Solar-Based Timekeeping for Batteryless Devices

Konstantin Gosch, Johannes Göpfert, Bernd-Christian Renner
Hamburg University of Technology, Germany
Email: {konstantin.gosch, johannes.goepfert, christian.renner}@tuhh.de

***Abstract*—The Internet of Things (IoT) is steadily gaining traction, but its reliance on battery power raises significant challenges. To address this, researchers are developing energy-harvesting IoT devices that operate on intermittent power, eliminating the need for batteries but complicating accurate timekeeping due to unreliable energy supply. Traditional Real-Time Clocks (RTCs) and synchronization methods are ineffective under such conditions, hindering tasks that require precise timing. We propose timekeeping using TinyML to predict sunrise and sunset based on power usage patterns, enabling autonomous time inference without continuous power or external synchronization. We trained and evaluated multiple lightweight models through simulation and with real hardware, and we demonstrate their potential and shortcomings compared to conventional methods.**

***Index Terms*—Batteryless systems, timekeeping, time synchronization, Internet of Things (IoT)**

I. Introduction

Batteryless devices are becoming increasingly prevalent in the Internet of Things (IoT) landscape, particularly in remote and isolated environments where traditional power sources, like batteries, are impractical or impossible to deploy. Their promise for autonomous longevity and reduced maintenance costs/effort, makes them an attractive option for various applications, such as environmental monitoring, smart agriculture, and industrial automation [1], [2]. Batteryless devices rely on energy harvesting, such as solar or vibration energy harvesting, to power their operations. Usually energy intake is not sufficient to maintain continuous operation, leading to frequent power interruptions causing an intermittent system behaviour.

The intermittent nature of these systems poses significant challenges such as timekeeping as the device loses its notion of time during power outages. Traditional timekeeping methods, such as Real-Time Clocks (RTCs) or synchronization protocols, are not suitable for these systems due to their reliance on a continuous power supply or energy intensive communication with external time sources. These constraints make it difficult to deploy batteryless devices in scenarios where timekeeping is essential, such as in scheduling tasks, recording time series or communication with other devices.

The problem of timekeeping for intermittent systems has been addressed already in various ways. The usual approach is to use methods similar to hourglasses that can be used to keep track of time during power outages. One approach is to use capacitors to store energy and measure time based on the discharge rate, as proposed by [3]. A disadvantage of this method is that it requires additional hardware and depending on the time range a large capacitor, which is not practical for most batteryless devices. Another method is shown in [4], which exploits timely staggered bitflippling in the memory of a microcontroller to measure the passage of time. The method's time range is highly dependent on the specific hardware used and requires careful calibration to ensure accuracy. While these methods provide a power interruption resistant timekeeping mechanism, their exponential clock drifts over time makes them unsuitable for long-term deployments. Additionally, if not starting from a known time, they cannot provide an absolute time reference, which would be useful for rendezvous with other devices. Another approach is to use external synchronization sources, such as GNSS or cellular networks, to periodically update the device's time. However, these methods are not always feasible in remote or isolated deployments where network connectivity is limited or non-existent. Beside the need for additional hardware, these methods also require a usually significant amount of energy to maintain the connection and synchronize the time, which is not suitable for batteryless devices.

In contrast to the existing solutions, we propose an approach that enables batteryless devices to autonomously infer the current time based on their energy harvesting patterns provided by a solar cell exposed to outdoor sunlight. Taking up the idea of a sun dial, we leverage the predictable nature of solar energy availability to create a timekeeping mechanism that does not rely on continuous power or external synchronization sources. By leveraging TinyML—machine learning designed for resource-constrained devices—we try to predict sunrise and sunset using them as known time anchors to estimate the current time. This makes absolute timekeeping viable even in remote and isolated deployments, where network connectivity or traditional timing hardware is unavailable.

To validate this approach, multiple lightweight machine learning models are trained and evaluated for their ability to predict sunrise and sunset events. Their performance is assessed in terms of classification accuracy as well as energy efficiency when deployed on an intermittent node. The results demonstrate that several models can accurately estimate temporal anchors while consuming up to three orders of magnitude less energy than conventional timekeeping methods in the best case scenario.

