

Adaptive Threshold Sleep Strategy for ESP32-Based IoT: A Comparative Power Analysis

Siti Maryam Zainol^{1,2,*}, R. Badlishah Ahmad¹, Shuhaizar Daud¹ and Ahmad Kadri Junoh¹

¹Faculty of Intelligent Computing (FKC), Universiti Malaysia Perlis (UniMAP), Pauh Putra Campus, 02600 Arau, Perlis, Malaysia.

²Jabatan Pendidikan Politeknik dan Kolej Komuniti JPPK, Aras 4-7, Galeria PJH, Jalan P4 W, Persiaran Perdana, 62100 Putrajaya, Malaysia.

ABSTRACT

In battery-powered Internet of Things (IoT) devices, energy efficiency is a significant limiting factor for operation time, particularly in remote or infrastructure-scarce environments. This paper explores the effects of high-performance sleep modes on microcontroller-based IoT device power usage. Microcontroller as sensor node has ready build in power saving features which is sleep mode. This save more operation cost because no need external system to reduce the power. This power optimization is directly on the microcontroller itself not just in the network layer with ESP32 as a platform, INA219 sensor is used to capture current for every situation and DHT11 sensor is used to read temperature. It compares three operating modes: continuous active mode (no sleep implementation), fixed scheduled sleep (in which there are periodic low-power states), and an adaptive threshold-based sleep approach that selectively triggers sleep in response to temperature changes. The results from the experiments demonstrate that the threshold-based deep sleep strategy provides a 78.24% energy saving over the active baseline and performs better than fixed scheduled strategies with guaranteed data integrity. These findings indicate the promise of context-aware adaptive sleep algorithms for reducing microcontroller power consumption without degrading system responsiveness. The solution outlined is a viable way forward for energy-resilient IoT deployments with minimal infrastructure support.

Keywords: Internet of Things, Energy Efficient, Context-Aware Adaptive Sleep Algorithms, Battery Extension, Microcontroller

1. INTRODUCTION

Internet of Things (IoT) has revolutionized modern industries by enabling real-time data collection and automation. However, the continuous operation of IoT devices leads to high power consumption, which can limit their efficiency, particularly in battery-powered applications and using coin battery such as animal monitoring tag, medical wearable, smart watch and also pest detection traps. There are many approaches to optimize power in IoT system such as hardware and software techniques in every layer in IoT architecture from sensor (sensing layer) to end user (application layer) as Figure 1. Most previous work such as [1], [2], [3], [4], [5], [6] overcome power consumption issues in between layer A: data processing layer and application layer or layer B: sensor layers and network layers as in figure 1. They consider data traffic, quantity of sensor nodes, transmission ranges and behaviour among sensor nodes not specifically in IoT device itself that process data to be sent to application layer (refer to figure 1). For example, [1] adapts deep reinforcement learning (DRL) according to the restoration mechanism, which the controller performs sleep scheduling to achieve high accuracy of the restored environmental map without energy outage of IoT devices. The framework approaches in a big monitoring environment which considers many sensor nodes and also not exactly approach in IoT device. It also had to add external devices such as energy harvesting module which costing more instead using the ready features like sleep mode in IoT devices

*maryamunimapphd@gmail.com

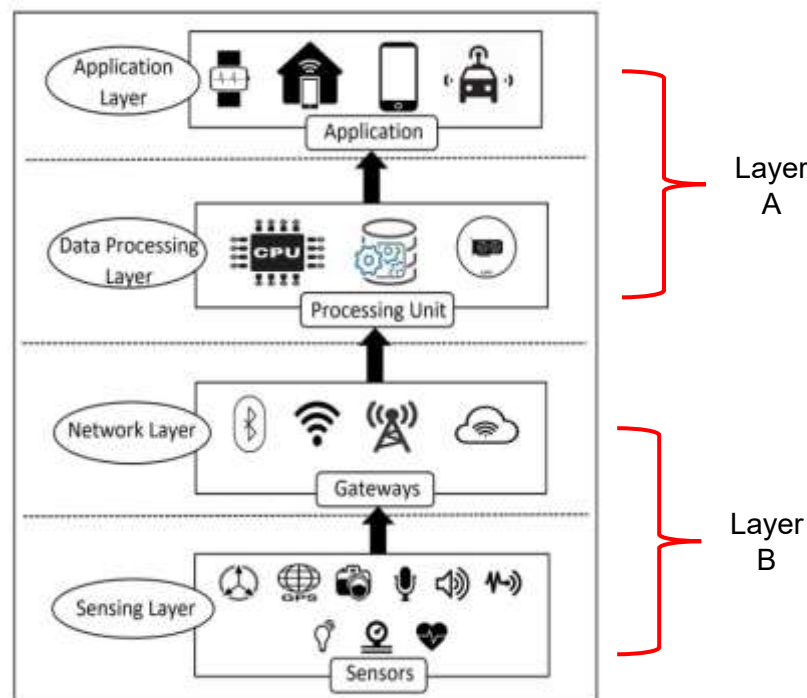


Figure 1. IoT Architecture Layers and Components [7]

[8] lists research works of variety areas using sleep strategy such as WSN - Wireless Sensor Node, WUR – Wake Up Radio, small cell and microcontroller itself. From the result and discussion, there is no research applying sleep strategy to minimize the power on both microcontroller and the system itself. Furthermore, most sleep mode is implemented using fix scheduled and less of adaptive sleep strategy such as threshold-based or environmental-based applied to reduce the power optimization on microcontroller-based IoT devices. Consequently, this paper will fill the gap which the power optimization is done on microcontroller-based IoT device itself. There are variety IoT device base available in the market such as FPGA - Field-Programmable Gate Array, DSP – Digital Signal Processor and microcontrollers [9], [10], [11]. Nonetheless, the majority of IoT systems utilize microcontrollers as platforms for many functions, including automation, data analytics, remote control, and real-time monitoring. This is due to the microcontroller being a significantly lower-cost technology compared to FPGA, DSP, and other alternatives.

Additionally, the microcontroller has various power modes features such as deep sleep mode, light sleep mode and hibernate mode to optimize energy consumption, particularly in battery-operated applications such as agriculture monitoring in the middle of outdoor field and solar monitoring in a large outdoor field. Even the wired-operated system in building need this power optimization to achieve SDG – Sustainable Development Goal 7 which is affordable and clean energy [12]. Because of microcontroller-based IoT has many benefits, a microcontroller, ESP32 is chosen. Three power mode of ESP32 are analysed in this paper which are unoptimized/active mode, deep sleep mode and light sleep mode. Furthermore, deep sleep mode and light sleep mode are analysed in two strategies which are fix scheduled sleep and threshold-based sleep. Subsequently, this chapter will discuss the most used power mode of familiar microcontroller-based IoT device in Table 1.

2. LITERATURE REVIEW

2.1 Power Mode Of Variety Microcontrollers

Table 1 categorizes the power modes of commonly used microcontrollers in IoT applications based on their energy-saving capabilities. Active Mode represents the normal operating state where the microcontroller is fully functional. Light Sleep / Idle Mode reduces power consumption by lowering the clock speed or disabling certain peripherals while keeping the CPU active, which is useful for applications requiring rapid wake-ups [13], [14]. Deep Sleep / Stop Mode further minimizes energy use by shutting down more components, retaining only essential functions such as RTC - Real-Time Clock and RAM, making it suitable for battery-powered IoT devices requiring periodic operation [15], [16], [17]. Hibernation / Power-down Mode is the lowest power state, where the microcontroller turns off most functions, often requiring an external interrupt or a hardware reset to wake up, optimizing long-term deployments in remote sensor networks [18], [19].

Table 1 Power mode in variety microcontrollers [20], [21]

Microcontroller	Active Mode	Light Sleep / Idle Mode	Deep Sleep / Stop Mode	Hibernation / Power-down Mode
ESP32	Active	Light Sleep	Deep Sleep	Hibernation
ESP8266	Active	Modem Sleep, Light Sleep	Deep Sleep	Hibernation
STM32	Run Mode	Sleep Mode	Stop Mode	Standby Mode, Shutdown Mode
ATmega328P	Active Mode	Idle Mode	Power-save Mode	Power-down Mode
Nordic nRF52	Active Mode	System ON Sleep	System OFF Sleep	System OFF Mode with RAM retention
TI MSP430	Active Mode	LPM0 (Low Power Mode 0)	LPM1, LPM2, LPM3	LPM4 (Deep Sleep)

Recent studies highlight the importance of sleep mode to balance power efficiency and performance. For example, [22] demonstrated that deep sleep modes in ESP32-based IoT devices could extend battery life by up to 83% in low-data applications. Similarly, [23], [24] explored how STM32's stop mode significantly reduces power draw in IoT deployments. Moreover, [25] investigated hibernation strategies in Nordic Semiconductor's nRF52 series, revealing that strategic wake-up scheduling can optimize energy consumption for smart city applications. Next, fix scheduled sleep and adaptive threshold-based sleep are reflected by example of previous works.

2.2 Fix Scheduled Sleep Strategy

There are two main sleep strategies often used which are fix scheduled sleep and adaptive threshold-based sleep. Fix scheduled sleep is a strategy that microcontroller drives to sleep mode at the set time without considering the environmental influence. It can save more energy but it can cause more data missing. It because of microcontroller will go sleep although the monitored data changed significantly. For instance, a study by [26] demonstrated that enabling deep sleep mode in an ESP32-based IoT system reduced energy consumption by 83.38%, leading to a fivefold increase in battery runtime. Similarly, [27] designed an IoT-based smart monitoring system for hydroelectric generators, employing deep sleep mode to minimize power consumption while continuously monitoring parameters such as pH levels, voltage, and current. In addition to environmental monitoring, power efficiency is a key factor in the design of wearable IoT devices, particularly in healthcare applications that require continuous operation.

