\_min \ \forall s \in SP\}$
15 calculate the energy required for the schedule
     $E_{req} = B[base\_consumption] +$
     $\sum_{s \in SP} S^s_{min\_period} \times s[consumption]$
16 for $t \in [T\_min, T]$ do
17     search for the sufficient energy time
18     if $E_{accum}[t] > E_{req}$ then
19         $T_{end} = t$
20     end
21 end
22 return $[S^s_{min\_period}, T_{end}]$
```

III. Evaluation of DynaES

Our evaluation aims to assess the performance of the DC power estimator and energy scheduling in DynaES. The accuracy of the DC power estimator is critical for DynaES as it predicts future energy gains for scheduling operations. We measured the accuracy (RMSE) and over-prediction rate of the DC power estimator, as well as the impact of various parameters on producing accurate prediction results.

Additionally, the performance evaluations of the dynamic energy scheduler were based on simulations with realistic scenarios of EH environmental sensors. Specifically, we measured changes in energy charging level in ESU, sensing operation frequency, and sensing intervals in DynaES with various sensors. Furthermore, we compared the performance of our dynamic energy scheduling against three baselines: static, adaptive energy scheduler, and AsTAR.

Evaluation dataset. As discussed in Section II-A, we merged two publicly available datasets collected by NREL to create new datasets. For the performance evaluation, we prepared datasets for a two-year period (2017-2018). The datasets include GHI, temperature, SZA parameters, and DC power gain in the Denver area, CO. It is worth noting that the dataset used to design DynaES was not used for its performance evaluation to ensure a fair evaluation.

Simulator design and implementation. We developed a trace-based simulator modeled on the behavior of environmental sensors in EH systems. The simulator has four main components: a timer, an energy generator, a battery, and a scheduler. The timer component is responsible for controlling the simulation time. The energy generator takes a trace file containing time-series data of GHI and other environmental parameters as input and generates corresponding values for GHI, temperature, and SZA at predefined simulation times. The battery component mimics the behavior of ESUs in EH sensors. This component simulates power charging and discharging functions and generates a trace of the DC power history, which is used by the scheduler component. The battery leak function is also implemented to simulate the real-world behavior of batteries. The scheduler is the core unit of the simulator and performs critical operations. This component predicts DC power gain, schedules energy delivery to sensors, traces all sensor operations and coordinates other system functions. This component ensures that the simulator accurately mimics the behavior of EH systems in real-world scenarios. This simulator is written in Python. Pandas[2] [22] and Numpy[3] [23] libraries are used to process the time-series data. Scikit-learn[4] [24] and TensorFlow[5] [25] are used to implement RF and LSTM models.

A. Accuracy of DC Power Gain Estimation

Dataset for power gain estimation. We used 12 weeks of data spanning from September 2017 to August 2018. Specifically, we tested the estimation accuracy of the first week of every month over a one-year period, resulting in a total of 12 prediction tests. This allowed us to determine if the DC power gain estimator consistently yields high prediction accuracy.

Power gain estimators. While we previously revealed that GHI is the most important parameter for predicting future DC power gain (in Section II-A), we also wanted to evaluate the contribution of other parameters, such as temperature, and SZA. To this end, we created three different versions of the DC power gain estimator, which are DynaES' DC power estimator, a DC power estimator utilizing GHI and DC power history, and a power estimator using only DC power history.

Evaluation results. Fig. 7 shows the accuracy results of the three power estimators. Specifically, as shown in Fig. 7a, DynaES' power estimator outperformed the other two models and consistently produced prediction results with low prediction errors. On average, DynaES' estimator had a RMSE of 96. The second estimator (Est.(GHI + DC) in the figure), which utilizes GHI and DC power history, had a RMSE of 143, which is 50% higher than DynaES. As this estimator does not take into account the other three environmental parameters, the results indicate that both temperature and SZA parameters significantly contribute to reducing prediction errors. The third estimator (Est.(DC Only) in the figure), which only utilizes DC power gain history, had the highest prediction errors with a RMSE of 190, significantly higher than DynaES. Moreover, the accuracy of the third estimator was much more variable than DynaES and the second estimator. This finding highlights the importance of GHI and other parameters in improving the prediction accuracy of future DC power gain.