In summary, we make the following key contributions:
(i) We propose a novel timekeeping approach for intermittent systems that leverages TinyML to synchronize on sunrise and sunset.
(ii) We evaluate the accuracy and energy efficiency of multiple

Published in the proceedings of the 22nd GI/ITG KuVS Fachgespräch Sensornetze (FGSN), 2025. https://doi.org/10.21268/20250822-1

TABLE I: Simulated state data used for training and evaluation. The table shows the node's state at specific timestamps, with the duration of the previous state.

Time	State	Duration	Label
2024-03-04 07:07:23.592	*init*	0 days 00:00:00	NA
2024-03-04 07:07:24.292	*active*	0 days 00:00:00.700	*Neither*
2024-03-04 07:07:36.017	*sleep*	0 days 00:00:11.725	*Neither*
2024-03-04 07:21:06.196	*active*	0 days 00:13:30.179	*Sunrise*
2024-03-04 07:21:25.710	*sleep*	0 days 00:00:19.514	*Sunrise*

machine learning models tailored for this prediction task.

II. Methodology

The general idea of our approach is to detect sunrise and sunset depending on the harvesting power that is available to the node and with this detection to estimate the time of day. Instead of measuring the input power directly, we use the node's state changes as an indirect measure. We therefore run the node in soft-IC configuration, meaning that instead of running the device until it runs out of energy, it goes to a deep sleep mode and keeps a timer running while sleeping. The node's state changes are then affected by the amount of harvested energy, which causes transitions between active and sleep state. The durations between these transitions are then used to train machine learning models that predict sunrise and sunset based on the node's current and previous states.

Figure 1a illustrates the harvesting power at the beginning of a day, while Figure 1b depicts the resulting state transitions driven by this harvested energy. To generate the transition data we use recorded power traces and simulate the node's state transitions with the simulator presented in [5].

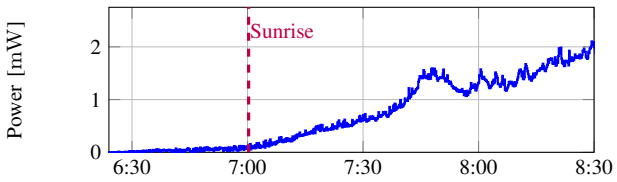
Table I presents the node's simulated state at specific timestamps along with the duration of the previous state, which is used to build the feature sets for training. We train our models to classify either *Sunrise*, *Sunset*, or *Neither* based on the state durations. For the labelling process, the actual sunrise and sunset times were obtained using the Python library *astral*. Since labelling only the exact event time for sunrise and sunset would yield to few features for these classes, sunrise and sunset points were expanded to windows and shifted via adjustable parameters, facilitating classification of these classes.

Figure 1b shows how the labelling windows mark data points as *Sunrise* or *Neither*. The parameters $\text{time\_offset}_{\text{rise}}$ and $\text{time\_offset}_{\text{set}}$ shift the windows, while $\text{time\_window}_{\text{rise}}$ and $\text{time\_window}_{\text{set}}$ set their durations. Although offsets can be corrected during time estimation, the uncertainty from window expansion introduces unavoidable ambiguity in label precision, expressed as:

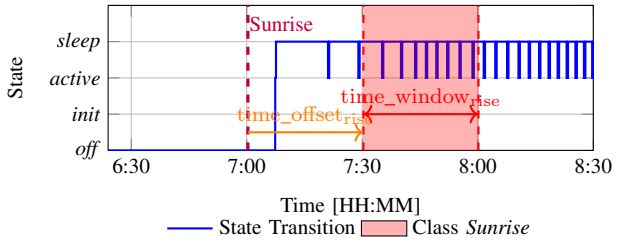
$$t_{\text{current}} = (\overline{\text{sunrise}} + \text{time\_offset}_{\text{rise}})_{-0}^{+\text{time\_window}_{\text{rise}}} \quad (1)$$

where t_{current} is the node's estimated time, $\overline{\text{sunrise}}$ the calculated sunrise, and the superscript/subscript denote uncertainty bounds. Similarly, for sunset:

$$t = (\overline{\text{sunset}} - \text{time\_offset}_{\text{set}})_{-\text{time\_window}_{\text{set}}}^{+0} \quad (2)$$



(a) Harvested solar power on April 3 in Hamburg, Germany.