[28] proposed a hybrid approach combining both hardware and software optimizations to minimize energy consumption in wearable devices for Alzheimer's patient care. Their method involved selecting low-power components and implementing deep sleep mode, reducing power draw to an average of 9.16 mA, thereby extending battery life to 38.2 hours. [29] investigated the use of second-generation photovoltaic (PV) cells to power a LoRa-based IoT system, demonstrating that a CdTe PV module with 16% efficiency could sustain an ESP32-based LoRa node. Their findings showed that in deep sleep mode, the node consumed only 165 μ W of power, allowing for continuous operation with supplementary energy storage. While this approach offers a promising solution for ultra-low power applications, it also underscores a limitation of fixed scheduled sleep—relying solely on predefined sleep intervals without considering variable energy availability from renewable sources may lead to inefficient power utilization.

2.4 Adaptive Threshold-Based Sleep Strategy

The research on energy-efficient microcontrollers highlights the significance of adaptive threshold-based sleep strategies in optimizing power consumption, particularly in IoT applications. Adaptive threshold-based sleep strategy concerns environmental factor. It does not go to sleep periodically. It considers the significant changes of the important data such as temperature, humidity and also pressure, depends on the requirement of the system. [30] adapt the sleep strategy with the amount of received and stored energy. The microcontroller will wake up when the capacitor achieved the desired charge. [31] conducted a comprehensive analysis of the Renesas DA16200 microcontroller, comparing its energy efficiency with the ESP8266 under identical conditions. Their study emphasized the importance of transient current profiling and operational mode analysis in understanding power consumption patterns. Similarly, [32] designed a solar-powered automatic weather station using the ESP32, incorporating deep sleep mode and periodic data transmission to optimize energy use. The findings indicate that adaptive sleep based on data acceptance can significantly extend battery life while maintaining a high data acceptance rate. However, while these studies demonstrate the effectiveness of sleep modes in reducing power consumption, they do not address the trade-offs between energy savings and system responsiveness, which are critical considerations in real-time applications.

Beyond conventional microcontroller-based sleep strategies, research has explored more innovative power optimization techniques, such as leveraging energy harvesting and adaptive activation mechanisms. [33] investigated the feasibility of hydraulic energy harvesting for IoT flow measurement devices, demonstrating how integrating microturbines with low-power sleep modes can achieve energy self-sufficiency. Their work highlights the potential of alternative energy sources in enhancing device sustainability. Similarly, [34] implemented an adaptive sleep strategy in a LoRa-based robotic arm control system, where the microcontroller remains in continuous sleep mode and only activates upon detecting a signal transmission event. Also, this method reduces unnecessary power drain and extends operational longevity. While these approaches showcase the potential of adaptive power management, they introduce design complexities, such as optimizing wake-up thresholds and ensuring consistent energy availability, which require further refinement.

Despite extensive research on power optimization in IoT systems, much of the existing work has concentrated on fixed sleep scheduling, hardware-level optimizations, or network-layer adjustments. Fixed sleep scheduling approaches, while simple, fail to adapt to dynamic workloads and environmental conditions, leading to either wasted energy during idle states or missed opportunities for timely data transmission. Hardware-level solutions often demand additional components or design changes, increasing cost and complexity, whereas network-level approaches primarily target communication efficiency rather than device-level energy management. For ESP32-based IoT devices which widely adopted due to their Wi-Fi and sensor integration, research has shown that built-in deep sleep and light sleep modes provide significant

power savings. However, studies often lack a comparative evaluation of different sleep strategies under environmental contextual aware parameters such as temperature changes. Consequently, there is limited understanding of how an adaptive threshold-based strategy performs relative to conventional approaches in balancing energy savings and data availability.

Addressing this gap is crucial because IoT deployments are increasingly deployed in energy-constrained and remote environments, where extending device lifetime directly reduces maintenance costs and enhances sustainability. An adaptive threshold sleep strategy has the potential to optimize power consumption without sacrificing responsiveness, making it especially relevant for real-world IoT applications such as environmental monitoring, smart agriculture, and healthcare. By systematically conducting a comparative power analysis of adaptive threshold sleep against fixed and unoptimized strategies on the ESP32 platform, this research will contribute novel insights into the trade-offs between power efficiency. Moreover, it provides a practical framework for developers and system designers to implement context-aware energy management in IoT devices, aligning with broader sustainability goals such as reducing battery waste, lowering carbon footprint and supporting SDG 7: Affordable and Clean Energy.

3. METHODOLOGY

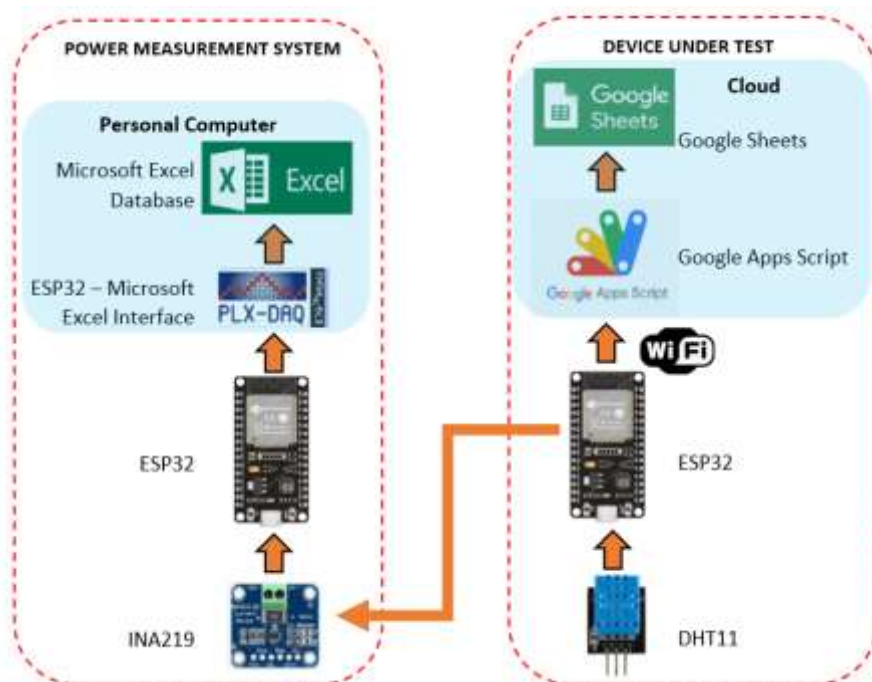


Figure 2. Block diagram of power analysis setup using current sensor, INA219.

This research starting with setting up a microcontroller-based IoT testbed (ESP32 is selected) which consist of two sides which are power measurement system and device under test as Figure 3. Power measurement system consist of microcontroller ESP32 connected to current sensor (INA219) and personal computer (PC) while device under test comprise of microcontroller ESP32 connected to DHT11, temperature sensor and cloud. The detail of this setup will be explained using Figure 2.

Current data are stored in personal computer through USB connection by using PLX DAQ software to interface ESP32 and Microsoft Excel. While Google Sheet is used as cloud database to store temperature data using WiFi connection and Google Apps Script as interface. Refer to Figure 3, Vcc, GND, SCL, and SDA connections between the ESP32 and INA219 indicates the use of an I2C communication protocol for data exchange. Then, positive probe and negative probe of current sensor, INA219 are connected serially between power supply and ESP32 to collect recent electrical current from another ESP32 that executes temperature data logging to cloud through

Wi-Fi. This cloud purposes for remote monitoring, where data can be accessed and analysed from any location. In this research, Google Sheet is used. At the same time, the current data will be sent to database in PC using USB cable. These steps will be repeated for three power mode situations: unoptimized, fix scheduled sleep and threshold based adaptive sleep strategy. Then, the data will be analysed in aspect of current consumption and quality of data which consider the missing of significant data. In this research, temperature data will be used and the changes between recent temperature and previous temperature will be considered.

