# Energy-Efficient Software Development: A Multi-dimensional Empirical Analysis of Stack Overflow

Bihui Jin
University of Waterloo
Waterloo, Canada
bihui.jin@uwaterloo.ca

Heng Li
Polytechnique Montréal
Montréal, Canada
heng.li@polymtl.ca

Pengyu Nie
University of Waterloo
Waterloo, Canada
pynie@uwaterloo.ca

Ying Zou
Queen's University
Kingston, Canada
ying.zou@queensu.ca

## Abstract

Energy consumption of software applications has emerged as a critical concern for developers to contemplate in their daily development processes. Previous studies have surveyed a limited number of developers to understand their viewpoints on energy consumption. We complement these studies by analyzing a meticulously curated dataset of 1,193 Stack Overflow (SO) questions concerning energy consumption. These questions reflect real-world energy-related challenges practitioners face during development. To understand practitioners' perceptions, we investigate the intentions behind these questions, semantic topics, and associated technologies (e.g., programming languages). Our results reveal that: (i) the most prevalent energy consumption topic is about balancing *Positioning* usage; (ii) efficiently handling data is particularly challenging, with these questions having the longest response times; (iii) practitioners primarily ask questions to understand a concept or API related to energy consumption; and (iv) practitioners are concerned about energy consumption across multiple levels—hardware, operating systems, and programming languages—during energy efficient software development. Our findings raise awareness about energy consumption's impact on software development. We also derive actionable implications for energy optimization at different levels (e.g., optimizing API usage or hardware accesses) during energy-aware software development.

## 1 Introduction

Technological advancement has led to increased energy consumption [57]. Particularly, in the era of artificial intelligence, the energy consumption of Information and Communication Technology (ICT) is projected to surge by 822.79% from 2001 to 2030, reaching 17959.11 TWh per year [109]. In 2020, ICT accounted for up to 7% of global electricity use [5] and 2.1% to 3.9% of global greenhouse gas emissions [41]. By 2030, data centers alone are projected to consume 10% of the world's electricity [84].

Software plays a crucial role in ICT energy consumption [106]. Energy-related issues affect every phase of the software lifecycle, including design, implementation, testing, and maintenance [28]. Energy-efficient software development can enable energy savings, extend battery life, and enhance user experience [79], helping mitigate the growing trend of global electricity consumption. Practitioners now recognize energy efficiency as an important non-trivial property of software and have demonstrated willingness to learn about energy issues in software development [63].

A few empirical studies have been conducted to understand practitioners' perceptions of energy consumption. For instance, prior studies [63, 79] conduct online surveys with 122 and 464 practitioners, respectively, who self-identify as experienced developers and testers across various application domains. The studies find that 65% of practitioners on Reddit recognize energy usage as a crucial factor in software quality [79], and many are willing to sacrifice other requirements to reduce energy consumption [63]. However, these studies only survey opinions without examining specific barriers practitioners face during their development of energy-efficient software. The most similar work to ours is by Pinto et al. [83], who study 325 Stack Overflow (SO) questions from 2008 to 2013 and identify five primary themes of energy-related questions (e.g., measurements and code design). With the rapid evolution of technologies, such as IoT devices, practitioners likely face different challenges today than a decade ago, making the previous studies less relevant to current practitioners.

We follow the same methodology outlined in Pinto et al. [83] to collect energy-related posts. Through keyword search followed by manual verifications, we curated a dataset of 1,193 energy-related SO questions, which contains **larger and more recent** questions compared to the work by Pinto et al. (325 questions by 2013). Inspired by recent work on analyzing SO posts [16, 90, 108, 111], we perform **LDA topic modeling instead of thematic analysis** to obtain low-level and actionable semantic topics (e.g., *positioning* or *data transmission*) in these questions. In addition, we investigate **new dimensions** of these energy-related questions, including their intentions (e.g., to understand a concept) and the associated technologies (e.g., an operating system). Furthermore, we analyze the **evolution** of these energy-related questions over the years.

We formulate three research questions to guide our study:

- **RQ1: What are the topics of the energy-related questions and their difficulty?**
  The energy-related questions asked by practitioners on SO are diverse in nature. To understand the characteristics of SO questions, we apply topic modeling to analyze their titles and bodies and group them into topics. We identify eight recurring topics related to energy issues: *Positioning*, *Computing Resource*, *Mobile Device*, *Sensor Timing*, *Polling*, *Datum Handling*, *Data Transmission*, and *Thread*. Questions about `GPS` (e.g., location tracking) are the most common, while questions about `Polling` are the most challenging, requiring the longest time to receive community responses.

- **RQ2: What are the intentions behind energy-efficient development questions?**
  Practitioners have diverse intentions when posting energy-related questions, such as finding the root causes of rising energy consumption. Understanding the intentions provides insights into the challenges associated with energy consumption in software

development [3]. We observe that nearly half of the energy-related questions are mainly concept-oriented, covering background knowledge of APIs and programming concepts like design patterns. These questions are frequently accompanied by `API USAGE` questions about learning how to use an API. The third most common intention is `DISCREPANCY`, where practitioners struggle to resolve discrepancies between the profiling results and their expectations regarding energy consumption.

- **RQ3: What technologies are concerned in energy-efficient development?**
  Energy-related questions span several levels of technologies, including operating systems (OS), programming languages (PL), and hardware (HW), usually identified by tags (e.g., `android`, `java`, and `raspberry-pi`). We analyze the tags attached to each SO post to identify the categories of energy-related concerns. Through our analysis of tag categories, we provide suggestions for allocating technical support to help practitioners solve energy-related problems.

This work aims to understand practitioners' challenges in energy-aware software development through examining SO questions where practitioners express their challenges. An SO question carries information from several mutually complementary perspectives about practitioners' challenges: the semantical meaning (topic) of the question (RQ1), the intentions of the question (RQ2), and the associated technology (RQ3). Therefore, we organize our research questions to provide a systematic investigation of energy-related SO questions. The main contributions of this study include:

- We provide a carefully curated dataset of 1,193 SO posts (spanning from 2008 - 2024) that conveys energy-related concerns, analyzed using a combination of manual, topic modeling, and LLM-based efforts. Our public dataset will be a valuable resource for future research on energy-efficient software development.
- Our approach and empirical findings gain insights on energy-related SO posts in three dimensions: topics, intentions, and associated technologies, extending Pinto et al.'s analysis that focuses only on the topics of 325 posts (from 2008–2013). We also discuss the potential root causes leading to developers' questions.
- Our analyses of topics, intentions and associated technological shifts provide insights on the common types of questions developers ask (e.g., questions about API usages) and the direction for support in application level and specialized device contexts.
- We discuss actionable implications for energy optimization at API-level and low-level OS- and hardware-level energy optimizations to improve effectiveness in energy-efficient software development.

Our dataset and experiment scripts are uploaded as supplementary materials along with the submission and will be open-sourced.

# 2 Related Work

## 2.1 Energy-Efficient Software Development

Cruz and Abreu [32] create a catalog of 22 design patterns for improving mobile app energy efficiency based on analyzing 1,027 Android and 756 iOS apps. They find Android developers are more aware of energy issues than iOS developers and provide actionable guidelines for improving energy consumption. Bao et al. [15] study 468 power management commits from 154 Android apps, categorizing them into six activities, including power adaptation, consumption improvement, and wake lock optimization. Their findings highlight how different app categories prioritize different power management strategies. Moura et al. [70] identify 12 themes in 371 energy-aware commits across 317 applications. They find that developers struggle to predict the energy impact of code changes and that energy-saving techniques may compromise other quality attributes. Our work complements these codebase and commit-based studies by examining the questions developers ask when facing energy-related challenges, providing insights into the knowledge gaps that exist in practice.

Pang et al. [79] survey over 100 practitioners and find that 86% lacked awareness of energy efficiency best practices and could not identify the causes of high energy consumption. Their study reveals that practitioners struggle to implement energy-efficient solutions due to insufficient information, tools, and infrastructure. Manotas et al. [63] analyze survey responses from 464 experienced practitioners at major tech companies and find that developers are willing to sacrifice other requirements to reduce energy usage but struggle to diagnose energy issues. While these survey-based studies provide valuable insights, our work complements them by analyzing real-world questions from Stack Overflow, revealing the specific challenges practitioners face during actual development.

Pinto et al. [83] analyze 325 Stack Overflow questions from 2008 to 2013 related to energy consumption, finding that such questions increased yearly and that mobile development accounted for 25% of the tags. Our work extends this research with a larger dataset (1,193 questions from 2008 to 2024), providing an updated view of practitioners' challenges. Unlike Pinto et al. [83], who focus primarily on question topics, we analyze posts not only by topics but also along two additional dimensions (intent-based and technology-based) to provide more comprehensive insights into energy-efficient software development challenges.

## 2.2 Analysis of Developer Forum Posts

Beyer et al. [19] inspect and classify 1,000 Android-related SO posts into seven intention categories: `API Usage`, `Conceptual`, `Discrepancy`, `Errors`, `Review`, `API change`, and `Learning`. They find `API Usage` is the most common intention, followed by `Discrepancy` and `Conceptual`. We adopt this classification scheme in our study but find that for energy-related questions, `Conceptual` questions are most common, highlighting the knowledge gap in energy-efficient development.