(b) Simulated state transitions for an intermittent sensor node. We define a window with offset to label the data points as *Sunrise* or *Sunset*

Fig. 1: Overview of solar power and simulated sensor node transitions.

A. Features

The exact feature set used for training the models is crucial for their performance. We use the last 10 *active* and the last 10 sleep durations for our feature set and additionally five statistical features—median, maximum, minimum, mean gradient, and absolute mean gradient—which are computed separately for each state type, resulting in 30 features. After power loss, duration arrays are zero-initialized. On each state change, the current duration, measured by the microcontroller's timer, is added to the front of the corresponding array, dropping the oldest value to keep a fixed size.

B. Model Selection

To balance accuracy, computational efficiency, and deployment feasibility, four machine learning models were selected:

a) XGBoost: Chosen for its high accuracy and efficient gradient-boosted trees, XGBoost scales well and suits resource-constrained nodes by balancing computational cost and performance [6].

b) Random Forest: Selected for robustness and consistent performance across datasets, Random Forest offers stable, interpretable results with moderate memory and runtime needs, making it popular in similar tasks [7].

c) Naive Bayes: Included for very fast training and inference, beneficial in real-time or low-power settings, though its conditional independence assumption may not hold for this data, as noted in evaluation [8].

d) Neural Networks: Considered for their flexibility in modelling complex relationships. Despite offline training, their higher resource demands pose deployment challenges on constrained devices; however, their accuracy warranted evaluation [9].

III. Evaluation

For an accurate time estimation it is important to correctly classify the sunset and sunrise. We therefore evaluate the trained models based on their **accuracy**.

Accuracy measures how well a model classifies the classes *Sunrise*, *Sunset*, or *Neither*. To reflect the goal of identifying temporal reference points, we introduce two custom metrics, $days_{\text{rise}}$ and $days_{\text{set}}$, which denote the proportion of days in the test set where at least one *Sunrise* or *Sunset* is detected correctly. This is necessary due to the fact that multiple feature sets are labelled as *Sunrise* or *Sunset* for the same day, as explained in section II. As one detection per day suffices for synchronization, these metrics are more relevant than the conventional recall. We also report class-wise precision.

To create a diverse dataset encompassing all seasons and multiple years, we used data from Wendlingen, Stuttgart, and St. Leon-Rot (Germany) for the period 2021–2023 [10] to test the accuracy.

A. Impact of Training Data Size

To determine how much training data is required for reliable performance, we evaluate classification accuracy as a function of training set size. The full dataset comprises approximately 9 years of transitions, of which 80% are allocated to the training set and 20% to the test set. Each data point corresponds to a mode transition from *Active* to *Sleep* or vice versa. The number of such transitions varies between 17 and 325, with an average of 63.97, per day.

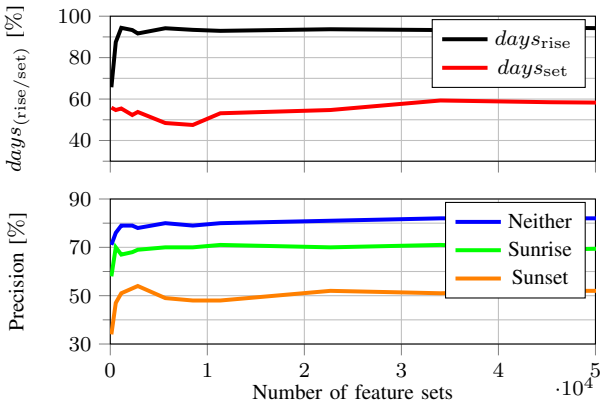


Fig. 2: Impact of training set size on classification performance for Random Forest.

Figure 2 shows that the Random Forest model—representative of all models—experiences rapid performance gains as the training size increases, up to approximately 2,000 feature sets (equivalent to roughly 31 d of data). Beyond this point, additional data yields diminishing returns, with performance gradually plateauing. Notably, none of the models exhibit signs of overfitting at larger training sizes.

Table II summarizes the optimal training set sizes for each model. Most models—except Naive Bayes—plateau in performance with around 2,000 transitions. XGBoost offers the best trade-off between detection coverage and class precision.