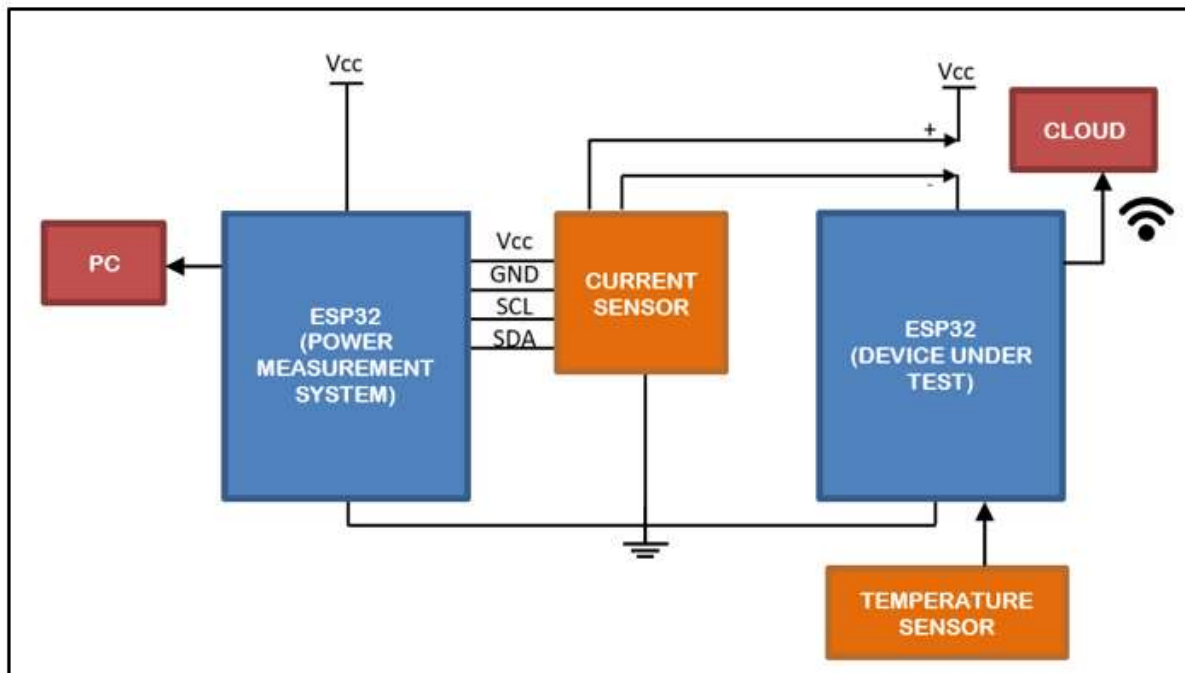


Figure 3. Connection among current sensor, ESP32- Temperature logger and ESP32- Current Logger.

Next, the flowcharts of current measurement for three power mode situations: unoptimized, fix scheduled sleep and threshold based adaptive sleep strategy are explained.

3.1 Current Measurement System

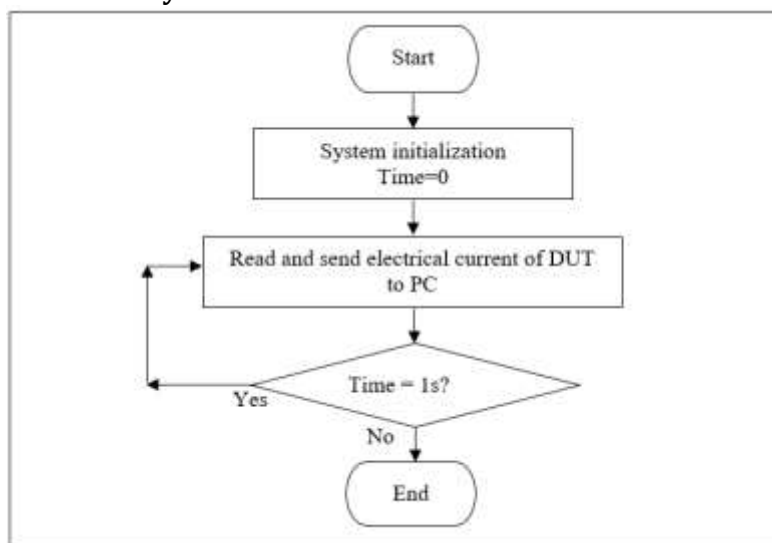


Figure 4. Current measurement flow chart

The process of current measurement as in Figure 4 is running parallel with every experiment running. The experiments including unoptimized system, fix scheduled sleep and threshold-based sleep. The INA219 streams one current sample per second ($f_s=1\text{Hz}$). Then, the current will be sent to Microsoft Excel through a software called PLX-DAQ in personal computer (PC). One current sample per second is decided to observe current without missing any situation especially the important event such as WiFi is turned on and data is sending. To calculate average of current, I_A , Eq. (1) is used.

$$I_A = \frac{1}{N} \sum_{k=1}^N I_k \quad (1)$$

where I_k is the k^{th} one-second sample and $N=7200$ for a two-hour run.

Next, power, P_k is considered by Eq. (2).

$$P_k = V \times I_k \quad (2)$$

where $V = 5\text{V}$.

So that, the total energy, E calculated using simplified formula, Eq. (5).

$$E = \sum_{k=1}^N P_k \Delta t \quad (3)$$

$$E = V \Delta t \sum_{k=1}^N I_k \quad (4)$$

Simplify as:

$$E = V \times N \Delta t \times I_A \quad (5)$$

where $\Delta t = 1\text{s}$.

Then, percentage of power saving is determined by Eq. (6).

$$\text{Saving (\%)} = \left[1 - \frac{I_{\text{strategy}}}{I_{\text{unoptimized}}} \right] \times 100 \quad (6)$$

where I_{strategy} is current consumption of sleep strategies such as fix scheduled sleep and threshold-based sleep strategy.

3.2 Unoptimized System

Figure 5 shows the flowchart of unoptimized system. This system will collect temperature data for every 5 minutes for 2 hours according to the minimum cycle time to capture temperature in [35] and send to cloud database, Google Sheet. At the same time, the electrical current reading is sent to PC for every 1 second as the current measurement flowchart in Figure 4. This is to prepare for data analysis among three strategies: unoptimized, fix scheduled sleep and threshold based adaptive sleep.

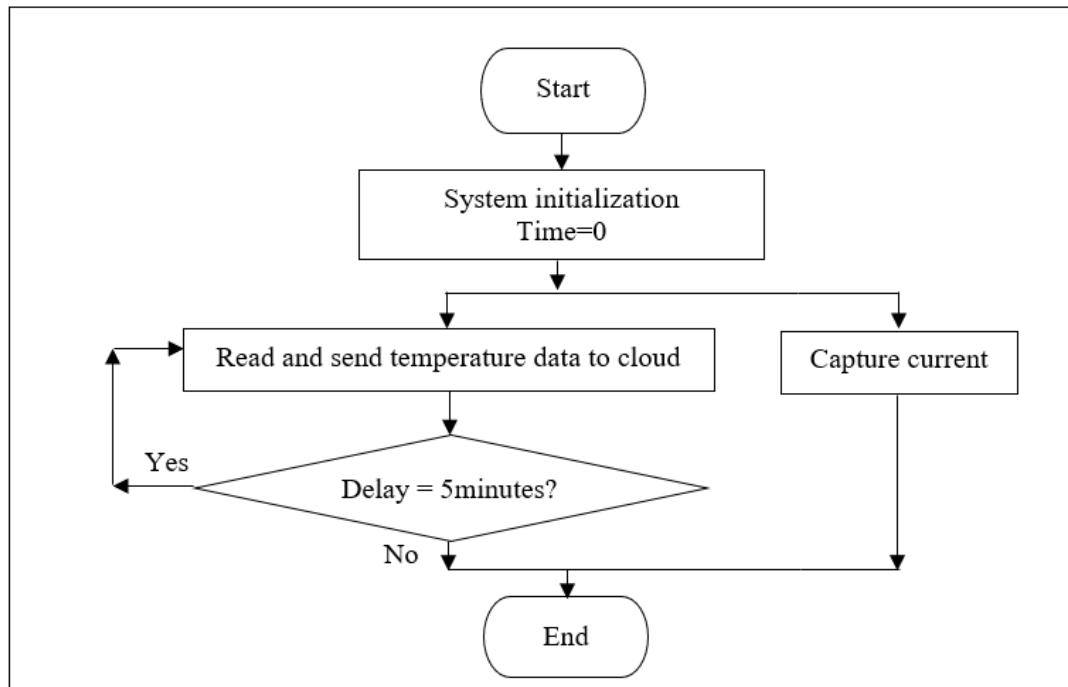


Figure 5. Unoptimized system flowchart.

3.3 Fix Scheduled Sleep

Refer to Figure 6, the data will be collected and send to cloud database, Google Sheet. Then, the system will go to sleep for 5 minutes in duration 2 hours. At the same time, the electrical current reading is sent to PC for every 1 second as the current measurement flowchart in Figure 4. It means that no data collection but the electrical current will still be observed. This sleep including two sets experiment which are light sleep and deep sleep.

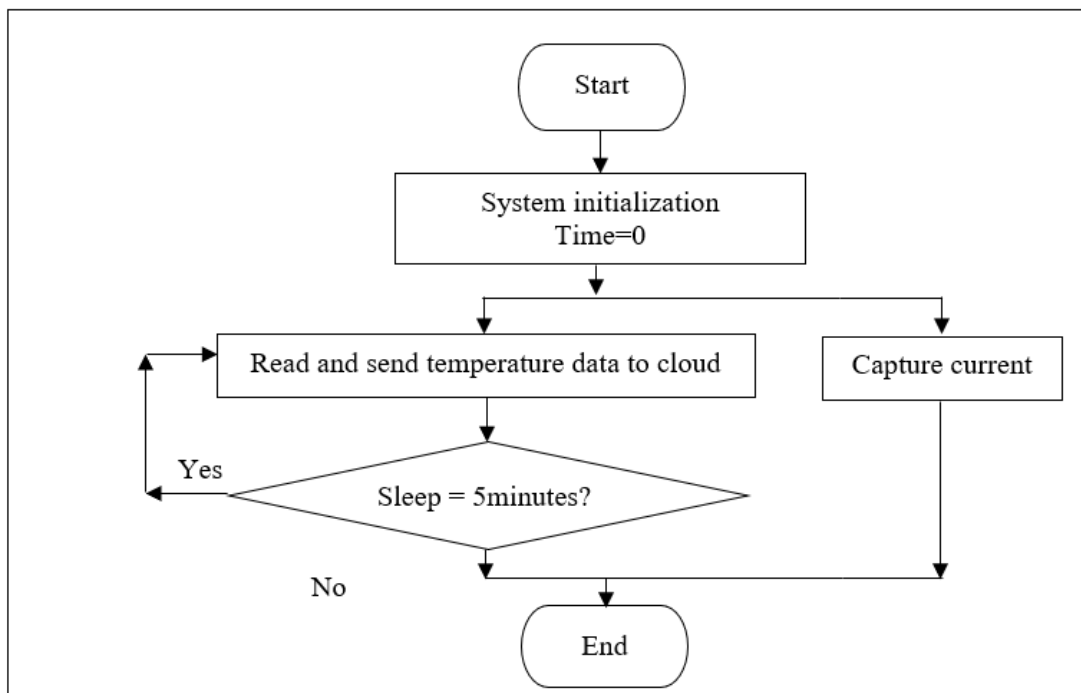


Figure 6. Fix scheduled sleep flowchart.