Previous studies have explored trends and topics in developer discussions using both temporal analysis and topic modeling. Venkatesh et al. [105] analyze 92,471 discussions about 32 Web APIs from developer forums, identifying five dominant patterns in how topics change over time. Their temporal analysis approach inspires our examination of how energy-related concerns have evolved over the years. In addition, several studies have applied topic modeling to analyze Stack Overflow posts: Wan et al. [108] cluster blockchain discussions; Rosen and Shihab [90] summarize mobile-related questions; Yang et al. [111] cluster security-related questions; and Barua

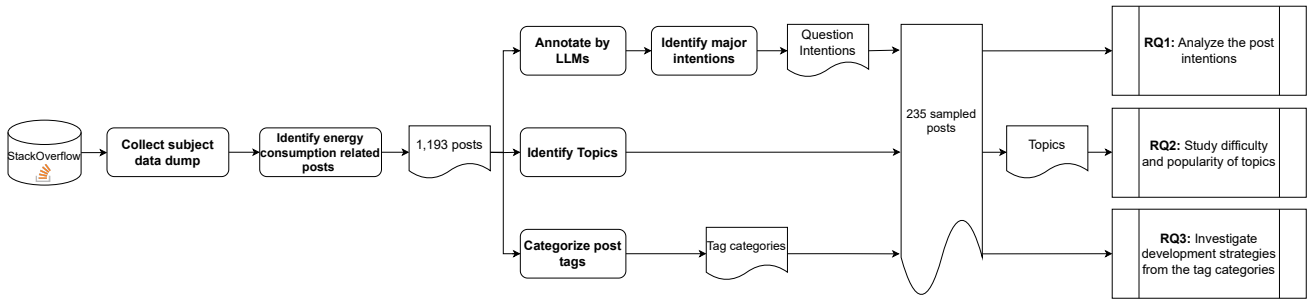


**Figure 1: An overview of our experiment design for collecting, filtering, and processing SO posts.**

et al. [16] investigate primary discussion topics and trends over time. Following these approaches, we apply LDA [21], a mature topic modeling technique, to identify eight recurring topics in energy-related questions, including *Positioning*, *Computing Resource*, and *Datum Handling*.

Our work uniquely combines these approaches to analyze energy-related SO posts. We adopt Beyer et al. [19]'s intention classification, use LDA for topic modeling like Barua et al. [16], Rosen and Shihab [90], Wan et al. [108], and analyze energy-related technologies to provide a multi-dimensional understanding of practitioners' energy-related concerns. This comprehensive approach reveals insights into the challenges developers face when dealing with energy efficiency that are not captured in previous studies.

## 3 Study Design

In this section, we present our data collection, pre-processing, and analysis steps. Figure 1 shows an overview of our approach.

### 3.1 Subject Dataset

We download the latest Stack Overflow official data dump[1] at the time of conducting this work, which contains 23,709,404 posts from January 2008 to December 2024. Figure 2 shows a sample SO post [6]. We extract the following information from each post:

- Title, which summarizes the question being asked or the topic being discussed;
- View count, showing how many times the post has been viewed;
- Body, containing details of the question or discussion;
- Tags, which are labels that have been assigned to the post according to the areas of the question;
- Question score, assigned by users based on its usefulness and clarity;
- Question creation time, when the question was first posted;
- Comments (to the question), including suggestions (that are not sufficient to be answers), clarifications, and relevant information added by other users;
- Comment creation time (of the first comment if it exists), which indicates how quickly it is to receive initial help to the question;
- Answers, which are the solutions to the question; an answer can be marked as "accepted" if it is verified to solve the question;
- Answer score (of the accepted answer if it exists), assigned by users based on its correctness and relevance;
- Answer creation time (of the accepted answer if it exists), which indicates how quickly the question is solved.

[1]https://archive.org/details/stackexchange_20241231

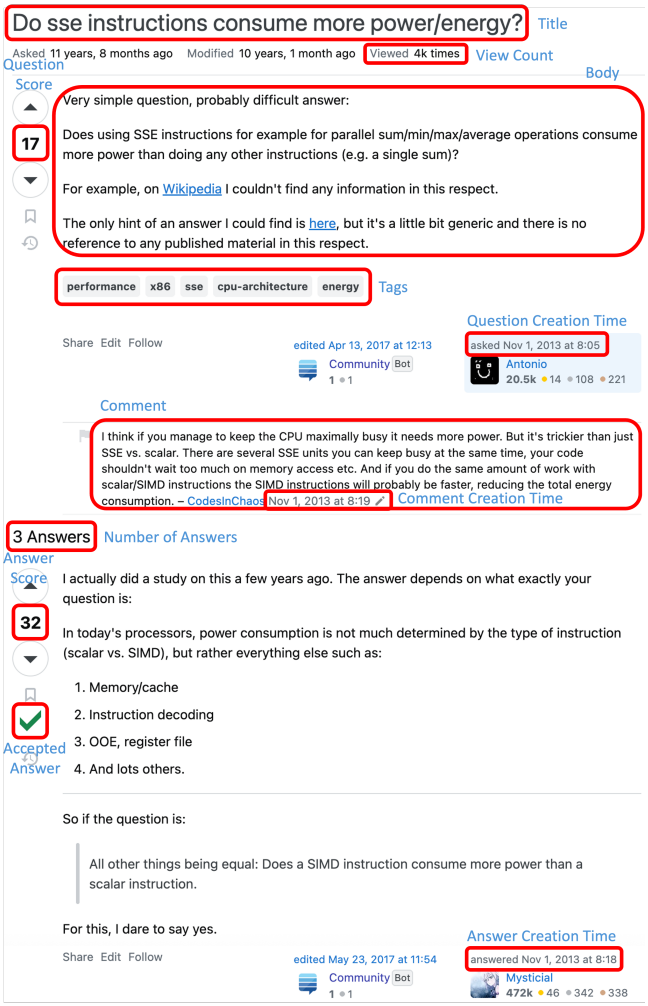


**Figure 2: An annotated screenshot of a Stack Overflow post.**

### 3.2 Identifying Energy-Related SO Posts

We use a two-phase approach to identify SO posts related to energy consumption: an automated filtering phase to search keywords related to energy consumption in the question title and body, followed by a manual confirmation phase.

We first exclude code snippets from the posts surrounded by <code></code>html tags, since they may contain incidental matches that do not reflect the intent of the questioner. Then, we search for energy-related keywords using twelve wildcards. The original

eight keywords from Pinto et al. are proposed in 2014, whereas new energy-related keywords may surface over the years. Thus, we manually inspect statistically significant samples and summarize four new keywords following the same methodology as outlined in Pinto et al., resulting in twelve keywords in total: %energy consum%, %power consum%, %energy efficien%, %power efficien%, %energy sav%, %power sav%, %energy us%, %power us%, %energy econ%, %power econ%, %sav% energy%, %sav% power%. The character '%' enables fuzzy matching using the stems of the keywords. This search yields 2,629 potential posts.

We manually review all 2,629 posts to eliminate false positives—posts that mention energy-related keywords but do not actually seek to address energy consumption issues. As an example, the questioner may mention "Energy Consumption" in the title but not seek for energy-efficiency solutions [69]. The labelling process involves two phases:

(1) Two annotators (both Software Engineering PhDs, including the first author) independently label the first half of the questions, following a closed coding procedure [92]. The Cohen's Kappa score [30] is 0.55 after this step, indicating a moderate level of agreement. Then, the two annotators discuss the labels of the first half and achieve consensus for each question, establishing a common understanding.
(2) The two annotators then independently label the second half of the questions. The Cohen's Kappa score is 0.69 for the second half, indicating a substantial level of agreement. Finally, the two annotators discuss the labels of the second half and achieve consensus for each question.

If a conflict persists during discussion, a third annotator (who is a Software Engineering Professor and also an author of this paper) steps in to resolve it. This process yields our final dataset of 1,193 posts with 1,478 answers.

### 3.3 Pre-processing Posts

We pre-process the posts to prepare them for topic modeling, following these steps:

- **Title and Body Composition**. We combine each post's title (which is a succinct description of intention) with its body (which elaborates the details) to provide complete context for the model.
- **Tokenization, Stemming, and Lemmatization**. We process the text using the `spaCy` model `en_core_web_trf` [52, 58], which tokenizes the text and also performs stemming (reducing words to their basic form) and lemmatization (normalizing word forms, such as converting past tense verbs to present tense).
- **Stop Words Removal**. Not all words contribute significantly to the overall meaning of the sentence. We remove common stop words, such as "the", "of", "an", "by", "is", and "what". We also inspect the top 100 most frequent words in our dataset and filter out additional stop words that are irrelevant for our study, such as "like", "thank", etc.
- **Bag of Words**. We create a Bag of Words representation using `Gensim`'s `doc2bow` function [88], which maps each document to a dictionary from token ids to counts, preserving word frequency while disregarding syntax and word order.

**Table 1: The LDA dominant topics in SO energy-related posts.**

| Topic | Keywords |
|---|---|
| Positioning | energy, battery, location, usage (power, app, consumption, android, use, application) |
| Computing Resource | cpu, gpu, core (power, consumption, energy, time, usage, device, code) |
| Mobile Device | power, consumption, android, device, mode, application save, phone (code, app) |
| Sensor Timing | sensor, app, time, find, run (power, consumption, use, user, code) |
| Polling | code, use, clock, sleep, module, wake (power, consumption, mode, device) |
| Datum Handling | memory, kwh, try, datum, image (energy, power, consumption, time) |
| Data Transmission | file, server, message (energy, device, app, power, consumption, datum, try) |
| Thread | user, thread (power, code, use, mode, application, try, consumption, datum) |

## 4 Results and Findings

In this section, we describe the motivation, the approach, and our findings for each of our three research questions.

### 4.1 RQ1: What are the topics of the energy-related issues and their difficulty?

*4.1.1 Motivation.* The energy-related SO questions raised by practitioners cover diverse topics. Understanding the topics helps uncover their predominant concerns. We use topic modeling to cluster similar energy-related concerns and analyze each topic's popularity and difficulty in receiving community support. This provides insights into the challenges developers face when working on energy-efficient software.

*4.1.2 Approach.* To discern question patterns, we perform the following steps:

**Topic Modeling**. To study the most significant concerns regarding energy consumption, we utilize topic modeling to group the related posts into a topic. Specifically, we apply Latent Dirichlet Allocation (LDA) [20] from the `Gensim` package [88]. LDA is a generative statistical model that automatically generates common topics based on the probability of the distribution of discrete words in a corpus, without requiring predefined taxonomies [21, 49]. LDA topic modeling is widely used in the analysis of SO posts [16, 90, 108, 111]. Instead of manual coding, we use LDA to uncover latent topic structures and achieve unbiased and reliable clustering results. LDA generates topics as collections of keywords with significance percentages. For example, in the set (0.023*cpu + 0.023*power + 0.016*consumption + 0.012*gpu...), "cpu" is the most significant word. Keywords can appear in multiple topics with different significance levels.