Fig. 7b reports the over-prediction rates of all three power estimators. Over the one-year evaluation period, DynaES had an over-prediction rate of 14% – 35%, with an average over-prediction rate of 25%. DynaES consistently generated lower over-prediction rates (32% – 47% lower) than the other two estimators, which can lead to more stable management in energy scheduling to target sensors.

B. Energy Scheduling Evaluation

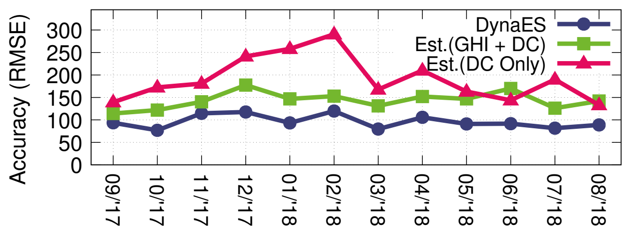
This evaluation is to test the effectiveness of the energy scheduling algorithm in DynaES. Our evaluation involves monitoring changes in the charged energy levels of the ESU, as well as adjustments in the sensing frequency and intervals. Specifically, monitoring charged energy-level changes in the ESU is to assess whether the EH sensors can operate efficiently without a complete power drain. Furthermore, changes in sensing frequency and intervals are also essential metrics, as the EH sensors should not compromise their performance simply to conserve energy for longer operation hours. To ensure efficient functionality, the energy scheduling algorithm in DynaES should be capable of maintaining the frequency and interval for each sensor based on its priority while considering the dynamics of the charged energy level.

Sensor and ESU models in simulation. We consider the deployment of an EH system with four sensors, including temperature, pH, EC, and ORP sensors. We assume that temperature and pH sensors have higher sensing priority (more frequent sensing operation, *e.g.*, two operations per hour) and EC and ORP sensors have lower priority (*e.g.*, one operation per hour). For realistic simulation, we profiled the power consumption of all the sensors using the same approach used in Section II-A and used the profiled data as simulation parameters. For the ESU models, we modeled ESU components in our simulator based on battery packs with 20,000 and 30,000mAh, commonly available on the market.

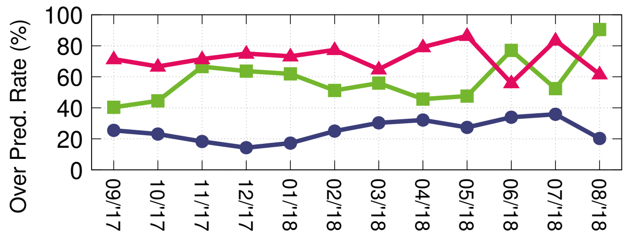
Dataset for scheduler evaluation. We randomly picked five individual weeks in our dataset. Three weeks are selected from 2017, and two weeks are chosen from the 2018 dataset.