3.4 Threshold Based Adaptive Sleep

Threshold based adaptive sleep strategy acts as in Figure 7. When current temperature, X_c is not equal to previous temperature, X_p , the latest temperature will be sent to cloud database. Else, microcontroller will go to sleep by timer, 5 minutes. So that, the significant data won't be missed because microcontroller do not sleep when there is significant change. At the same time, the electrical current reading is sent to PC for every 1 second as the current measurement flowchart in Figure 4. This observation is divided into two parts which are deep sleep and light sleep implementation. It's because both sleep mode have their own benefits. So that, the best power consumption for both sleep mode can be determined.

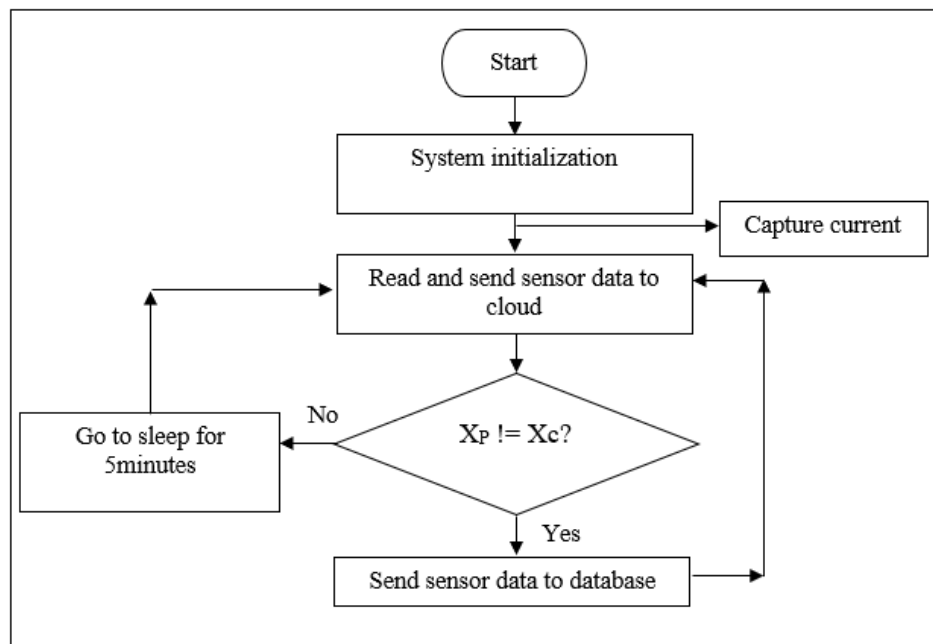


Figure 7. Threshold based adaptive sleep strategy flowchart.

4. RESULTS AND DISCUSSION

The experimental result given in Table 2 describes the impact of various power mode and sleep strategies on energy consumption of a microcontroller-based IoT system which has made significant missing factors a part of consideration. The power mode involves unoptimized system, fix scheduled deep sleep, fix scheduled light sleep, threshold deep sleep and threshold light sleep strategy.

4.1. Current Observation

The unoptimized mode achieves maximum energy usage at 1256.964 Joules and it becomes the reference to be compared with others. The fixed scheduled deep sleep reduces to extremely low energy usage at 283.606 Joules which 77.44% reduction in energy from unoptimized mode. However, this mode tends to have high data loss, and it may influence its reliability. Fixed scheduled light sleep consumes 318.973 Joules with 74.62% savings in energy but experiences considerable data loss.

Table 2 Comparison of energy consumption among variety power mode

Strategy	Average of energy in an hour (Joule)	Energy Saved (%)	Comparison with Fixed Scheduled Sleep	Significant Data missing possibility?
Unoptimized	1256.964	0.00	-	No
Fix Scheduled - Deep Sleep	283.606	77.44	0.00	Yes
Fix Scheduled - Light Sleep	318.973	74.62	12.47	Yes
Threshold - Deep Sleep	273.518	78.24	3.56	No
Threshold - Light Sleep	312.734	75.12	10.27	No

On the other hand, Figure 8 shows the threshold-based deep sleep scheme experiences the maximum energy saving of 78.24%, and consumes 273.518 Joules, with no considerable data loss, and hence is an acceptable choice. The threshold-based light sleep strategy employs 312.734 Joules, achieving a 75.12% savings in energy, losing very little data, further contributing to its reliability.

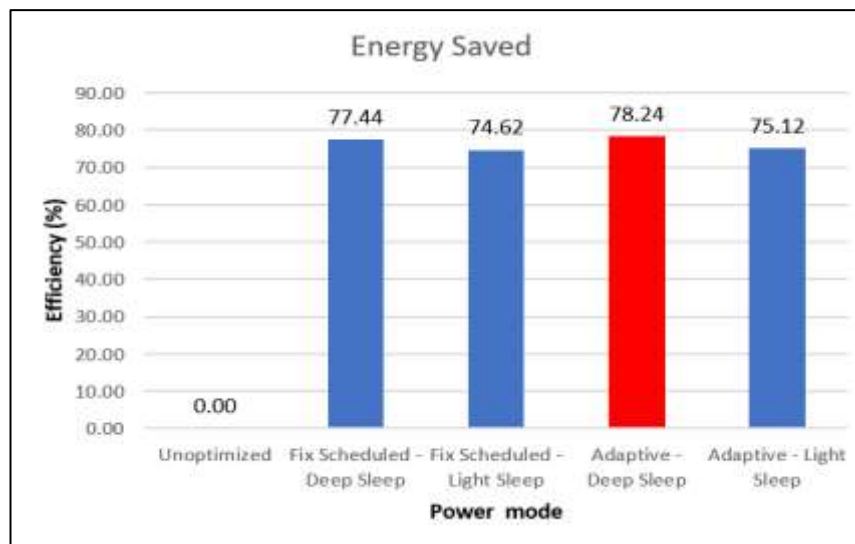


Figure 8. Percentage of energy saved of various power mode strategies compared to unoptimized system on a microcontroller-based IoT system

Table 2 also presents the comparative analysis of power consumption of different sleep strategies in a microcontroller-based IoT system such as fix-scheduled sleep and threshold-based sleep. The fixed scheduled deep sleep mode is used as the reference with 0.00% power consumption. The fixed scheduled light sleep mode demonstrates a significant increase in power consumption up to 12.47%, which validates that light sleep in a fixed schedule is less power-efficient due to frequent wake-ups. The threshold-based deep sleep mode, however, has improved power efficiency as it reduces energy consumption by 3.56%, which shows that an adaptive approach can be more effective at power optimization than a fixed schedule. The threshold-based light sleep mode still consumes 10.27% more energy than the baseline but performs better than the fixed scheduled light sleep mode.

Now, let say battery 5000mAh LiPo Battery [36] is used to expect battery life and it is calculated as in Eq. (7) [37]:

$$\text{Expected Battery Life} = \frac{\text{Battery Capacity(mAH)}}{\text{Current Consumption (mA)}} \quad (7)$$

Table 3 Significant impact of different sleep strategies on the expected battery life

Strategy	Expected Battery Life
Unoptimized – no sleep	$\frac{5000}{67.02} = 3 \text{ days } 2 \text{ hours } 35 \text{ minutes } 58 \text{ seconds}$
Fix Scheduled - Deep Sleep	$\frac{5000}{15.75} = 13 \text{ days } 5 \text{ hours } 26 \text{ minutes } 21 \text{ seconds}$
Fix Scheduled - Light Sleep	$\frac{5000}{17.72} = 11 \text{ days } 18 \text{ hours } 9 \text{ minutes } 19 \text{ seconds}$
Threshold - Deep Sleep	$\frac{5000}{15.20} = 13 \text{ days } 17 \text{ hours } 2 \text{ minutes } 45 \text{ seconds}$
Threshold - Light Sleep	$\frac{5000}{17.37} = 11 \text{ days } 23 \text{ hours } 47 \text{ minutes } 4 \text{ seconds}$

The results presented in Table 3 clearly demonstrate the significant impact of different sleep strategies on the expected battery life of microcontroller-based IoT devices. In the unoptimized case, where no sleep mode is applied, the device consumes an average current of 67.02 mA, leading to a battery life of only about 3 days. This short operational period highlights the inefficiency of keeping the device fully active at all times, as continuous operation results in rapid battery depletion. Such an approach may be acceptable for short-term experimental setups but is unsustainable for long-term IoT deployments, especially in remote or battery-powered environments.

When fixed scheduled sleep strategies are introduced, a notable improvement in energy efficiency is observed. By applying deep sleep on a fixed schedule, the average current consumption drops to 15.75 mA, extending the battery life to more than 13 days. In comparison, fixed scheduled light sleep achieves slightly lower efficiency, with a current consumption of 17.72 mA and an expected battery life of about 11 days and 18 hours. These findings suggest that while both fixed sleep approaches are effective in prolonging device operation, deep sleep is more advantageous due to its lower current draw, making it preferable for energy-critical IoT systems.