However, LDA requires a hyperparameter of the number of topics, the optimal number of which is unknown before running the analysis. We train the model for 100 iterations with varying numbers of topics (1–100) and using both *Jaccard similarity* and *topic coherence* metrics to determine the optimal number of topics. For other parameters, such as $\alpha$, $\beta$, and *passes*, we follow best practices suggested by prior work [42, 107].

Jaccard similarity measures the similarity between two adjacent topics using Equation (1) [1, 47]:

**Table 2: Definitions of the derived energy consumption related topics.**

| Topic | Definition (D) - Quote (Q) | Freq |
|---|---|---|
| Positioning | **D**: Questions that seek to improve the efficiency of location updates, reduce GPS-related battery drain, and balance real-time tracking with energy savings. **Q**: "*How to energy efficiently track GPS/[...]*" [101] | 325 (27.2%) |
| Computing Resource | **D**: Questions that concern about perceiving the use of computing resources and the efficiency of resource allocation. **Q**: "*Is there a way to use processors/[...] more effectively?*" [86] | 217 (18.2%) |
| Mobile Device | **D**: Questions about balancing power states, or device-specific features on mobile devices. **Q**: "*How to sense phone's mode/[...] to save power consumption?*" [10] | 165 (13.8%) |
| Sensor Timing | **D**: Questions that focus on optimizing the scheduling and continuous operation of sensors within an app. **Q**: "*Why [...] every second/minute from a service would consume so much power?*" [74] | 120 (10.1%) |
| Polling | **D**: Questions about concerns about overhead due to continuously "polling" hardware or software flags. **Q**: "*How to reduce instruction count/[...] in code/clock sources?*" [81] | 110 (9.2%) |
| Datum Handling | **D**: Questions related to storing, selecting, or allocating data efficiently. **Q**: "*Is it optimized in terms of processing/allocating/sending/storing data [...]?*" [60] | 97 (8.1%) |
| Data Transmission | **D**: Questions that revolve around capturing, sending, or synchronizing data or messages between devices. **Q**: "*How to send data to a server [...] with less energy consumption?*" [50] | 94 (7.9%) |
| Thread | **D**: Questions that involve synchronizing states between threads or processes. **Q**: "*How to manage thread/[...] execution efficiently?*" [93] | 65 (5.4%) |

$$J(w_i, w_j) = \frac{(w_i \cap w_j)}{(w_i \cup w_j)} \tag{1}$$

where $i$ and $j$ represent different topics, and $w_i$ and $w_j$ are sets of keywords in each topic. A Jaccard similarity of 1 indicates topics share all keywords, while 0 means they have no common keyword.

We use Jaccard similarity to maximize the topic divergence and minimize the degree of topic overlap, defined by Equation (2):

$$Sim(LDA) = mean(\{J(w_i,\ w_j)\ \mid\ \forall w_i, w_j \in LDA\}) \tag{2}$$

where $LDA$ is the computed LDA model, $w_i$ and $w_j$ represent each pair of topics in the model.

Topic coherence measures the degree of semantic similarity of words within each topic [72]. We compute topic coherence `Coh(LDA, corpus)` using `Gensim`'s `CoherenceModel` [91], where `LDA` is the LDA model and `corpus` is the pre-processed texts.

To determine the optimal number of topics, we maximize intra-topic coherence while minimizing inter-topic similarity, as shown in the optimization function in Equation (3):

$$\arg\max_i f(i) = Coh(LDA(i),\ corpus) - Sim(LDA(i)) \tag{3}$$

where $i$ is the number of topics, $Coh$ measures topic coherence between the LDA model with $i$ topics and the corpus, and $Sim(i)$ is the average topic overlap computed by Equation (2).

**Topic Assignment**. After performing 100 iterations with 1-100 topics, we determine that the optimal number of topics is 8. To name the topics, we randomly select 235 posts with accepted answers (at most 30 from each topic, ensuring a 97% confidence level with a 5% margin of error) and manually summarize the keywords into a high-level description. The resulting topics, keywords, and our manually determined descriptive names are shown in Table 1. LDA assigns topic membership probabilities to each post, indicating how likely a post belongs to each topic. We assign each post to its most probable topic. For example, post 61882 [43] has the likelihood of 92% being in topic 7 (*Mobile Device*), 7.3% being in topic 6 (*Polling*), etc., thus it is assigned to topic 7 (*Mobile Device*).

To further understand energy-related posts, we conduct a systematic analysis to each LDA topic, specifically: (1) computing their popularity; and (2) calculating difficulty-related metrics that represent how difficult it is to receive a satisfactory answer.

**Computing Popularity**. We measure relative topic popularity using Equation (4) proposed by Pinto et al. [83]:

$$\mathbb{P} = \mathbb{S} + \mathbb{A} + \mathbb{C} + \mathbb{V} \tag{4}$$

where $\mathbb{S}$ is the question score, $\mathbb{A}$ is the accepted answer score, $\mathbb{C}$ is the number of comments, and $\mathbb{V}$ is the normalized view count. $\mathbb{V}$ is computed using Equation (5):

$$\mathbb{V} = \frac{\texttt{QuestionViews}}{\texttt{TotalViews}} \tag{5}$$

where `QuestionViews` is the current question's view count, and `TotalViews` is the total view count of all questions in our dataset.

Note that although the original computation of popularity in Pinto et al. [83] includes the favorization score of questions (i.e., how many users favorized the question), this feature is deprecated on Stack Overflow and has been removed in the latest data dump we obtain. Thus we exclude it from our computation.

All of $\mathbb{S}$, $\mathbb{A}$, $\mathbb{C}$, and $\mathbb{V}$ are normalized using min-max normalization in Equation (6), to mitigate the problem that the different metrics can have different scales, or one metric dominates the calculation of popularity:

$$X_{norm} = \frac{x - min(x)}{max(x) - min(x)} \tag{6}$$

where $x$ is the original value, and $min(x)$ and $max(x)$ are the minimum and maximum values in our dataset.

**Calculating Difficulty-Related Metrics**. We calculate a set of metrics to characterize the level of difficulty of the questions related to each topic. The difficulty metrics include the proportion of questions with an accepted answer and without an answers, as well as the median response time (in hours) for receiving the first answer and the first comment.

*4.1.3 Findings.* **We discern eight topics from the energy-related questions.** Table 2 lists the topics with their definitions, examples, and popularity. The most frequently discussed topics are *Positioning* (27.2% of questions), *Computing Resource* (18.2%), and *Mobile Device* (13.8%), followed by *Sensor Timing* (10.1%), *Polling* (9.2%), *Datum Handling* (8.1%), *Data Transmission* (7.9%), and *Thread* (5.4%). The

**Table 3: Common energy-consumption topics with cumulative normalized popularity score metric. The bold numbers highlight the most popular topic in terms of each metric.**

| Topic | P | S | A | C | V |
|---|---|---|---|---|---|
| **Positioning** | **61.5** | **26.5** | **8.2** | **20.5** | **6.4** |
| Computing Resource | 53.3 | 19.9 | 7.5 | 20.3 | 5.6 |
| Mobile Device | 31.9 | 13.4 | 4.5 | 9.8 | 4.1 |
| Sensor Timing | 22.1 | 9.4 | 2.4 | 8.5 | 1.8 |
| Polling | 21.9 | 8.5 | 2.4 | 8.8 | 2.2 |
| Datum Handling | 23.3 | 7.7 | 2.1 | 10.8 | 2.7 |
| Data Transmission | 17.4 | 6.7 | 1.9 | 7.4 | 1.3 |
| Thread | 13.6 | 5.7 | 2.0 | 4.4 | 1.5 |

* P represents the popularity of the question, S is the score of the question, A denotes the adopted answer score of the question, and C and V stand for comments and the normalized number of views, respectively.

**Table 4: Common energy-consumption topics with their metrics representing difficulties. The bold numbers highlight the most difficult topic in terms of each metric.**

| Topic | With an accepted answer | Without an answer | Receiving an answer (Hr) | Receiving a comment (Hr) |
|---|---|---|---|---|
| Positioning | 36.0% | 21.8% | 2.25 | 0.32 |
| Computing Resource | 39.6% | 18.9% | 3.42 | 0.27 |
| Mobile Device | 37.6% | 20.6% | 1.48 | 0.39 |
| Sensor Timing | 38.3% | 25.0% | 1.75 | 0.22 |
| Polling | 33.6% | 26.4% | **12.0** | 0.63 |
| Datum Handling | **28.9%** | **29.9%** | 9.08 | 0.46 |
| Data Transmission | 37.2% | 19.1% | 3.12 | 0.33 |
| Thread | 53.8% | 23.1% | 1.36 | **0.78** |
| Average | 38.1% | 23.1% | 4.31 | 0.42 |

alignment of frequency-based (Table 2) and popularity-based (Table 3) rankings shows that the topics with the most questions tend to accumulate the most engagement (simply counts posts), whereas popularity considers other metrics, such as views and up-votes, to capture overall importance.

**Questions related to *Positioning* receive the most attention from the SO community**, because the positioning services (such as GPS) directly impact battery life while being essential for many modern applications (e.g., navigation). According to Table 3, positioning-related questions rank highest in popularity. For instance, positioning services have long been recognized for their significant impact on energy consumption [114]. Our manual investigation also reveals the challenges in balancing the need of real-time position data and energy efficiency, e.g., practitioners ask questions about managing position update frequencies (post 26336225 [13], post 29616977 [56]) or time periods (post 23560949 [36]), and using alternative position tracking sensors to reduce power consumption (post 43596157 [39], post 10920904 [101]).

***Computing Resource* is also of high interest to practitioners** due to their critical role in system performance and efficiency. The increasing complexity of managing computing resources, particularly in cloud computing and virtualization environments, drives interest in this topic. Practitioners seek to optimize resource utilization to improve overall performance and energy efficiency, at application level (post 58156084 [86]), framework level (post 53577434 [85]), and hardware level (post 28003660 [110]). There is also a need to measure or estimate the energy consumption of computing resources (post 44228005 [68], post 4485153 [38]).