Baselines. Static energy scheduler (ES-stat), adaptive energy scheduler (ES-adap), and AsTAR are used as baselines. ES-stat performs sensing operations based on a predetermined sensing interval. As ES-stat does not consider

[2]https://pandas.pydata.org
[3]https://numpy.org
[4]https://scikit-learn.org
[5]https://www.tensorflow.org

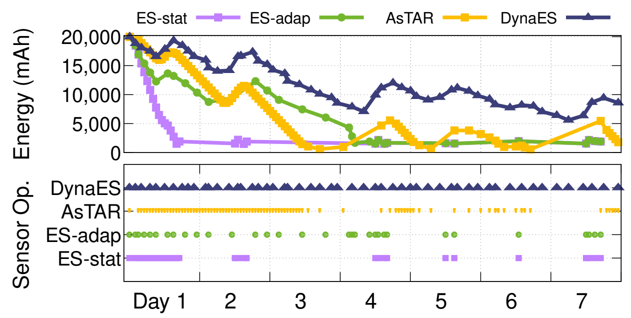


(a) Overall Accuracy (RMSE)

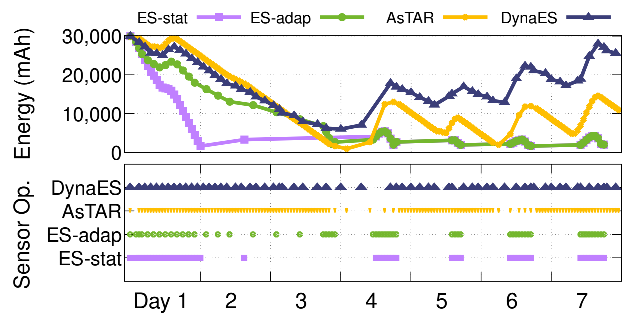


(b) Over-prediction Rate (%)

Fig. 7. Accuracy evaluation results of three DC power gain estimators. (a) shows RMSE results for 12 weeks (the first week of every month) from September 2017 to August 2018. (b) shows the over-prediction ratio of three estimators. Please note that Est.(GHI + DC) in the figure indicates the DC power estimator using GHI and DC power history, and Est.(DC Only) is the estimator only using DC power history.



(a) 4th Week in December (2017)



(b) 2nd Week in November (2017)

Fig. 8. Evaluation results of energy scheduling performed by `DynaES`, `ES-stat`, `ES-adap`, and `AsTAR`. (a) represents the evaluation results with a 20,000mAh of battery capacity in the 4th week of December 2017, and (b) shows the results with a 30,000mAh of battery capacity in the 2nd week of November 2017. The upper graphs show the battery level changes (charged by solar panel and the battery consumption by sensing operations) during evaluation weeks. The lower graphs represent the sensing operation frequencies.

the changes in the ESU's energy level or other environmental dynamics, `ES-stat` is to show the baseline performance of the energy scheduler in EH sensors.

`ES-adap` dynamically adjusts sensor operation frequencies based on the current energy level in the ESU. `ES-adap` starts scheduling by checking if the current energy level exceeds a threshold set to decrease the chance of complete battery drain. If the current energy level is higher than the threshold, `ES-adap` enables sensing operations. After each sensor operation, `ES-adap` updates the energy level by calculating the energy gain and drain from the sensors. It then determines the next sensing interval by calculating $\alpha \times \frac{battery\_level - threshold}{threshold}$, where α is a scaling factor used to regulate the energy-capacity scaling. In this way, `ES-adap` adjusts the sensing interval based on the current energy level. Thus, when the battery level is higher, `ES-adap` schedules more frequent sensing operations, and when the battery level is lower, it schedules fewer sensing operations.

`AsTAR` [26] is an energy scheduler focused on performing energy-aware adaptation for sensor tasks. It aims to balance the trade-off between higher temporal resolution of sensed data and increased energy consumption. `AsTAR` is designed to operate without benchmarking or environmental modeling, instead observing and reacting to the system state at runtime. This energy scheduler is also designed to address platform heterogeneity and unpredictable energy-harvesting environments.

Evaluation results. We performed energy scheduling in the five weeks in 2017 and 2018 and measured energy level changes in the ESU and sensing frequency (interval) in the five weeks. Fig. 8 reports the energy scheduling results in two weeks. Please note that we only show two weeks' results due to the page limit, and the other three weeks had similar results. As shown in the upper graphs of Fig. 8 (the changes in battery level), `DynaES` demonstrated its ability to effectively manage battery levels and enable more frequent sensing operations without fully depleting the battery. Although there were fluctuations in the charged battery level, these fluctuations were primarily due to weather dynamics. Conversely, the other three baselines could not support sustainable operations of EH sensors. For example, `ES-stat` experienced a complete battery drain on day 1 of both weeks and thereafter, `ES-stat` only conducted very limited sensing operations. `ES-adap` performed better than `ES-stat`. However, `ES-adap` and `AsTAR` also encountered a complete battery drain on days 3 and 4 of Fig. 8a and day 3 of Fig. 8b, leading to limited support for future operations due to the shortage of charged energy.