The threshold-based strategies further enhance performance by adapting the sleep-wake cycle to environmental or operational conditions rather than relying on a rigid schedule. Threshold-based deep sleep records the best performance overall, with an expected battery life of almost 14 days, reflecting its ability to minimize unnecessary wake-ups while maximizing time spent in low-power states. Meanwhile, threshold-based light sleep achieves close to 12 days of battery life, slightly outperforming the fixed scheduled light sleep strategy due to its adaptive behaviour. These results highlight the importance of context-aware power management, where adaptive strategies offer superior efficiency compared to static scheduling. Overall, the analysis confirms that adopting deep sleep modes and threshold-based mechanisms significantly optimizes battery usage, thereby extending the sustainability and practicality of IoT deployments.

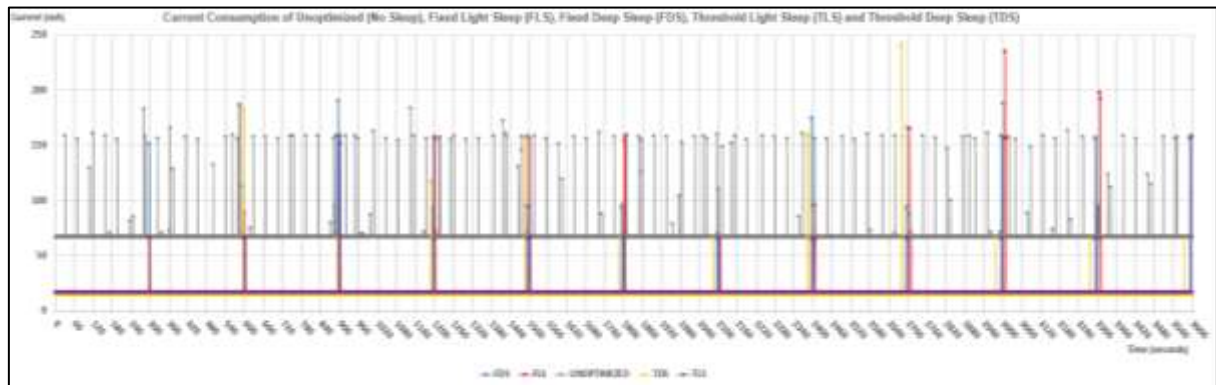


Figure 9. Current consumption of Unoptimized (No Sleep), Fixed Light Sleep (FLS), Fixed Deep Sleep (FDS), Threshold Light Sleep (TLS) and Threshold Deep Sleep (TDS)

Figure 9 presents comparative current consumption profiles for five operating modes: Unoptimized (No Sleep), Fixed Light Sleep (FLS), Fixed Deep Sleep (FDS), Threshold Light Sleep (TLS), and Threshold Deep Sleep (TDS) over a 3600-second interval. The Unoptimized mode is seen to have a constantly high baseline current, 70 mA with frequent current spikes in excess of 150 mA, indicative of continuous operation and frequent Wi-Fi transmissions regardless of data significance. In contrast, the two fixed-schedule methods (FLS and FDS) reduce the baseline consumption and add periodic intervals of high current peaks, corresponding to periodic wake-up events every 5 minutes. FDS is observed to have slightly lower idle current and shorter peak duration than FLS, reflective of the deeper power-down state. More significantly, however, the threshold-based methods (TLS and TDS) produce much flatter current profiles with fewer spikes, particularly TDS, which is an almost constant low-current line with very occasional transmission events. This clearly illustrates the advantage of adaptive, context-aware sleep policies in eliminating redundant wake-ups and conserving energy. TLS, while somewhat more active than TDS, nonetheless illustrates improved efficiency relative to both fixed methods. These results validate the hypothesis that threshold-driven methods, by linking activity to actual environmental change, yield more efficient energy utilization without impacting responsiveness as a key requirement for long-life, autonomous IoT deployments.

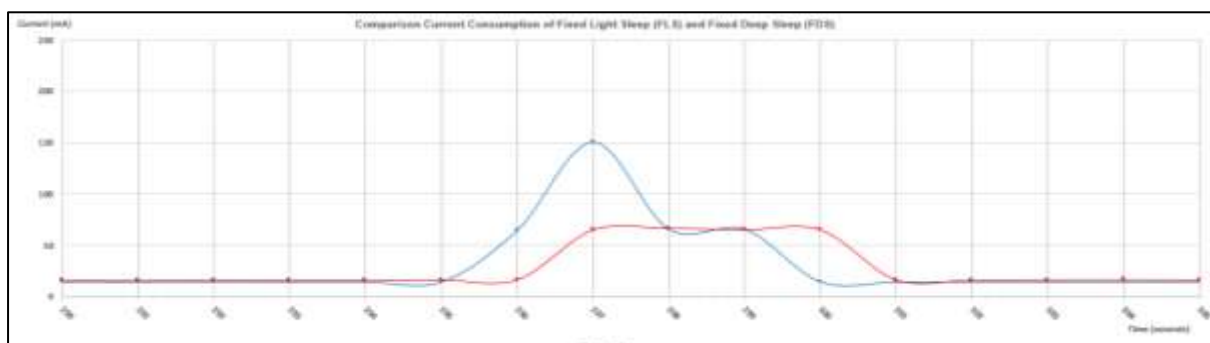


Figure 10. Zoomed waveform comparison of current consumption between Fixed Light Sleep (FLS) and Fixed Deep Sleep (FDS)

Next, current consumption using deep sleep mode and light sleep mode are compared by zooming the waveform of three distinct wake/sleep cycles on an ESP32 as Figure 10. This waveform is taken from observation of fix scheduled sleep strategy. Light sleep (red line) has a higher idle current (20 – 30 mA) with wider wake windows and smoother transitions, indicating partial peripheral retention and faster stabilization. While deep sleep (blue) has abrupt transitions with narrow current spikes which up to 200 mA followed by nearly zero idle current because of characteristic of full reinitialization and Wi-Fi handshake after deep sleep.

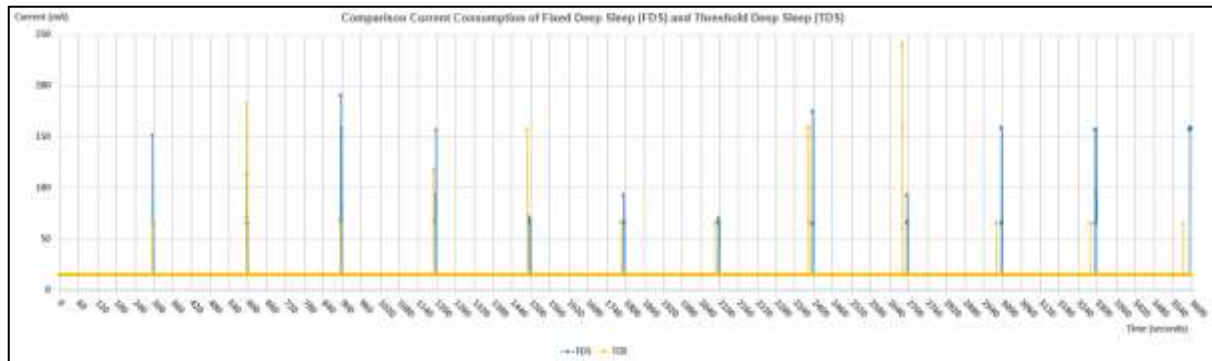


Figure 11. Current consumption of Fixed Deep Sleep (FDS) and Threshold Deep Sleep (TDS)

Figure 11 compares live consumption profiles of FDS (blue line) and TDS (yellow line) approaches during a one-hour (3600second) logging session. For FDS mode, current spikes occur at regular 5-minute intervals, illustrating periodic wake-up and transmission activity regardless of environmental change. Each spike is up to about 190–200 mA, indicating full Wi-Fi wake-up and sensor readout. Between wake intervals, the system deep sleeps, with minimal idle current close to 0 mA. For comparison, the TDS profile illustrates far fewer and more random current peaks, only when temperature change is observed—illustrating the adaptive nature of the threshold logic. Most of the TDS plot is flat, around the 20mA level, indicating extended deep sleep intervals. This confirms that TDS effectively eliminates unnecessary wake-ups and transmissions when environmental conditions have remained steady. Interestingly, both modes benefit from the deep sleep capability of the ESP32, but TDS achieves more energy saving by dynamically associating activity with substantial data change. The randomness of TDS peaks also illustrates real-world environmental fluctuation, further confirming the effectiveness of threshold-triggered operation over fixed scheduling for IoT systems aiming for long-term battery independence.