**Questions related to *Datum Handling* are the most challenging to receive satisfactory answers.** Table 4 shows that *Datum Handling* questions have the highest non-acceptance ratio (71.1%) and lowest accepted answer ratio (28.9%), with a median response time of 9.08 hours. In contrast, prior studies show better metrics for other domains: Rosen and Shihab [90] report a 30% non-acceptance ratio and 21-minute median response time for mobile development questions, while Bagherzadeh and Khatchadourian [12] report a 60.5% non-acceptance ratio and 3.3-hour median response time for big data questions. Our analysis of sampled posts reveals that this difficulty stems from mismatches between developers' expectations and the underlying complexities of data management in energy-sensitive contexts (e.g., post 1738515 [60], post 23617388 [22], post 54028927 [78], and post 8782922 [29]). The specialized domain knowledge required, OS-level abstractions, and lack of universal libraries contribute to higher non-acceptance rates and longer response times.

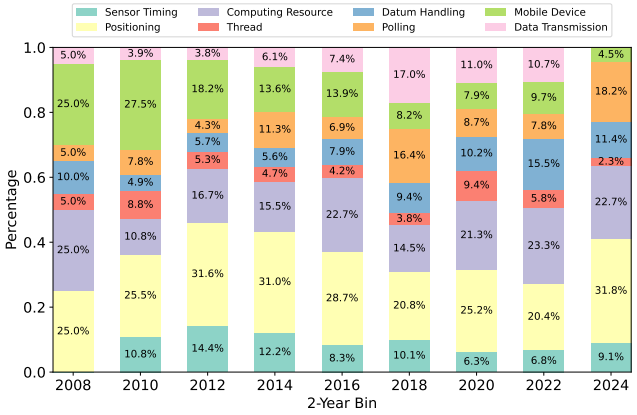


**Figure 3: Topic Distribution Every 2 Years.**

***Polling* questions take the longest time to receive responses.** Table 4 shows that polling-related questions have the longest median time of 12 hours before receiving an answer. Our manual analysis reveals that polling typically involves low-level hardware details such as microcontroller registers (post 50064364 [82]), clocks (post 27383269 [80]), and ADC sampling (post 53173447 [81]). About 62.1% of answers require in-depth knowledge of specific MCUs or peripherals, however, only a few community members may have that expertise. Additionally, broad questions requiring clarification further delay responses (post 28760898 [104], post 39804144 [65]). The complexity of polling issues—stemming from low-level hardware interactions, specialized domain knowledge, and lack of standards—leads to prolonged answer delays, lower acceptance rates, and more unresolved posts compared to other topics.

**Technological concerns shift from hardware limitations (e.g., *Mobile Device*) to data-oriented connective topics (e.g., *Data Transmission* and *Datum Handling*).** As shown in Figure 3, *Data Transmission* notably increases from about 5.0% in 2008 to a peak of 17.0% in 2018, and stabilizes at around 10-11%, which reflects growing concerns and developments in big data, cloud computing, IoT, and wireless communication technologies. *Datum Handling* significantly grows, especially from 2016 onwards, indicating increased attention toward data processing, analytics, and big data technologies, which peaks at 15.5% in 2022, reflecting the rapid adoption of data-intensive applications. *Positioning* remains consistently significant across all years, indicating its continuous importance in technology, maintaining around 25-32%. *Mobile Device* initially dominates but decreases over time, from 25.0% in 2008

to less than 10% after 2018, reflecting maturity in mobile technologies or shifting interest towards other emerging technologies.

Pinto et al. identify seven causes of energy-related questions (e.g., "faulty GPS behaviour") [83]. Among them, "unnecessary resource usage" and "excessive synchronization" are related to our topics *Computing Resource* (throttling idle cores) and *Thread* (avoiding busy-waits and lock overhead), separately; "faulty GPS behaviour" and "void polling" can be mapped our topics *Positioning* (minimizing GPS on-time or avoiding fine-grained location) and *Polling* (favoring interrupt-driven approaches), respectively. Beyond these overlaps, our study uncovers several new topics (*Mobile Device*, *Sensor Timing*, *Datum Handling*, and *Data Transmission*) not emphasized by Pinto et al., reflecting how the energy-efficient software landscape has evolved far beyond the CPU and GPS focus of 2013. Our eight topics not only capture evolving concerns, such as sensor batching and network I/O trade-offs, but also furnish actionable implications, such as optimizing *Datum Handling* at the OS/kernel, hardware, and data structure levels to directly control resource usage rather than merely tweaking high-level application code.

***Summary:*** *We derive eight common topics using LDA: Positioning, Computing Resource, Mobile Device, Sensor Timing, Polling, Datum Handling, and Thread. Our results indicate that questions related to Positioning are the most prevalent, reflecting practitioners' strong interest in optimizing battery performance for location-based services. Meanwhile, questions about Polling and Datum Handling are the most difficult to receive community support due to in-depth knowledge required. Moreover, our topic evolution analysis reveals a shift from hardware-centric concerns toward data-oriented topics. Our inspections suggest that providing more context-specific support (e.g., on how to efficiently handle positioning) could help practitioners overcome challenges in energy-efficient software development; our findings highlight these specific areas.*

## 4.2 RQ2: What are the intentions behind the energy-efficient development questions?

*4.2.1 Motivation.* Practitioners seek assistance on Stack Overflow with various intentions, from understanding concepts to troubleshooting API usages. The tags associated with the questions usually do not reveal why questions are asked. By categorizing these intentions across different topics (identified in RQ1), we can uncover the most common challenges in energy-efficient software development and provide deeper insights into practitioners' challenges and needs.

*4.2.2 Approach.* We categorize question intentions based on the taxonomy proposed by Beyer et al. [19], which includes seven categories: `API USAGE`, `CONCEPTUAL`, `DISCREPANCY`, `ERRORS`, `REVIEW`, `API CHANGE`, and `LEARNING`. We employ three state-of-the-art large language models to classify question intentions: QwQ-32b (zero-shot) [100], GPT-4o (zero-shot) [76], and Deepseek-R1 (one-shot) [34]. We follow best practices in prompt engineering [53, 75] and prior work [40, 59, 102] to optimize model performance, and select zero-shot/one-shot prompting style based on preliminary experiments. In our experiments, the scenario where all three LLMs produce different intents does not occur. We statistically sample a subset (277 posts, sufficient for covering a 90% confidence level and 5% interval) and manually verify the LLM-voted results. At least two LLMs produce the same intent in the majority voting. We compare the manually labeled intents with LLM-voted results (determined by the majority) with a Cohen's Kappa of 0.64, indicating substantial consensus between human work and LLMs.

**Table 5: Number of posts per respective intention category.**

| Intention Category | Definition | Example | Freq |
|---|---|---|---|
| Conceptual | Questions that understand concepts and ask the limitations of an API and API behavior | 30027148 | 586 (48.6%) |
| API Usage | Questions about how to implement certain functionality or how to use an API | 3412026 | 441 (37.0%) |
| Discrepancy | Questions related to exception problems that the observed result is different from the expectation | 22339063 | 272 (22.8%) |
| Review | Questions that ask for the best practice approaches or ask for help to make decisions | 6866236 | 230 (19.3%) |
| Errors | Questions about problems of exceptions with or without code snippets, as well as requiring help in fixing the error or understanding the meaning of the exception | 60771095 | 33 (2.8%) |
| Learning | Questions that ask for documentation or tutorials to learn a tool or language by their own, without asking for a specific instruction or solution | 63105570 | 28 (2.3%) |

* Posts are visitable at https://stackoverflow.com/questions/{ExampleID}

**Table 6: Intents behind each topic.**

| | CONCEPTUAL | API USAGE | DISCREPANCY | REVIEW | ERRORS | LEARNING |
|---|---|---|---|---|---|---|
| Positioning | 50.8% | 40.6% | 16.9% | 23.1% | 1.5% | 1.5% |
| Computing Resource | 54.4% | 27.6% | 24.0% | 16.1% | 1.4% | **3.22%** |
| Mobile Device | **59.4%** | 39.4% | 8.5% | 9.1% | 1.8% | 3.0% |
| Sensor Timing | 36.7% | 43.3% | 25.0% | **29.2%** | 3.3% | 1.7% |
| Polling | 50.0% | 34.5% | 30.9% | 10.9% | 4.5% | 1.8% |
| Datum Handling | 43.3% | 34.0% | **36.1%** | 22.7% | 5.2% | 2.1% |
| Data Transmission | 39.4% | 34.0% | 34.0% | 22.3% | **5.3%** | 3.19% |
| Thread | 41.5% | **44.6%** | 30.8% | 23.1% | 4.6% | 3.1% |

* Each cell represents the frequency of such intent relative to the total number of posts within the corresponding topic. The sum of the frequency values can be greater than 100% since one post may have more than one intentions.

*4.2.3 Findings.* **Understanding fundamental *concepts* of energy usage is a top priority for many developers, occurring in nearly half of their energy-related questions.** Table 5 lists the number of questions for each intention category. Understanding concepts (`CONCEPTUAL`) is the most common intention, accounting for 48.6% of questions. This is in stark contrast to the prior work [19] studying Android development SO questions, where only 26.80% of the questions pertain to `CONCEPTUAL`. Through in-depth analyses, we identify two reasons for this high proportion of conceptual questions: (1) energy consumption spans diverse hardware and software platforms, requiring different measurement approaches and tools. For example, post 66697291 [25] shows that measuring energy consumption of Python scripts requires platform-specific hardware, software, and APIs; and (2) as confirmed by prior work [63, 79], developers struggle with fundamental energy concepts, API limitations, and behavior. For example, post 48766582 [115] illustrates how developers working on battery-powered ARM-based embedded Linux systems face challenges in optimizing power consumption and selecting appropriate kernel APIs.

`CONCEPTUAL` **issues dominate energy-related discussions on multiple topics, especially for the *Mobile Device* topic, implying the complexity of balancing power states on mobile devices.** Table 6 shows the intention distribution across topics, where the *Mobile Device* topic has the highest proportion (59.4%) of `CONCEPTUAL` issues. Our manual analysis reveals three key reasons: (1) mobile devices use multiple operating systems with different power-management models, requiring developers to understand foundational concepts like wake locks and battery monitoring before implementation (post 724349 [89]); (2) the diverse hardware features across devices (e.g., AMOLED screens, custom power-saving

modes) necessitate understanding how components interact to affect battery life (post 2902382 [45]); and (3) Developers need to grasp the principles of CPU usage, screen brightness, sensor polling, and network connectivity to avoid degrading user experience or battery life (post 38635068 [31]).