The lower graphs of Fig. 8 further illustrate the sensing operation frequencies performed by `DynaES` and three baselines. The results show that `DynaES` enabled $1.8\times$ – $4\times$ more frequent sensing operations, indicating that `DynaES` does not

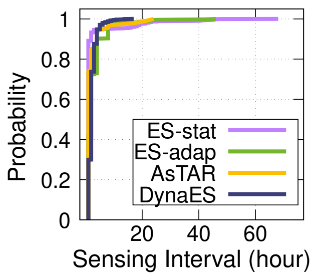


Fig. 9. CDF of sensing intervals in four schedulers

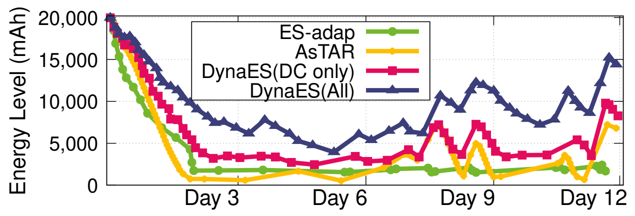


Fig. 10. Changes in energy level by four schedulers. DynaES (all) refers to DynaES with DC power gain prediction using GHI, SZA, temperature, and DC power history. DynaES (DC only) only uses DC power gain estimation using DC power history. All the dots on the lines indicate sensing operations.

sacrifice the frequency of its sensing operations to conserve more energy. By conducting more sensing operations, the EH sensors equipped with DynaES can collect more data from its sensors, thereby facilitating the achievement of the sensors' objectives. Moreover, we observed that DynaES could dynamically adapt the sensing interval based on the charged battery level. *i.e.*, day 4 of Fig. 8b's lower graph. DynaES extended the sensing interval to prevent the complete battery runout because the less energy gain on day 4 was predicted from the DC power gain estimator.

Fig. 9 shows the CDF of the sensing interval for the four schedulers. The results show that the sensing intervals for DynaES were consistently smaller compared to the three baselines. Conversely, the three baselines exhibited long-tail distributions of sensing intervals, indicating their inability to support consistent sensing operations. By having shorter sensing intervals (without long-tail distribution) in Fig. 9 and higher sensing frequency in Fig. 8, DynaES can support more frequent and consistent sensing operations, leading to improving the sensing quality of EH sensors.

In this evaluation, we showed that DynaES outperformed the three baselines. The better performance of DynaES is primarily due to its accurate prediction of future power gain using various parameters and its scheduling algorithm, which can dynamically adjust scheduling intervals to manage the battery level, thereby supporting sustainable overall runtime of EH sensors with increased frequency of sensing operations.

IV. Discussion

This study presents DynaES, an energy scheduler for EH sensors without reliable network access to online weather forecasting APIs. DynaES uses a DC power gain estimator to predict future solar panel power gain for ensuring sustainable sensor operations. However, to utilize DynaES, EH sensors should be capable of collecting GHI, temperature, and SZA parameters. This section discusses two key questions for adopting DynaES in real-world deployments: 1) how to obtain GHI and other parameters for EH sensors, and 2) DynaES's performance when GHI and other parameters are not available.

A. Question 1: How to collect GHI, SZA, and temperature parameters in in-situ EH sensors.

A pyranometer can collect GHI parameters [16]. A typical pyranometer setup includes a light sensor that filters out non-sunlight waves. This light is then converted into thermal energy, which is subsequently transformed into electricity using a thermoelectric device, such as a thermopile.

In a practical deployment of EH sensors, a pyranometer can be installed near or alongside the EH sensors to collect GHI data. This information can then be utilized in our DC power gain estimator in DynaES. Furthermore, the pyranometer can help to enhance the prediction accuracy of our power gain estimator by comparing the predicted GHI parameters against the pyranometer's actual measurements.