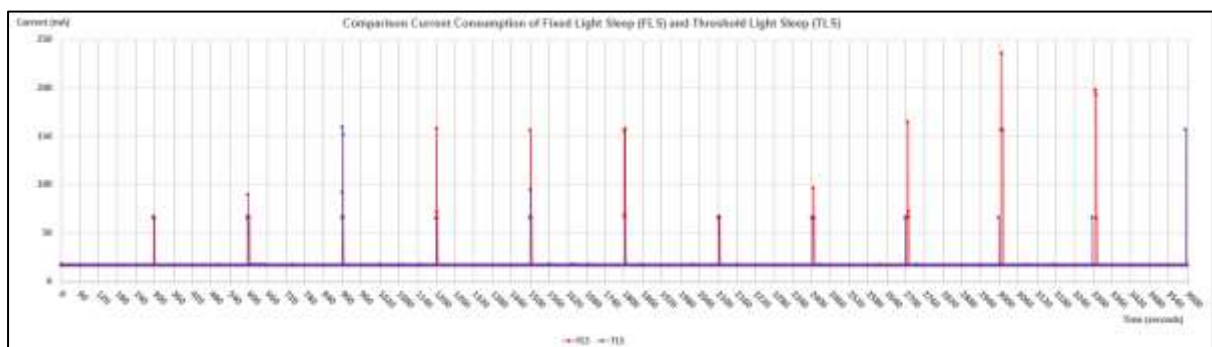


Figure 12. Current consumption of Fixed Light Sleep (FLS) and Threshold Light Sleep (TLS)

Figure 12 plots the side-by-side current consumption of FLS and TLS over a 3600-second operation window. The FLS plot (in red) is characterized by periodic 5-minute spikes in current consumption, with amplitudes of more than 200 mA, reflecting fixed periodic wake-up cycles independent of real environmental changes. In between activations during idle periods, current consumption stays moderately high which between 18 – 20 mA, which is the elevated baseline consumption of the light sleep mode over deep sleep. On the other hand, TLS (in purple) presents

a leaner current profile with fewer and aperiodic spikes corresponding to detected temperature changes. Baseline current is slightly higher but steadier, while peak activity is significantly lower and less frequent compared to FLS. This adaptive performance in TLS demonstrates its efficiency in balancing system responsiveness with the elimination of redundant operations.

Compared with the previous deep sleep graph (Figure 11), it is clear that both deep sleep methods (FDS and TDS) provide lower idle current levels than their light sleep equivalents. Yet, whereas TDS provides minimal activity and maximum savings, TLS offers a compromise, trading power efficiency for quicker wake-up times and faster network rejoining, which is useful for applications needing semi-frequent responsiveness. The basic trade-off of light sleep, lower latency but higher standby consumption. It is demonstrated by the higher baseline of TLS relative to TDS. By linking power consumption to actual sensing requirements, threshold-based implementations of both sleep modes (TLS and TDS) ultimately perform better than fixed alternatives, with TDS offering the highest overall efficiency.

4.2. Temperature Observation

Threshold-based temperature transmission as in figure 16 and 17 is the best method in terms of data reliability and energy saving. Unlike the unoptimized method as in Figure 13, which transmits data every five minutes regardless of change, and the fixed scheduled sleep method as in Figure 14 and 15, which periodically transmits data after sleeping for a predetermined time, the threshold-based method records temperature only if there is a significant change compare to previous temperature. Therefore, redundant transmissions of stable temperature are avoided and only meaningful data is sent to the cloud.

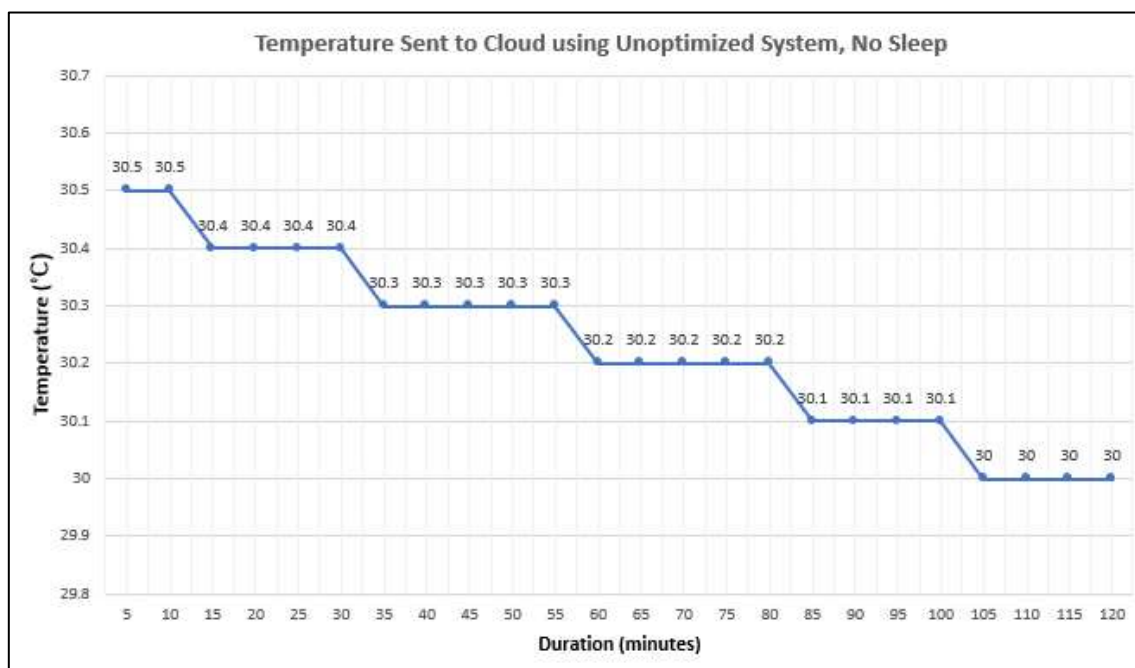


Figure 13. Temperature sent to cloud using unoptimized on a microcontroller-based IoT system

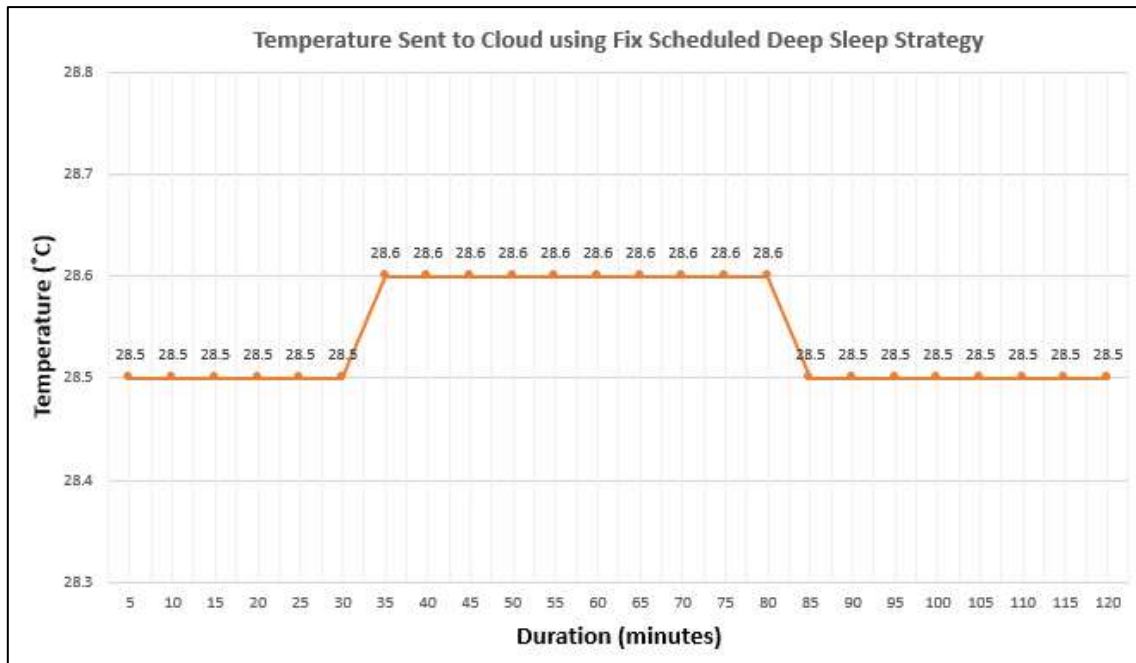


Figure 14. Temperature sent to cloud using fix scheduled deep sleep on a microcontroller-based IoT system

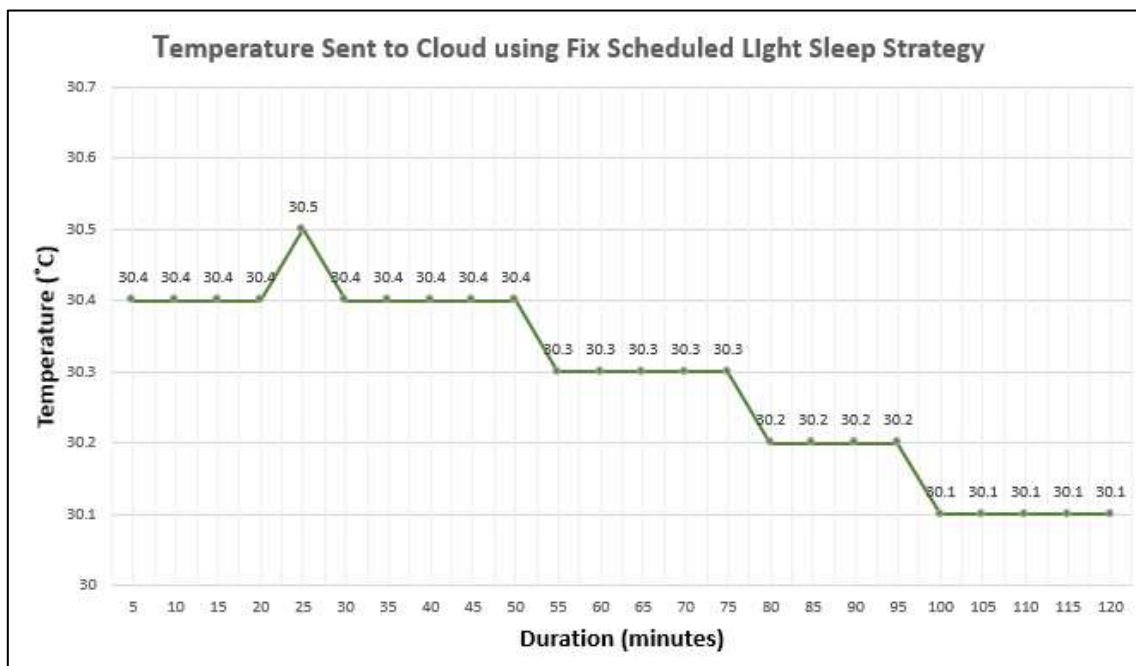


Figure 15. Temperature sent to cloud using fix scheduled light sleep on a microcontroller-based IoT system