**Approximately one-third of the energy-related questions concern `API USAGE`, reflecting developers' need for practical implementation guidance.** `API USAGE` questions account for 37.0% of energy-related SO questions, similar to the 38.8% observed in Android development SO questions [19]. These questions often focus on the practices of using certain APIs in energy-efficient ways, such as post 38158828 [103] and post 9863131 [46] that ask about location services. As applications for energy-constrained platforms proliferate, developers increasingly need to optimize system resources (e.g., CPU, memory, and battery) while navigating compatibility challenges across diverse APIs.

***Thread* management is a particularly challenging area, as evidenced by the high frequency of `API USAGE` questions within this topic.** Table 6 reveals that *Thread*-related questions have the highest proportion (44.6%) of `API USAGE` issues. Our investigation identifies three key challenges: (1) threads require precise control for starting, stopping, and synchronization, with improper management leading to energy-wasting issues like infinite loops or race conditions (post 61884014 [93]); (2) efficiently waking threads from idle states is difficult, as poor timing can cause delays that impact responsiveness (post 51973350 [87]); and (3) developers struggle to choose between continuous event monitoring and periodic activation (e.g., using alarm services) to balance power consumption and performance (post 15698999 [11]).

**Practitioners face challenges in resolving discrepancies in energy-aware development.** `DISCREPANCY` issues rank third (22.8%) among energy-related questions, reflecting the need to troubleshoot energy consumption bugs. In post 42850419 [94], a developer seeks help resolving unexpected power consumption in their app and is unsure which components may be relevant to the issue. Energy-related discrepancies are particularly challenging because they require understanding complex interactions between system components and identifying inefficiencies that may not be immediately apparent. The abstract nature of energy consumption, leading to fewer resources and solutions available for practitioners to refer to, further complicates troubleshooting.

***Datum Handling*-related issues are particularly challenging in energy-efficient development, reflecting the intricate nature of efficiently handling data under energy constraints.** We observe that *Datum Handling*-related questions have the highest frequency (36.1%) of discrepancy questions. Our investigation reveals three key challenges: (1) understanding data allocation's impact on power consumption requires deep system-level insights that many developers lack (post 1738515 [60]); (2) inefficient data processing patterns (looping, filtering, selecting) can cause unexpected energy consumption that is difficult to diagnose (post 23617388 [22]); and (3) retrieving data from external systems or libraries often leads to misalignments between expected and actual data definitions, causing energy inefficiencies (post 54028927 [78]). Even minor errors in data processing can significantly impact energy consumption, making data handling particularly challenging in energy-constrained environments. This complexity highlights the need for better tools and guidelines to help developers implement energy-efficient data processing patterns.

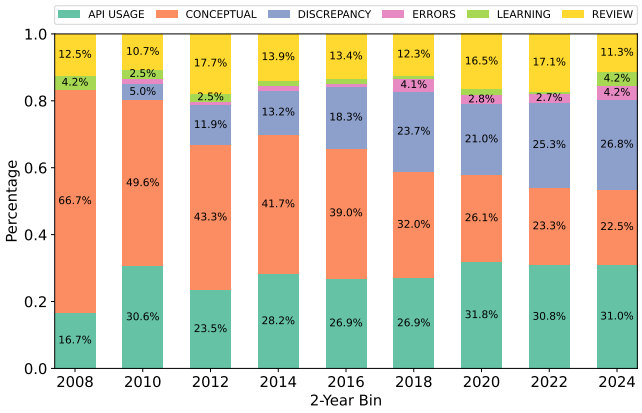


**Figure 4: Intent Distribution Every 2 Years.**

**Intentions of developing energy-efficient software shift from primarily concerning `CONCEPTUAL` issues toward addressing unexpected results/behavior problems (`DISCREPANCY`) over the years**, highlighting evolving focus from theoretical exploration to practical unexpected results/behavior-handling in energy-efficient coding. Pinto et al. identify the theme *General Knowledge* (18.3% by 2013), which is related to `CONCEPTUAL` in our study. As shown in Figure 4, `CONCEPTUAL` questions have been declining over the years, which indicates improvements in the availability and clarity of foundational resources and conceptual documentation for energy-efficient programming. However, the proportion of `DISCREPANCY` issues rises dramatically from 5.0% in 2010 to around 27% by 2024, reflecting a growing demand for actionable guidance on diagnosing and resolving unexpected results/behavior that impact energy consumption. The *Code Design* theme in Pinto et al. (questions about programming techniques that can help in saving energy) accounts for 16.5% of the posts by 2013; this theme approximately corresponds to the `REVIEW` intent in our study, whose frequency is similar (10.7–17.7%).

> ***Summary:*** *Nearly half of energy-related questions focus on understanding fundamental concepts, highlighting the complexity of energy consumption as a cross-cutting concern spanning multiple components. In particular, conceptual questions are particularly prevalent for mobile devices (59.4%), where developers struggle with diverse operating systems, hardware features, and power management models. Questions about API usages (37.0%) and discrepancies (22.8%) are also common, reflecting the need for practical guidance and troubleshooting resources. Our intent evolution analysis reveals a shift from theoretical exploration (`CONCEPTUAL`) to practical unexpected results/behavior-handling (`DISCREPANCY`).*

## 4.3 RQ3: What technologies are concerned in energy-efficient development?

*4.3.1 Motivation.* Stack Overflow questions are usually tagged with their relevant technologies, e.g., operating systems (Linux, Windows, etc.), programming languages (Java, C, etc.), and hardware components (mobile, server, etc.). Identifying which technologies are most frequently discussed helps direct technical support

toward addressing the most prevalent energy-related challenges faced by practitioners.

*4.3.2 Approach.* We identify the concerned technologies of SO questions by analyzing their tags in the following steps:

**Merging Synonymous Tags**. The same technology may sometimes be described with several semantically similar tags, e.g., "bluetooth" and "bluetooth-lowenergy", which we call synonymous tags. We merge synonymous tags into a single category, specifically the lower-frequency tag is merged into the higher-frequency tag (e.g., "bluetooth-lowenergy" is merged into "bluetooth").

**Selecting Frequently Occurring Tags**. After merging synonymous tags, we have 1,008 unique tags occurring 4,803 times in total. The tags follow a long-tail distribution, where many tags only appear once or twice in our dataset. To focus on the most important technologies, we follow the Pareto principle (i.e., 80/20 rule) and select the top frequently occurring tags that account for at least 80% of all occurrences. After including tags with the same frequency, we have 363 tags covering 84.2% of all tag occurrences. We further inspect remaining tags and decide to focus on the tags related to software development technologies (77 out of 363), as the remaining tags are irrelevant or infrequent (many with ≤1% frequency) and may introduce noises to the analysis.

**Grouping Tags**. We group the 77 tags into three main categories: operating systems (*OS*, 10 tags, e.g., android, ios, and linux), programming languages (*PL*, 23 tags, e.g., java, c, and python), and hardware (*HW*, 44 tags). The hardware category is further divided into subcategories of mobile (e.g., ipad), accessory (e.g., bluetooth), server (e.g., cloud), embedded (e.g., rasberry-pi), and processing units (e.g., cpu, gpu, and memory). Among the posts, 325 (27.2%) include multiple tags that span different categories. We experimented with making tags unique per post by randomly selecting one of multiple tags, which led to a frequency shift by only -1.7% to 2.3% per category from our results. We choose to keep multiple tags, as they are assigned by SO users in arbitrary orders and there is no evidence to indicate the importance of tags.

**Counting Posts**. For each tag, we count the number of posts associated with it to determine its prevalence in energy-related discussions with regard to topics (§4.1) and developer intents (§4.2). This helps us identify which technologies are most frequently concerned by practitioners in relation to energy consumption.

*4.3.3 Findings.* ***OS* is the most discussed category of technologies in energy-related posts.** Table 7 lists the technologies and their corresponding tags, with the numbers of posts belong to each technology category. *OS*-related tags appear in 533 posts, more than *HW* (415 posts) and *PL* (357 posts). Note that posts can have multiple tags (e.g., post 17134522 [48] has both embedded and Objective-C tags), the percentages across categories do not sum to 100%.

**A higher number of energy-related questions are found in the context of Mobile OS than PC/server OS.** Table 8 shows the percentage of posts for each specific technology, with Android (28.1%) and iOS (11.5%) together accounting for nearly 40% of all posts. Meanwhile, Linux (6.2%), Windows (4.9%), and MacOS (1.3%) collectively represent only about 12%. This aligns with Bao et al. [15]'s findings on the complexity of Android power management, where different app categories require distinct power management strategies.

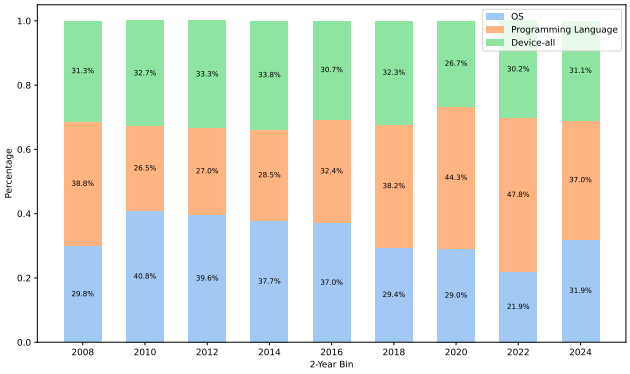


**Figure 5: Tag Category Distribution Every 2 Years**

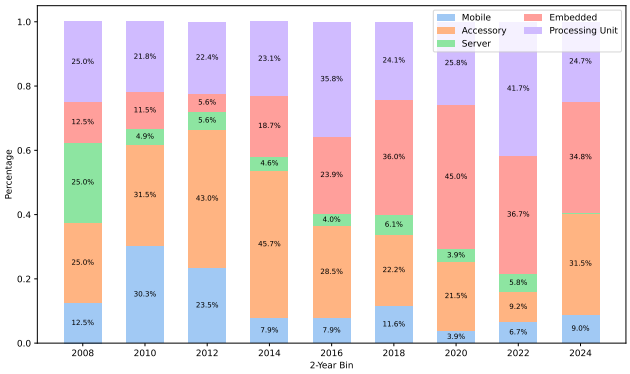


**Figure 6: Device Details**

**Resolving energy-related questions often requires referring to other knowledge sources via links.** Our manual inspection reveals that over 60% the accepted answers require linking to external knowledge, such as documentations of APIs and tools, practical experience, and in-depth knowledge of the underlying concepts (e.g., post 11366329 [27], post 1298600 [24], post 3655806 [77], post 6051807 [97], and post 6553595 [73]). Meanwhile, 43.0% of questions can be resolved using internal resources like other users' solutions, API documentation, software specifications, and debugging (e.g., post 13873570 [99], post 40090627 [35], post 31368757 [98], post 8100506 [51], and post 11398732 [33]). However, even though APIs provide specifications and documentation, practitioners still face challenges in finding and understanding them, reflected in the fact that 54.0% of accepted answers include links to supplementary materials to support practitioners better follow and understand the solution and usage. Additionally, we notice that questions with respect to PC/server-based systems tend to focus more on measuring and profiling energy consumption rather than managing hardware complexity (e.g., post 49202065 [62], post 7312597 [66], post 410122 [61], post 3655806 [77], and post 72711521 [55]).