TABLE II: Best-performing training set sizes and associated class precision and daily detection rates.

	Feature Sets	$days_{\text{rise}}$	$days_{\text{set}}$	***Neither***	***Sunrise***	***Sunset***
XGBoost	22,681	93%	65%	82%	72%	58%
Random Forest	45,362	94%	58%	82%	69%	52%
Naive Bayes	11,340	99%	73%	69%	27%	34%
Neural Net	102,065	95%	41%	80%	71%	48%

TABLE III: Best-performing window and offset combinations and associated metrics.

	Win/ Offset	$days_{\text{rise}}$	$days_{\text{set}}$	***Neither***	***Sunrise***	***Sunset***
XGBoost	40/10	98%	68%	74%	84%	59%
Random Forest	35/5	98%	0%	68%	86%	100%
Naive Bayes	65/5	99%	75%	74%	59%	38%
Neural Network	35/5	98%	32%	70%	90%	57%

In contrast, Naive Bayes, though converging quickly, performs poorly in distinguishing classes, particularly for *Sunrise* and *Sunset*.

Assuming an average of 63.97 feature sets per day, the *Neural Network* required 1595.51 d (4.37 years) of data, while *XGBoost* needed only 354.56 d (0.97 years). All models, however, performed acceptably when training was stopped earlier—around 2,000 feature sets (31.26 d)—once accuracy converges. Note that training data was not grouped by season; samples from all months were mixed across training and testing.

Naive Bayes consistently performed worse than the other models, primarily due to its assumption of conditional independence between features, which does not hold for this task. Its reliance on Gaussian distributions and linear decision boundaries further limits its effectiveness when dealing with the likely non-linear structure of the data.

B. Sensitivity to Labelling Window and Offset

We next evaluate how sensitive the models are to the size and offset of the labelling window used to assign ground-truth labels to transitions. Each transition is labelled based on whether it falls within a certain time window around the known sunrise or sunset time on that day. We vary both the window size and the offset independently from 5 to 65 minutes, in 5-minute increments. When evaluating one class, the parameters for the other class are held fixed.

As Table III shows, all models achieve their best performance with window sizes between 35 and 40 minutes and small offsets of 5 or 10 minutes. Larger offsets generally degrade performance, likely because they cause the labelling window to shift too far from the actual solar transition. Among the two events, *Sunrise* transitions are detected more reliably. Using this time window, the approach attains comparable accuracy to *CHRT*, provided that the smallest possible capacitor and the MCU from [5] are used.

Interestingly, Random Forest achieves high precision for *Sunset*, which is due to the extremely low recall—the model predicts almost no transitions as *Sunset*. This is illustrated in

TABLE IV: Confusion matrix for Random Forest using a $35\,\mathrm{min}$ window and $5\,\mathrm{min}$ offset.

		Predicted Classes		
		Neither	*Sunrise*	*Sunset*
Actual Classes	*Neither*	13421	1172	0
	Sunrise	1347	7334	0
	Sunset	5063	13	2

TABLE V: Energy consumption and execution times for the different classification models per prediction.

Model	**Energy** [µJ]	**Time** [µs]
XGBoost	4.00	605.96
Random Forest	0.18	26.62
Naive Bayes	6.05	916.69
Neural Network	14.16	2146.11
CHRT [8 h]	4480.00	NA
GNSS	44 900.00	NA

the confusion matrix in Table IV, where almost no sunset transitions (only 2) are identified.

C. Runtime Performance on Embedded Devices

Since classification is not the node's primary task, minimizing the time and, with that, the energy required for classification is essential. We evaluate the execution time and energy consumption of the classification task on a sample hardware presented in [2]. Therefore, we translated our models into C code and measured with the microcontroller's internal timer the computation time for the following steps:

1) Determine the median, maximum, and minimum for the sleep durations
2) Calculate the mean gradient and its absolute value for sleep durations
3) Repeat computations for the active durations
4) Take the feature vector and classify it with the model

We repeated the classification ten times per model and obtained the average calculation time. Based on our measured execution times, we calculated the energy consumption, assuming a current draw of $2\,\mathrm{mA}$ and a supply voltage of $3.3\,\mathrm{V}$, as detailed in [5]. Table V shows that *Random Forest* outperforms the other models in terms of execution time and energy consumption. One reason for that is that our translated *Random Forest* model uses solely integer operations, whereas *XGBoost* relies on floating-point arithmetic. Additionally, *Random Forest* employs simple majority voting, while *XGBoost* uses the computationally intensive *Softmax* function.