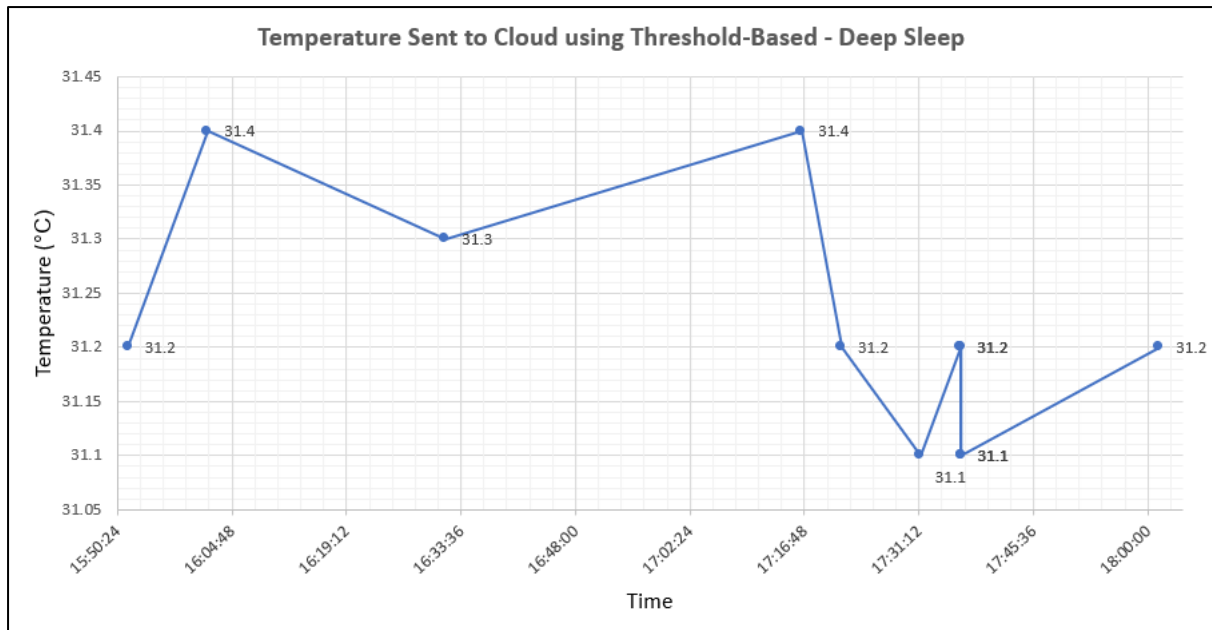


Figure 16. Temperature sent to cloud using threshold-based- deep sleep on a microcontroller-based IoT system

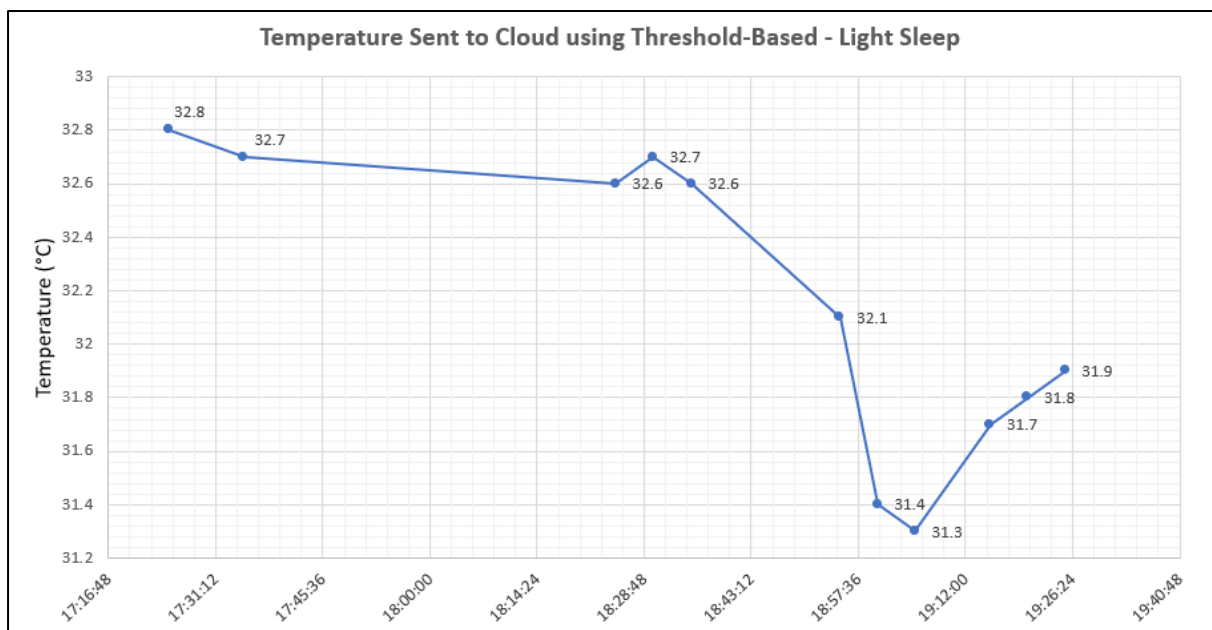


Figure 17. Temperature sent to cloud using threshold-based- light sleep on a microcontroller-based IoT system

In this way, data reliability is enhanced as redundant recordings are reduced and significant variations are given priority. In addition, this approach minimizes the possibility of missing crucial data because any substantial temperature change is recorded instantly while minute, insignificant changes are ignored. In terms of energy efficiency, the threshold approach is much better than the other two approaches by allowing the device to remain in the sleep mode for longer periods when the temperature is stable. This achieves a tremendous power saving, and thus it is the most suitable method for battery-powered IoT devices.

5. CONCLUSION

This study filled the gap by regulating data transmission and sleep behaviour based on meaningful changes in sensed temperature, aligning well with autonomous IoT deployments. Rather than most of previous studies only focus on maximizing energy utilization from weak or intermittent sources and event transmission without sleep and also the big system such as sensor nodes sleep. This study also confirms that adaptive sleep mechanisms which is threshold-based deep sleep significantly enhance the energy efficiency and data trade off of microcontroller-based IoT devices. Threshold-based deep sleep is identified as the most energy-conserving approach among the power modes tested, with 78.24% of energy savings while providing assured data delivery. Although fixed scheduled sleep mechanisms also provide significant power savings, they have a potential of losing critical data and hence are less desirable for sensitive or real-time IoT applications. Threshold-based light sleep provides a balanced approach, achieving significant energy saving with no appreciable loss of data but threshold-based deep sleep does more power saving. The findings establish that the implementation of adaptive threshold-based sleep mechanisms plays a vital role in optimizing power consumption in IoT devices with stable data communication. The research can be further extended to include machine learning algorithms in dynamically adapting sleep mechanisms based on real-time data traffic and environmental conditions for further efficiency and reliability of IoT deployments.

ACKNOWLEDGEMENTS

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

- [1] H. Ko, H. Choi, and S. Pack, "Restoration-Aware Sleep Scheduling Framework in Energy Harvesting Internet of Things: A Deep Reinforcement Learning Approach," *IEEE Trans. Sustain. Comput.*, vol. 10, no. 1, pp. 190–198, 2025, doi: 10.1109/TSUSC.2024.3442918.
- [2] S. Gupta *et al.*, "Intelligent resource optimization for scalable and energy-efficient heterogeneous IoT devices," *Multimed. Tools Appl.*, 2024, doi: 10.1007/s11042-024-18176-1.
- [3] G. Mahalakshmi, S. Ramalingam, and A. Manikandan, "An energy efficient data fault prediction based clustering and routing protocol using hybrid ASSO with MERNN in wireless sensor network," *Telecommun. Syst.*, vol. 86, no. 1, 2024, doi: 10.1007/s11235-024-01109-6.
- [4] S. Kaleem, A. Sohail, M. Babar, A. Ahmad, and M. U. Tariq, "A hybrid model for energy-efficient Green Internet of Things enabled intelligent transportation systems using federated learning," *Internet of Things (Netherlands)*, vol. 25, 2024, doi: 10.1016/j.iot.2023.101038.
- [5] X. Bai, B. Li, I. Ullah, Z. Wu, S. Basheer, and A. K. Bashir, "Energy-efficient routing for IoT-enabled multi-truck multi-drone pickup and delivery systems," *Appl. Energy*, vol. 400, p. 126546, 2025.
- [6] W. Li, R. Tang, S. Wang, and Z. Zheng, "An optimal design method for communication topology of wireless sensor networks to implement fully distributed optimal control in IoT-enabled smart buildings," *Appl. Energy*, vol. 349, 2023, doi: 10.1016/j.apenergy.2023.121539.
- [7] A. K. Sikder, G. Petracca, H. Aksu, T. Jaeger, and A. S. Uluagac, "A survey on sensor-based threats to internet-of-things (iot) devices and applications," *arXiv Prepr. arXiv1802.02041*, 2018.
- [8] S. M. Zainol, S. Daud, R. B. Ahmad, and M. H. Imran, "A Comprehensive Systematic Review: Sleep Strategy on Microcontroller for Power Management," *J. Adv. Res. Micro Nano Eng.*, vol. 23, no. 1, pp. 91–105, 2024, doi: 10.37934/armne.23.1.91105.
- [9] D. Roggen, R. Cobden, A. Pouryazdan, and M. Zeeshan, "Wearable FPGA Platform for Accelerated DSP and AI Applications," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events, PerCom Workshops 2022*, 2022. doi: 10.1109/PerComWorkshops53856.2022.9767398.
- [10] M. Carminati and G. Scandurra, "Impact and trends in embedding field programmable gate arrays