***OS*-related energy questions show a declining trend over time.** Figure 5 shows a declining trend in *OS*-related energy questions over time[2]. This likely reflects three developments: (1) cloud providers now handle energy optimization at the infrastructure level; and (2) modern OS kernels have integrated sophisticated power-saving mechanisms like CPU scheduling and dynamic resource allocation [9, 67, 96].

[2]The increasing trend from 2022 to 2024 is mainly due to the small sample size in 2024.

**Table 7: Technology specification.**

| Tag Category | | | Tag List |
|---|---|---|---|
| OS (533) | | | android, ios (ios, ios7, and ios8), linux, windows, macos, ubuntu, wear-os, contiki |
| Programming Language (357) | | | java, c, python, c++, swift (swift and swift3), c#, objective-c, sql, php, asp.net, verilog, node.js, assembly, shell, vhdl, matlab, bash, kotlin, json, jquery, css, javascript |
| Hardware (415) | Mobile (22) | | apple, ipad, mobile, phone, tablet |
| | Accessory (175) | | usb, battery, accelerometer, beacon, bluetooth, gps, ibeacon, usb, wifi, sensors |
| | Server (7) | | server, computer, cloud |
| | Embedded (92) | | embedded, stm32, esp8266, esp32, arduino, pi, raspberry, fpga, xilinx, iot |
| | Processing Unit (123) | Central (77) | microchip, microcontroller, amd, arm, core, cpu, intel, x86, msr |
| | | Graphics (39) | gpgpu, gpu, nvidia, cuda |
| | | Storage (12) | memory |

* The number of posts for each tag category or sub-category is listed in parentheses (e.g., OS (533)).
** Since a post may span multiple tag categories, the sum of subcategories does not add up directly (e.g., the number of posts in Processing Unit does not equal to the sum of Central, Graphics, Storage, and Embedded).

**Table 8: Distribution of posts related to technologies.**

| | Operating System | | | | | Programming Language | | | | | | | | | Hardware | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | Processing Unit | | |
| Tag | **Android** | iOS | Linux | Windows | MacOs | **Java** | Python | C | Swift | C++ | C# | SQL | Javascript | Objective-C | **Accessory** | Embedded | Mobile | Server | **Central** | Graphics | Storage |
| Freq | **28.1%*** | 11.5% | 6.2% | 4.9% | 1.3% | **5.9%** | 5.5% | 4.9% | 4.0% | 3.9% | 2.5% | 1.8% | 1.6% | 1.4% | **14.7%** | 7.7% | 1.8% | 0.6% | **6.5%** | 3.3% | 1.0% |

* The figure indicates the percentage of tag category-related posts (e.g., Android) among all posts.

**Among *HW* technologies, practitioners are most concerned about the energy consumption of accessories (e.g., Bluetooth or GPS), followed by processing units (e.g., CPU, GPU, or microchip).** Table 8 shows that accessories (14.7%) are the most discussed hardware components, followed by embedded systems (7.7%) and CPUs (6.5%). Accessories like Bluetooth and GPS draw significant attention because they continuously send and receive signals, consuming substantial energy—a finding consistent with prior work [14]. Processing units (CPU, GPU, and microchips) also receive considerable attention, as they are responsible for executing code and performing calculations. Nowadays, modern processors have become more energy-efficient through architectural improvements and low-power states like Intel's SpeedStep and AMD's Cool'n'Quiet [4, 54]. Figure 6 reveals interesting trends in hardware concerns: server and accessory-related questions have decreased over time, while embedded hardware questions have increased. Server energy concerns have diminished due to cloud migration and mature data center optimizations [17, 64]. Accessory concerns have decreased as components like Bluetooth and GPS now include robust power-saving features and established best practices [44]. Conversely, embedded hardware questions have surged with IoT growth, increasing microcontroller complexity, and battery-powered device proliferation [7, 37].

Java **is the most concerned single programming language, and the C family of programming languages adding together holds a significant part of the practitioners' concerns.** Java is the most discussed individual programming language (5.9%), likely due to its use in Android development. However, the C family of programming languages collectively dominates energy-related discussions, with C (4.9%), C++ (3.9%), C# (2.5%), and Objective-C (1.4%) together accounting for 11.3% of programming language discussions. This reflects the widespread use of C family in hardware, OS development (e.g., Linux kernel), and embedded systems. Figure 5 shows increasing attention to programming languages in energy-efficient development. This trend emerges as large-scale applications (cloud services and machine learning) written in high-level languages like Java and Python become more prevalent. Even small code optimizations can yield substantial energy savings when scaled across thousands of servers [18, 26]. Additionally, as OS-level power management matures, developers increasingly seek efficiency gains at the application and language levels [23, 113].

***Summary:*** *Our analysis of the technologies concerned in energy-related posts highlights the need for targeted support in three key areas: (1) mobile OS energy optimization, where battery constraints and complex component management create unique challenges; (2) efficient hardware utilization, particularly for accessories and embedded systems in IoT applications; and (3) language-specific optimization techniques, especially as high-level languages become more prevalent in large-scale applications. The shifting trends suggest that energy optimization focus is moving from system-level to application-level and specialized device contexts.*

## 5 Implications

Our findings suggest that **energy-aware software development should be addressed and trained in a systematic way: from hardware devices (e.g., optimizing the usage of energy-intensive devices such as GPS), to operating systems (the considerations for servers and mobile operating systems), then to programming languages (the particularity of different languages' support for energy-aware development)**. Below, we discuss some detailed implications based on our findings.

**Minimizing direct usage of hardware accessories when software-based saving is possible (e.g., by relying on deferred location APIs, or batching/deferring hardware accesses).** §4.1 emphasizes a continuous concern of low-level hardware (*Positioning*), and §4.3 highlights that the majority of questions posed by practitioners are related to the energy consumption of hardware accessories, such as Bluetooth and GPS. Specifically, a software application that frequently makes use of Bluetooth or GPS can be tuned to activate the hardware accessories only when necessary rather than keeping them on continuously. For example, if coarse accuracy of position is acceptable, practitioners could call `requestLocationUpdates()` in Android's `LocationManager` with a large minimum time interval and `NETWORK_PROVIDER` (which relies on cell-tower/Wi-Fi rather than GPS) to avoid unnecessary high-energy GPS fixes (post 3034890 [2]).

**Leveraging platform's built-in scheduling and power management APIs to minimize unnecessary background task wake-ups.** As discussed in §4.3, mobile OS like Android have significant energy concerns, whereas `Java` is the most concerned single *PL* for Android. We observe that excessive background tasks, such as tight wakelock loops, frequent sensor listeners, and unmanaged threads, keep CPU and other hardware active, which rapidly drain the battery. To mitigate this, practitioners could use `AlarmManager` instead of a manual wakelock loop to save power at idle time. For example, tasks can be scheduled with an exact alarm `AlarmManager.setExactAndAllowWhileIdle()` that fires even under the idle mode, allowing the system to sleep fully until that moment instead of constantly holding CPU awake (e.g., post 10468305 [112]). Besides, practitioners could hold the `WakeLock` in persistent fields rather than locals so they can reliably release it and let CPU return to idle sooner. For example, `WakeLock` could be used as a class-level (non-local) field to ensure that the lock is not garbage-collected immediately (e.g., post 38325958 [8]).

**Making optimized usage of built-in API routines, especially being precautionary of per-element processing patterns.** In §4.2, our findings suggest that `API USAGE` issues remain consistently significant (~30%) across all years. We observe that such issues typically stem from per-element processing patterns (e.g. nested loops or full-list rebuilds) and by mismanaging asynchronous or lifecycle APIs, which lead to repeated work or busy-wait loops that spike CPU usage and power draw. For instance, in Kivy GUI updates (post 66972381 [95]), rather than reconstructing the full RecycleView dataset on each incoming serial-port message, practitioners are suggested to keep a persistent Python list of messages, append only the new entries to it, and then update the RecycleView's `data` property only when necessary to avoid a full list rebuild and drastically cutting CPU and power usage.

**Pushing data-handling optimizations down to the level where they can actually control resource usage (e.g., OS/kernel, hardware components, and data structures) rather than trying to tweak high-level app code.** As discussed in §4.1, we notice a technological shift to an emphasis on data-centric topics (e.g., *Datum Handling*). In fact, questions about *Datum Handling* not only have the highest non-acceptance rates but also a longer median response time. To address this challenge, practitioners could dynamically cache power-gating via dynamic voltage and frequency scaling—by lowering the supply voltage and clock rate of cache subarrays and ramping up or powering on the data banks only when a tag lookup hits—to conserve power on workloads with sparse cache traffic. For example, post 41749881 [71] suggests that physically separating the tag and data arrays to keep the tag array powered at a low voltage/frequency and only power up the data banks on a cache-hit, thereby saving energy on workloads with low cache-access intensity.