In addition, SZA parameters [17] can be calculated using equations that utilize the current time and location coordinates (longitude and latitude). Temperature parameters can be obtained using low-cost sensors widely available on the market.

B. Question 2: The performance DynaES without GHI, SZA, and temperature parameters.

Given the critical role of GHI along with SZA and temperature parameters in estimating future DC power gain, we seek to determine whether DynaES can still effectively schedule energy to sensors even without GHI, SZA, and temperature parameters, as these sensors (*e.g.*, pyranometer) may not always be available in real-world deployment scenarios. In this discussion, we evaluate the performance of DynaES powered solely by DC power gain prediction with historical DC power information, which can be easily obtained by APIs supported by commercial battery packs.

Fig. 10 illustrates the changes in energy levels of four schedulers; DynaES(All), DynaES(DC only), AsTAR (baseline), and ES-adap (baseline). DynaES(All) employs all parameters for DC power gain prediction, while DynaES(DC only) only uses DC power history for energy-gain prediction. The results show that DynaES(DC only) was not able to maintain the same energy level as DynaES(All) due to lower accuracy of future energy gain prediction. However, it showed higher efficiency than the baseline by managing the battery level and performing more sensing operations (dots on lines). Notably, DynaES(DC only) was observed to perform a similar frequency of sensing operations with DynaES(All).

Additionally, we measured the sensing intervals performed by the schedulers. Fig. 11 reports the CDF of the sensing interval. The results show that DynaES(DC only) had similar sensing intervals to DynaES(All), with only the lower 20% of sensing intervals being slightly longer than those

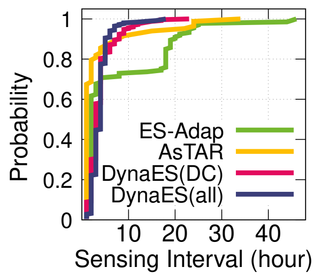


Fig. 11. CDF of sensing intervals of `DynaES(All)`, `DynaES(DC only)`, and `ES-adap` (baseline)

of `DynaES(All)`. Specifically, we did not observe a long-tail distribution of sensing intervals in `DynaES(DC only)`, suggesting that it can support sufficiently frequent sensing operations despite relatively inaccurate DC power gain estimations.

V. Related Work

A. Weather forecasting for EH sensors.

Exponentially weighted moving average (EWMA) is a commonly used approach for weather prediction in EH sensors due to its lightweight nature [27], [28]. However, in Section II-A, HWES (a type of EWMA) was shown to have lower accuracy compared to more advanced LSTM models. While we used HWES, it is only for less significant parameters (*e.g.*, temperature). Guermoui *et al.* [29] utilized SVR to predict solar irradiance and demonstrated higher accuracy than a prediction model based on Multi-layer Perceptron. However, this approach did not consider other important parameters for predicting solar irradiance. Jalali *et al.* [30] used a variation of LSTM to predict GHI with historical data from weather stations, while Capizzi *et al.* [31] employed a different variation of RNN to predict GHI by incorporating other weather parameters. However, both approaches only provided short-term predictions of GHI (*e.g.*, 2 days). In contrast, `DynaES` can produce GHI predictions for a much longer period (at least weeks). Moreover, our prediction goal is not only to predict GHI but also eventually estimate DC power gain by incorporating other important parameters.

Sharma *et al.* [32] developed an approach to predict the energy harvested from natural sources (*e.g.*, solar and wind) by combining forecast data (cloud coverage) from online weather forecasts [9] with data collected from weather stations. While their approach shares a similar goal with the DC power gain estimator in `DynaES`, we did not consider cloud coverage from weather forecasts for power gain prediction. Furthermore, the DC power gain estimator in `DynaES` focuses on capturing the seasonality in various environmental parameters. Importantly, `DynaES` can be utilized for EH sensors without network access to obtain online weather forecasting information.