Assuming that our model detects a sunrise on the first attempt and stops classification it consumes two to three orders of magnitude less energy than existing methods such as *CHRT* [3] and *GNSS* [11]. Even in the worst-case scenario, where classification is continued throughout the day, energy consumption remains substantially lower than that of traditional methods. For example, the *Neural Network*, with 162 predictions in a single day, still consumes only $2.293\,\mathrm{mJ}$—about half the energy consumption of *CHRT*. This significant energy reduction frees resources for the node's primary tasks.

IV. Conclusion

We examined the use of TinyML to enable absolute timekeeping on batteryless IoT devices by predicting sunrise and sunset based on state transition patterns. Models such as XGBoost demonstrated a favourable balance between accuracy and energy efficiency, providing a synchronization window of $40\,\mathrm{min}$.

The energy savings compared to traditional methods are promising, and importantly, our model operates without requiring any additional hardware. However, trade-offs between model complexity and resource constraints must be carefully managed. Future work should focus on improving model accuracy and evaluating its generalization capabilities across different locations and seasonal variations.

Acknowledgment

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as part of the "DI2T" project (project number 492151098).

References

[1] M. Afanasov, N. A. Bhatti, D. Campagna, G. Caslini, F. M. Centonze, K. Dolui, A. Maioli, E. Barone, M. H. Alizai, J. H. Siddiqui, and L. Mottola, "Battery-less zero-maintenance embedded sensing at the mithræum of circus maximus," in *SenSys, 2020*. Virtual Event, Japan: ACM, 2020, p. 368–381. [Online]. Available: https://doi.org/10.1145/3384419.3430722

[2] S. Mosavat, M. Zella, M. Handte, A. J. Golkowski, and P. J. Marrón, "Experience: Aristotle: Wake-up receiver-based, star topology batteryless sensor network," in *IPSN, 2023*. San Antonio, TX, USA: ACM, 2023, pp. 177–190. [Online]. Available: https://doi.org/10.1145/3583120.3586961

[3] J. de Winkel, C. Delle Donne, K. S. Yildirim, P. Pawełczak, and J. Hester, "Reliable timekeeping for intermittent computing," in *ASPLOS, 2020*. Lausanne, Switzerland: ACM, 2020, p. 53–67. [Online]. Available: https://doi.org/10.1145/3373376.3378464

[4] A. Rahmati, M. Salajegheh, D. Holcomb, J. Sorber, W. P. Burleson, and K. Fu, "TARDIS: Time and remanence decay in SRAM to implement secure protocols on embedded devices without clocks," in *USENIX Security, 2012*. Bellevue, WA, USA: USENIX Association, 2012, pp. 221–236.

[5] J. Göpfert and B.-C. Renner, "High-level simulation of the timely behavior of intermittent systems," in *ENSsys, 2023*. Istanbul, Turkiye: ACM, 2023, pp. 1–7. [Online]. Available: https://doi.org/10.1145/3628353.3628539

[6] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *KDD, 2016*. ACM, 2016, pp. 785–794. [Online]. Available: http://dx.doi.org/10.1145/2939672.2939785

[7] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[8] Vikramkumar, V. B, and Trilochan, "Bayes and naive bayes classifier," 2014. [Online]. Available: https://arxiv.org/abs/1404.0933

[9] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2021.

[10] A. Dittmann, F. Dinger, W. Herzberg, N. Holland, S. Karalus, C. Braun, R. Zähringer, W. Heydenreich, and E. Lorenz, "Pv-live dataset - measurements of global horizontal and tilted solar irradiance," 2025. [Online]. Available: https://doi.org/10.5281/zenodo.14750853

[11] A. Adewumi and A. Ibraheem Abiodun, "A review of global navigation satellite systems (GNSS) and its applications," *International Journal of Scientific and Engineering Research*, vol. 12, pp. 1042–1049, 2021.