## 6 Threats to Validity

**Construct Validity**. Even though our data cover a wide range of SO posts, our selection of posts may not be exhaustive. The automatic keyword filtering may missed some relevant posts, but manually verifying all posts in the data dump is infeasible. The size of the data and the linguistic freedom of presentation—different words present the same meaning—can have a significant impact on the results. To mitigate these threats, we conduct manual inspection twice: one is to filter out the false positive posts (matching keywords but unrelated to energy consumption), and another is to validate the LDA result and intention categories. The manual inspection is performed by specialists (two PhD researchers in Software Engineering with over 7 years of experience) following Seaman's closed coding procedures [92] (§3.2). While analyzing questions and text content may provide deeper insights (§4.3), we rely on SO tags as they are community-curated to reflect the primary technologies or topics involved. Leveraging tags allows our analysis to remain consistent and replicable across large datasets. Meanwhile, prior studies [16, 83] demonstrate that SO tags accurately reflect developers' primary technological concerns. To mitigate limitations from tag-based analysis, we merge synonyms or related tags, thereby capturing broader trends and technology categories comprehensively to ensure our results represent the underlying data.

**Internal Validity**. We choose the LDA model as our research method, whose results may be inaccurate. To mitigate this, the LDA topics are further validated via manual inspection. Another challenge arises from employing LLMs to label intents in RQ2. To ensure that the LLMs robustly generate accurate and contextually relevant outputs, we follow best practices from previous studies [40, 59, 102] and engineered prompts [53, 75] and also manually verify the results are valid. We select the tags of posts to summarize technology categories with regard to the occurrences of the tags. Nonetheless, the occurrence may not indicate the true relationship between the theme of a post and the technologies. To mitigate this, we cover the most development category-related tags, which increases our relevance to the study. We utilize LLM voting to infer question intentions, which might be unreliable if not carefully configured. To mitigate this, we manually label intents and compare them with LLM-voted results. The result shows a substantial consensus (Cohen's Kappa of 0.64) between human work and LLMs.

**External Validity.** Our results apply only to questions interested in energy usage on SO. It does not include other Q&A websites, such as Ask Ubuntu, nor does it include the posts asked in other languages (e.g., French). Although SO is the most popular development Q&A website, further investigation could be conducted to subsume additional sources.

## 7 Conclusion

In this paper, we conduct an empirical study to investigate how practitioners on Stack Overflow are concerned about energy consumption. By studying 1,193 energy-related questions, we observe the following: (1) understanding the energy impact of *Positioning* service and *Computing Resource* used by their programs are of particular important to practitioners; (2) *Polling* and *Datum Handling* are the most difficult topics to receive community support due to in-depth knowledge required; (3) most practitioners struggle with fundamental energy concepts due to complex API documentation and practical experience required; and (4) practitioners consider energy-efficient development from multiple levels—*OS*, *HW*, and *PL*, where `Android`, `Accessory` , and `Java` are the most concerned technologies, respectively. Based on our findings, we provide actionable suggestions for knowledge-sharing communities and practitioners.

## Acknowledgments

We would like to thank Dr. Stefanos Georgiou for the helpful discussions and feedback on the early work of this paper.

## References


[1] Amritanshu Agrawal, Wei Fu, and Tim Menzies. 2018. What is wrong with topic modeling? And how to fix it using search-based software engineering. *Information and Software Technology* 98 (2018), 74–88.

[2] ahsteele. 2010. Determine Android phone's proximity to known point while conserving power. https://stackoverflow.com/questions/3034890

[3] Miltiadis Allamanis and Charles Sutton. 2013. Why, When, and What: Analyzing Stack Overflow Questions by Topic, Type, and Code. In *Proceedings of the 10th Working Conference on Mining Software Repositories*. 53–56.

[4] AMD. 2004. Cool 'n' Quiet™ Technology Installation Guide for AMD Athlon™ 64 Processor Based Systems. https://web.archive.org/web/20070409045621/http://www.amd.com/us-en/assets/content_type/DownloadableAssets/Cool_N_Quiet_Installation_Guide3.pdf

[5] Anders Andrae. 2020. New perspectives on internet electricity use in 2030. *Engineering and Applied Science Letters* 3 (June 2020), 19–31.

[6] Antonio. 2013. Do sse instructions consume more power/energy? https://stackoverflow.com/questions/19722950

[7] Ons Aouedi, Thai-Hoc Vu, Alessio Sacco, Dinh C. Nguyen, Kandaraj Piamrat, Guido Marchetto, and Quoc-Viet Pham. 2024. A Survey on Intelligent Internet of Things: Applications, Security, Privacy, and Future Directions. *IEEE Communications Surveys & Tutorials* (2024), 1–1.

[8] apidae. 2016. android service logging sensor data uninterrupted. https://stackoverflow.com/questions/38325958

[9] Apple. 2013. Kernel Programming Guide. https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/performance/performance.html#//apple_ref/doc/uid/TP30000905-CH207-BEHJDFCA

[10] atasoyh. 2011. How can I sense if a phone is in standby or sleep mode(dor Nokia)? https://stackoverflow.com/questions/5151872

[11] awonder. 2013. Implementing a battery widget with alarmservice? https://stackoverflow.com/questions/15698999

[12] Mehdi Bagherzadeh and Raffi Khatchadourian. 2019. Going Big: A Large-Scale Study on What Big Data Developers Ask. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Tallinn, Estonia) *(ESEC/FSE 2019)*. Association for Computing Machinery, New York, NY, USA, 432–442.

[13] Manish Bane. 2014. Save continuous location data on server through android. https://stackoverflow.com/questions/26336225

[14] Abdul Ali Bangash, Daniil Tiganov, Karim Ali, and Abram Hindle. 2021. Energy Efficient Guidelines for iOS Core Location Framework. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 320–331.

[15] Lingfeng Bao, David Lo, Xin Xia, Xinyu Wang, and Cong Tian. 2016. How Android App Developers Manage Power Consumption? - An Empirical Study by Mining Power Management Commits. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. 37–48.

[16] Anton Barua, Stephen W. Thomas, and Ahmed E. Hassan. 2014. What Are Developers Talking about? An Analysis of Topics and Trends in Stack Overflow. *Empirical Software Engineering* 19, 3 (June 2014), 619–654.

[17] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, and Albert Zomaya. 2011. Chapter 3 - A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems. Advances in Computers, Vol. 82. Elsevier, 47–111.

[18] Luca Benini, Mahmut Kandemir, and J Ramanujam. 2011. *Compilers and Operating Systems for Low Power*. Springer Science & Business Media.

[19] Stefanie Beyer, Christian Macho, Massimiliano Di Penta, and Martin Pinzger. 2020. What Kind of Questions Do Developers Ask on Stack Overflow? A Comparison of Automated Approaches to Classify Posts into Question Categories. *Empirical Softw. Engg.* 25, 3 (May 2020), 2258–2301.

[20] David Blei, Andrew Ng, and Michael Jordan. 2001. Latent Dirichlet Allocation. *The Journal of Machine Learning Research* 3, 601–608.

[21] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3, null (March 2003), 993–1022.

[22] bob dylan. 2014. How to select element in a php multi dimentional array like the WHERE in mysql. https://stackoverflow.com/questions/23617388

[23] Aaron Carroll and Gernot Heiser. 2010. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference* (Boston, MA) *(USENIXATC'10)*. USENIX Association, USA, 21.

[24] Antony Carthy. 2009. Bluetooth UUID discovery. https://stackoverflow.com/questions/1298600

[25] cesarcastellon.sec. 2021. Measure energy consumed by a python script in Raspberry Pi (Raspbian). https://stackoverflow.com/questions/66697291

[26] Yanpei Chen, Archana Ganapathi, Rean Griffith, and Randy Katz. 2011. The Case for Evaluating MapReduce Performance Using Workload Suites. In *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*. 390–399.

[27] Chris. 2012. Implementing an event and periodically driven "script language" in C++? https://stackoverflow.com/questions/11366329

[28] Paolo Ciancarini, Shokhista Ergasheva, Kholmatova Zamira, Artem Kruglov, Giancarlo Succi, Xavier Vasquez, and Evgeniy Zuev. 2020. Analysis of Energy Consumption of Software Development Process Entities. *Electronics* 9 (October 2020), 1678.

[29] Codifier. 2012. Decentralized Clustering library for Java. https://stackoverflow.com/questions/8782922

[30] Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20, 1 (1960), 37–46.

[31] Luis Cruz. 2016. Does Android have a different behavior for WakeLock when the device is connected to power source? https://stackoverflow.com/questions/38635068

[32] Luis Cruz and Rui Abreu. 2019. Catalog of energy patterns for mobile applications. *Empirical Softw. Engg.* 24, 4 (Aug. 2019), 2209–2235.

[33] dda. 2012. How do I receive the system broadcast when GPS status has changed? https://stackoverflow.com/questions/11398732

[34] DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL]

[35] Jackie Degl'Innocenti. 2016. How to structure a notification system for a chat app using Firebase Database and Firebase Notification. https://stackoverflow.com/questions/40090627

[36] dev_android. 2014. Android LocationListener for a certain time. https://stackoverflow.com/questions/23560949

[37] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. 2017. *Microservices: Yesterday, Today, and Tomorrow*. Springer International Publishing, Cham, 195–216.

[38] Eitan. 2010. Estimating process energy usage on PCs (x86). https://stackoverflow.com/questions/4485153

[39] Eitan. 2017. Detect car parking efficiently on Android. https://stackoverflow.com/questions/43596157

[40] Pouya Fathollahzadeh, Mariam El Mezouar, Hao Li, Ying Zou, and Ahmed E Hassan. 2025. Towards Refining Developer Questions using LLM-Based Named Entity Recognition for Developer Chatroom Conversations. *arXiv preprint arXiv:2503.00673* (2025).

[41] Charlotte Freitag, Mike Berners-Lee, Kelly Widdicks, Bran Knowles, Gordon Blair, and Adrian Friday. 2021. The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations. *Patterns* 2 (September 2021), 100340.

[42] Gensim. [n. d.]. LDA Model. https://radimrehurek.com/gensim/auto_examples/tutorials/run_lda.html

[43] golden. 2008. Power Efficient Software Coding. https://stackoverflow.com/questions/61882

[44] Carles Gomez, Joaquim Oller, and Josep Paradells. 2012. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *sensors* 12, 9 (2012), 11734–11753.