B. Sensor scheduling in EH sensors.

Limited energy is a major challenge for EH sensors, and several schedulers have been proposed to tackle this problem. Kansal *et al.* [28] designed a greedy algorithm that adjusts sensing intervals according to sensing frequency requirements, but their approach did not consider the power consumption of each sensor. Caruso *et al.* [33] and Loreti *et al.* [34] addressed scheduling using optimization techniques. Yang *et al.* [26] also developed the AsTAR task scheduler that adjusted sensing intervals based on available energy. However, all these approaches did not consider each sensor's priority. In contrast, `DynaES` schedules sensors with different priorities by giving high-priority sensors more energy and scheduling priority to achieve their sensing goals, unlike most existing approaches that use a fixed sensor priority. Moreover, in our evaluation, we compared `DynaES` against AsTAR, and `DynaES` significantly outperformed AsTAR by performing more frequent sensing operations and having longer operation hours.

VI. Conclusion

This work presents `DynaES`, a new energy scheduler for solar-powered EH environmental sensors, which are deployed in environments without access to weather information networks. We designed `DynaES` with two components: a DC power gain estimator and a dynamic energy scheduler. For the DC power gain estimator, we first merged two datasets and then used deep learning and statistical time-series approaches to predict changes in GHI, SZA, and temperature parameters to estimate future DC power gain. `DynaES` schedules energy distributions to each sensor based on priority, dynamically adjusts sensing intervals/frequency and ensures energy efficiency and data quality. Evaluation results showed that `DynaES` accurately predicts future energy gain and effectively schedules energy to support more frequent operations and longer battery life. In the near future, we plan to deploy solar-powered EH sensors with `DynaES` and evaluate the efficacy and performance of `DynaES` in a real-world deployment.

Acknowledgment

The authors would like to thank the anonymous reviewers for their insightful comments and constructive suggestions. This research was in part supported by the U.S. Department of Agriculture (USDA), under award number 2021-67019-34342. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the view of the USDA.

References

[1] Ali Saffari, Vikram Iyer, Zerina Kapetanovic, and Vaishnavi Ranganathan. Smart Pallets: Toward Self-Powered Pallet-Level Environmental Sensors for Food Supply Chains. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2022.

[2] Long Liu et al. Promoting Smart Cities Into The 5G Era With Multi-field Internet of Things (IoT) Applications Powered With Advanced Mechanical Energy Harvesters. *Nano Energy*, 88, 2021.

[3] Madeleine I. G. Daepp and et al. Eclipse: An End-to-End Platform for Low-Cost, Hyperlocal Environmental Sensing in Cities. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2022.

[4] Lea Dujic Rodic, Tomislav Zupanovic, Toni Perkovic, Petar Solic, and Joel J. P. C. Rodrigues. Machine Learning and Soil Humidity Sensing: Signal Strength Approach. *ACM Transactions on Internet Technology*, 22(2):39:1–39:21, 2022.

[5] Ju Wang, Liqiong Chang, Shourya Aggarwal, Omid Abari, and Srinivasan Keshav. Soil moisture sensing with commodity RFID systems. In *ACM International Conference on Mobile Systems Applications, and Services (MobiSys)*, 2020.

[6] Andy Rosales Elias, Nevena Golubovic, Chandra Krintz, and Rich Wolski. Where's The Bear?: Automating Wildlife Image Processing Using IoT and Edge Cloud Systems. In *IEEE/ACM International Conf. on Internet-of-Things Design and Implementation (IoTDI)*, 2017.

[7] Naomi Stricker, Ying Zhao Lian, Yuning Jiang, Colin N. Jones, and Lothar Thiele. Joint Energy Management for Distributed Energy Harvesting Systems. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2021.

[8] Kai Geissdoerfer, Raja Jurdak, Brano Kusy, and Marco Zimmerling. Getting more out of energy-harvesting systems: energy management under time-varying utility with preact. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2019.

[9] NOAA. National Weather Services, 2023. `www.weather.gov`.

[10] OpenWeather, 2023. `openweathermap.org`.