- and microcontrollers in scientific instrumentation," *Rev. Sci. Instrum.*, vol. 92, no. 9, 2021, doi: 10.1063/5.0050999.
- [11] A. Mellit and S. Kalogirou, "Internet of things (IoT) and embedded systems for photovoltaic systems," in *Handbook of Artificial Intelligence Techniques in Photovoltaic Systems*, 2022. doi: 10.1016/b978-0-12-820641-6.00008-9.
- [12] U. Nations, "SDG 7." Accessed: May 27, 2024. [Online]. Available: <https://sdgs.un.org/goals/goal7>
- [13] H. Wu, C. Chen, and K. Weng, "An energy-efficient strategy for microcontrollers," *Appl. Sci.*, vol. 11, no. 6, 2021, doi: 10.3390/app11062581.
- [14] R. Cheour, S. Khriji, M. Abid, and O. Kanoun, "Microcontrollers for IoT: Optimizations, Computing Paradigms, and Future Directions," in *IEEE World Forum on Internet of Things, WF-IoT 2020 - Symposium Proceedings*, 2020. doi: 10.1109/WF-IoT48130.2020.9221219.
- [15] A. M. Abdal-Kadhim and K. S. Leong, "Electrical power flow of typical wireless sensor node based on energy harvesting approach," *Int. J. Electron. Lett.*, vol. 8, no. 1, pp. 17–27, 2020, doi: 10.1080/21681724.2018.1545255.
- [16] M. Szymczyk and P. Augustyniak, "Selected Energy Consumption Aspects of Sensor Data Transmission in Distributed Multi-Microcontroller Embedded Systems," *Electronics*, vol. 11, no. 6, 2022. doi: 10.3390/electronics11060848.
- [17] T. Kim, S. Lee, H. Choi, H. S. Park, and J. Choi, "An Energy-Efficient Multi-Level Sleep Strategy for Periodic Uplink Transmission in Industrial Private 5G Networks," *Sensors*, vol. 23, no. 22, 2023, doi: 10.3390/s23229070.
- [18] Y. Prabowo, S. Broto, T. W. Winsuadji, and Siswanto, "Analisa Power Mode ESP32 Untuk Catu Daya Pada Sistem Berbasis IoT," *Pros. Semin. Nas. Sist. Inf. dan Teknol.*, vol. 6(1), 2022, [Online]. Available: <https://seminar.iaii.or.id/index.php/SISFOTEK/article/download/351/298/541>
- [19] E. H. Currie, "Microcontroller subsystems," in *Mixed-Signal Embedded Systems Design: A Hands-on Guide to the Cypress PSoC*, Springer, 2021, pp. 35–97. doi: https://doi.org/10.1007/978-3-030-70312-7_2.
- [20] PCBOnline, "Top 6 Popular MCU Microcontrollers and Their Comparison." Accessed: Mar. 12, 2024. [Online]. Available: <https://www.pcbonline.com/blog/top-6-popular-microcontrollers.html>
- [21] M. Samiullah, M. Z. Irfan, and A. Rafique, "Microcontrollers: A Comprehensive Overview and Comparative Analysis of Diverse Types," 2023, doi: <https://doi.org/10.31224/3228>.
- [22] K. H. Kim and H.-D. Kim, "Deep Sleep Mode Based NodeMCU-Enabled Humidity Sensor Nodes Monitoring for Low-Power IoT," *Trans. Electr. Electron. Mater.*, vol. 21, no. 6, pp. 617–620, 2020, doi: 10.1007/s42341-020-00236-6.
- [23] E. L. Alvanaki, M. Katsaragakis, D. Masouros, S. Xydis, and D. Soudris, "Decoupled Access-Execute enabled DVFS for tinyML deployments on STM32 microcontrollers," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2024, pp. 1–6. doi: <https://doi.org/10.23919/DAT58400.2024.10546540>.
- [24] Y. Liu and Y. Cheng, "Intelligent Wardrobe System Design Based on the Internet of Things," in *2024 4th International Symposium on Computer Technology and Information Science (ISCTIS)*, IEEE, 2024, pp. 351–355. doi: <https://doi.org/10.1109/ISCTIS63324.2024.10699229>.
- [25] E. Ripoll-Vercellone *et al.*, "A miniature and intelligent Low-Power in situ wireless monitoring system for automotive wheel alignment," *Wirel. Pers. Commun.*, vol. 19, no. 6, pp. 1537–1540, 2023, doi: 10.1109/JSSC.2021.3063134.
- [26] M. A. Syahmi Md Dzahir and K. Seng Chia, "Evaluating the Energy Consumption of ESP32 Microcontroller for Real-Time MQTT IoT-Based Monitoring System," in *2023 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies, 3ICT 2023*, 2023, pp. 255–261. doi: 10.1109/3ICT60104.2023.10391358.
- [27] J. D. Caneda and R. A. C. O. Calimpusan, "Design and Development of IoT-based SMART Monitoring System for Hydro-Powered Generator," in *8th International Conference on Engineering and Emerging Technologies, ICEET 2022*, 2022. doi: 10.1109/ICEET56468.2022.10007109.
- [28] S. Q. Hadi and M. A. Ali, "ENERGY-EFFICIENT SMART MEDICAL BRACELET FOR ALZHEIMER'S PATIENT MONITORING BASED WIRELESS COMMUNICATION SYSTEM," 2025, doi: <https://doi.org/10.26782/jmcms.2025.01.00004>.
- [29] Z. Qian, D. R. Thinesh, B. Sivaneasan, K. T. Tan, K. J. Tseng, and P. Chakrabarti, "Advancing Low-Power Self-Sustaining IoT Sensors Through CdTe PV-Driven LoRa Communication," in *IECON Proceedings (Industrial Electronics Conference)*, 2023. doi: 10.1109/IECON51785.2023.10312654.
- [30] M. Walton and J. Woods, "Intelligent control of micro power – Immortal machine," *Nano Energy*, vol. 72, 2020, doi: 10.1016/j.nanoen.2020.104699.

- [31] S. Kondratiev, D. Koshutina, O. Liubomska, and I. Baskov, "Research on Energy Efficiency of Wi-Fi IoT Systems on Renesas DA16200 Platform," in *Proceedings of International Conference on Applied Innovation in IT*, 2024, pp. 143–148. [Online]. Available: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85190992523&partnerID=40&md5=81929cb6f37b84d1c5c640e124e4d77c>
- [32] A. S. Priambodo and A. P. Nugroho, "Design & Implementation of Solar Powered Automatic Weather Station based on ESP32 and GPRS Module," in *Journal of Physics: Conference Series*, 2021. doi: 10.1088/1742-6596/1737/1/012009.
- [33] N. Vijayaregunathan, V. Periyasamy, and A. Munimathan, "Harvesting hydraulic energy for self-sustainable IoT flow measurement devices," *Energy Sources, Part A Recover. Util. Environ. Eff.*, vol. 46, no. 1, pp. 5094–5111, 2024, doi: 10.1080/15567036.2024.2334922.
- [34] A. S. Okoro, S. Yousef, and A. Qashou, "Investigating Gesture Control of Robotic Arm via Lora Technology for Smart Cities," in *Advanced Sciences and Technologies for Security Applications*, vol. Part F1307, 2023, pp. 71–101. doi: 10.1007/978-3-031-33631-7_3.
- [35] W. M. Organization, *Guide to meteorological instruments and methods of observation*. Geneva, Switzerland: World Meteorological Organization, 2018. [Online]. Available: <https://library.wmo.int/records/item/41650-guide-to-instruments-and-methods-of-observation>
- [36] Cytron, "5000mAh LiPo Battery." Accessed: Aug. 23, 2025. [Online]. Available: https://my.cytron.io/p-3.7v-5000mah-li-ion-battery-with-jst-ph2.0?r=1&gad_source=1&gad_campaignid=17507688159&gbraid=0AAAAADh9ZNDF1l-0qBm7iQKoXSDyqxTdR&gclid=Cj0KCQjwzaXFBhDIARIsAFPv-u8rZ8NvbTBD1ViOS4mdCdwSOuBPMdnsouqCbhFwgJcfyiTvW5aRL-kaAnL2EALw_wcB
- [37] A. Khalid *et al.*, "Extended Lifetime of IoT Applications using Energy Saving Schemes," in *2022 International Conference on Future Trends in Smart Communities, ICFTSC 2022*, 2022, pp. 93–97. doi: 10.1109/ICFTSC57269.2022.10040064.