[11] Xiaojie Wang, Zhaolong Ning, Xiping Hu, Lei Wang, Lei Guo, Bin Hu, and Xinyu Wu. Future Communications and Energy Management in the Internet of Vehicles: Toward Intelligent Energy-Harvesting. *IEEE Wireless Communications*, 26(6):87–93, 2019.

[12] Deniz Gunduz, Kostas Stamatiou, Nicolo Michelusi, and Michele Zorzi. Designing intelligent energy harvesting communication systems. *IEEE communications magazine*, 52(1):210–216, 2014.

[13] Vikrant Bhatnagar and Philip Owende. Energy Harvesting for Assistive and Mobile Applications. *Energy Science & Eng.*, 3(3):153–173, 2015.

[14] National Renewable Energy Laboratory. NSRDB: National Solar Radiation Database, 2023. `nsrdb.nrel.gov`.

[15] Chris Deline and et al. Photovoltaic Data Acquisition Public Datasets, 2021. `www.osti.gov/dataexplorer/biblio/dataset/1846021`.

[16] P.I. Raptis, S. Kazadzis, B. Psiloglou, N. Kouremeti, P. Kosmopoulos, and A. Kazantzidis. Measurements and model simulations of solar radiation at tilted planes, towards the maximization of energy capture. *Energy*, 130:570–580, 2017.

[17] Mark Z. Jacobson. Fundamentals of Atmospheric Modeling. 2005.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[19] Oskar Triebe, Hansika Hewamalage, Polina Pilyugina, Nikolay Laptev, Christoph Bergmeir, and Ram Rajagopal. NeuralProphet: Explainable Forecasting at Scale. *CoRR*, abs/2111.15397, 2021.

[20] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The Element of Statistical Learning: Data Mining, Inference, and Prediction. 2011.

[21] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[22] Wes McKinney. Data Structures for Statistical Computing in Python. In *9th Python in Science Conference (SciPy)*, 2010.

[23] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[24] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[25] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A System for Large-Scale Machine Learning. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016.

[26] Fan Yang, Ashok Samraj Thangarajan, Gowri Sankar Ramachandran, Wouter Joosen, and Danny Hughes. AsTAR: Sustainable Energy Harvesting for the Internet of Things through Adaptive Task Scheduling. *ACM Transactions on Sensor Networks*, 18(1), 2021.

[27] Alessandro Cammarano, Chiara Petrioli, and Dora Spenza. Pro-Energy: A novel energy prediction model for solar and wind energy-harvesting wireless sensor networks. In *IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, 2012.

[28] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B Srivastava. Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(4):32–es, 2007.

[29] Mawloud Guermoui, Abdelaziz Rabehi, Kacem Gairaa, and Said Benkaciali. Support vector regression methodology for estimating global solar radiation in algeria. *The European Physical Journal Plus*, 133:1–9, 2018.

[30] Seyed Mohammad Jafar Jalali, Sajad Ahmadian, Abdollah Kavousi-Fard, Abbas Khosravi, and Saeid Nahavandi. Automated Deep CNN-LSTM Architecture Design for Solar Irradiance Forecasting. *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, 52(1):54–65, 2022.

[31] Giacomo Capizzi, Christian Napoli, and Francesco Bonanno. Innovative Second-Generation Wavelets Construction With Recurrent Neural Networks for Solar Radiation Forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 23(11):1805–1815, 2012.

[32] Navin Sharma, Jeremy Gummeson, David Irwin, and Prashant Shenoy. Cloudy Computing: Leveraging Weather Forecasts in Energy Harvesting Sensor Systems. In *IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2010.

[33] Antonio Caruso, Stefano Chessa, Soledad Escolar, Xavier del Toro, and Juan Carlos López. A Dynamic Programming Algorithm for High-Level Task Scheduling in Energy Harvesting IoT. *IEEE Internet of Things Journal*, 5:2234–2248, 2018.

[34] Pierpaolo Loreti, Lorenzo Bracciale, and Giuseppe Bianchi. StableSENS: Sampling time decision algorithm for IoT energy harvesting devices. *IEEE Internet of Things Journal*, 6(6):9908–9918, 2019.