[45] Maksym Gontar. 2010. Android Nexus One - Can I save energy with color scheme? https://stackoverflow.com/questions/2902382

[46] gorn. 2012. Android accelerometer, sensor usage and power consumption. https://stackoverflow.com/questions/9863131

[47] Derek Greene, Derek O'Callaghan, and Padraig Cunningham. 2014. How Many Topics? Stability Analysis for Topic Models. 498–513.

[48] gregko. 2013. How to compare the performance of Android Apps written in Java and Xamarin C#? Anyway to check quantitative data (code & results). https://stackoverflow.com/questions/17134522

[49] Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in Semantic Representation. *Psychological Review* 114, 2 (2007), 211–244.

[50] Iman H. 2018. Java TCP server for communicating with an IoT device. https://stackoverflow.com/questions/53359800

[51] halfer. 2011. How do I use Android PowerProfile private API? https://stackoverflow.com/questions/8100506

[52] Chunqi Hu, Huaping Gong, and Yiqing He. 2022. Data driven identification of international cutting edge science and technologies using SpaCy. *PLoS ONE* 17 (2022).

[53] Huggingface. [n. d.]. Chat Templates. https://huggingface.co/docs/transformers/main/en/chat_templating

[54] Intel. 2022. Overview of Enhanced Intel SpeedStep® Technology for Intel® Processors. https://www.intel.com/content/www/us/en/support/articles/000007073/processors.html

[55] Jun Ish. 2022. Win11 22H2 Programmatically switch between predefined power modes (Best power efficiency/Balanced/Best performance). https://stackoverflow.com/questions/72711521

[56] Jamaal. 2015. Getting latitude and longitude updates based off of change in distance for Android. https://stackoverflow.com/questions/29616977

[57] Lei Jin, Keran Duan, and Xu Tang. 2018. What Is the Relationship between Technological Innovation and Energy Consumption? Empirical Analysis Based on Provincial Panel Data from China. *Sustainability* 10 (01 2018), 145.

[58] Swaranjali Jugran, Ashish Kumar, Bhupendra Singh Tyagi, and Vivek Anand. 2021. Extractive Automatic Text Summarization using SpaCy in Python & NLP. In *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. 582–585.

[59] Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R Bowman, Tim Rocktäschel, and Ethan Perez. 2024. Debating with more persuasive llms leads to more truthful answers. *arXiv preprint arXiv:2402.06782* (2024).

[60] kjv. 2009. RAM memory reallocation - Windows and Linux. https://stackoverflow.com/questions/1738515

[61] Ksempac. 2009. Reducing power consumption on a low-load server running WinXP? https://stackoverflow.com/questions/410122

[62] lel. 2018. How to get CPU power consumption in PowerShell? https://stackoverflow.com/questions/49202065

[63] Irene Manotas, Christian Bird, Rui Zhang, David Shepherd, Ciera Jaspan, Caitlin Sadowski, Lori Pollock, and James Clause. 2016. An Empirical Study of Practitioners' Perspectives on Green Software Engineering. In *Proceedings of the 38th International Conference on Software Engineering* (Austin, Texas) *(ICSE '16)*. Association for Computing Machinery, New York, NY, USA, 237–248.

[64] Dan C Marinescu. 2022. *Cloud computing: theory and practice*. Morgan Kaufmann.

[65] maryam. 2016. network-on-chip verilog code. https://stackoverflow.com/questions/39804144

[66] MetallicPriest. 2011. Is using hardware performance counters a good idea. https://stackoverflow.com/questions/7312597

[67] MicroSoft. 2021. Scheduling. https://learn.microsoft.com/en-us/windows/win32/procthread/scheduling

[68] MinsubKim. 2017. How to measure power consumption of Jetson TX1? https://stackoverflow.com/questions/44228005

[69] Lags Moomba. 2012. Optimising an 1D heat equation using SIMD. https://stackoverflow.com/questions/13476223

[70] Irineu Moura, Gustavo Pinto, Felipe Ebert, and Fernando Castor. 2015. Mining Energy-Aware Commits. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. 56–67.

[71] Mrchacha. 2017. Separated tag array versus combined with data array. https://stackoverflow.com/questions/41749881

[72] David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*.

[73] Nimantha. 2011. Sending a String array through a tigase server from one Android to another using XMPP protocol. https://stackoverflow.com/questions/6553595

[74] olivierg. 2011. Is it ok to update a widget frequently for a short period of time? https://stackoverflow.com/questions/5765212

[75] OpenAI. [n. d.]. Prompt engineering. https://platform.openai.com/docs/guides/prompt-engineering

[76] OpenAI. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL] https://arxiv.org/abs/2303.08774

[77] Ciro Santilli OurBigBook.com. 2010. How to measure power consumed by my algorithm? https://stackoverflow.com/questions/3655806

[78] Jovan Pacheco. 2019. Implementing a battery widget with alarmservice? https://stackoverflow.com/questions/54028927

[79] C. Pang, A. Hindle, B. Adams, and A. E. Hassan. 2016. What Do Programmers Know about Software Energy Consumption? *IEEE Software* 33, 03 (May 2016), 83–89.

[80] Guillaume Petitjean. 2014. Why dynamic power consumption is always zero? https://stackoverflow.com/questions/27383269

[81] Guillaume Petitjean. 2018. Optimize power consumption with STM32L4 ADC. https://stackoverflow.com/questions/53173447

[82] Guillaume Petitjean. 2018. Power consumption of a Systick timer on STM32. https://stackoverflow.com/questions/50064364

[83] Gustavo Pinto, Fernando Castor, and Yu David Liu. 2014. Mining Questions about Software Energy Consumption. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (Hyderabad, India) *(MSR 2014)*. Association for Computing Machinery, New York, NY, USA, 22–31.

[84] Sanjay Podder, Adam Burden, Shalabh Kumar Singh, and Regina Maruca. 2020. How green is your software?

[85] popoe. 2018. OpenGL: How to minimize drawing? https://stackoverflow.com/questions/53577434

[86] programmerRaj. 2019. Is there a way to limit the amount of CPU a c++ application uses. https://stackoverflow.com/questions/58156084

[87] quantumfoam. 2018. windowed OpenGL first frame delay after idle. https://stackoverflow.com/questions/51973350

[88] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50.

[89] Daniel Rikowski. 2009. How can I measure the energy consumption of my application on Windows Mobile and Windows CE? https://stackoverflow.com/questions/724349

[90] Christoffer Rosen and Emad Shihab. 2016. What are mobile developers asking about? a large scale study using stack overflow. *Empirical Software Engineering* 21, 3 (2016), 1192–1223.

[91] Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the Space of Topic Coherence Measures. *ACM* (2015), 399–408.

[92] C.B. Seaman. 1999. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering* 25, 4 (1999), 557–572.

[93] Alejandro Servetto. 2020. How do I make sure that a Java thread will never run an infinite loop? https://stackoverflow.com/questions/61884014

[94] Amit Shadadi. 2017. My app keep the device awake all the time. https://stackoverflow.com/questions/42850419

[95] SharonWingle. 2021. Kivy Desktop App for Serial Data communication in fastest possible way. https://stackoverflow.com/questions/66972381

[96] Wander Siemers, Luís Cruz, and Mattia Fazzini. 2025. Toward Understanding and Detecting Battery Saver Issues in Android Apps. In *Proceedings of the 12th International Conference on Mobile Software Engineering and Systems* (Ottawa, Canada) *(MOBILESoft '25)*. Association for Computing Machinery, New York, NY, USA.

[97] skaffman. 2011. per process power consumption in Android. https://stackoverflow.com/questions/6051807

[98] Spirrow. 2015. How to obtain the remaining energy from BatteryManager.BATTERY_PROPERTY_ENERGY_COUNTER. https://stackoverflow.com/questions/31368757

[99] Squonk. 2012. Correct way to implement a stock price alert system. https://stackoverflow.com/questions/13873570

[100] Qwen Team. 2025. QwQ-32B: Embracing the Power of Reinforcement Learning. https://qwenlm.github.io/blog/qwq-32b/

[101] Prashant Tiwari. 2012. Energy efficient GPS tracking. https://stackoverflow.com/questions/10920904

[102] Haley Triem and Ying Ding. 2024. "Tipping the Balance": Human Intervention in Large Language Model Multi-Agent Debate. *Proceedings of the Association for Information Science and Technology* 61, 1 (2024), 361–373.

[103] user3290180. 2016. Android Google Play Service: Best practices to make location request. https://stackoverflow.com/questions/38158828

[104] user534498. 2015. Power consumption of a Systick timer on STM32. https://stackoverflow.com/questions/28760898

[105] Pradeep K. Venkatesh, Shaohua Wang, Feng Zhang, Ying Zou, and Ahmed E. Hassan. 2016. What Do Client Developers Concern When Using Web APIs? An Empirical Study on Developer Forums and Stack Overflow. In *2016 IEEE International Conference on Web Services (ICWS)*. 131–138.

[106] Roberto Verdecchia, Patricia Lago, Christof Ebert, and Carol de Vries. 2021. Green IT and Green Software. *IEEE Software* 38, 6 (2021), 7–15.

[107] Hanna M. Wallach, David Mimno, and Andrew McCallum. 2009. Rethinking LDA: Why Priors Matter. In *Neural Information Processing Systems*.

[108] Zhiyuan Wan, Xin Xia, and Ahmed E. Hassan. 2021. What Do Programmers Discuss About Blockchain? A Case Study on the Use of Balanced LDA and the Reference Architecture of a Domain to Capture Online Discussions About Blockchain Platforms Across Stack Exchange Communities. *IEEE Transactions on Software Engineering* 47, 7 (2021), 1331–1349.

[109] Peng Wang, peirong zhong, Min Yu, Yanru Pu, Shuainan Zhang, and Ping Yu. 2022. Trends in energy consumption under the multi-stage development of ICT: Evidence in China from 2001 to 2030. *Energy Reports* 8 (July 2022).

[110] wondering. 2015. Why does reading the CP15 c1 control register fail (ARMv7 inline asm)? https://stackoverflow.com/questions/28003660

[111] Xin-Li Yang, David Lo, Xin Xia, Zhiyuan Wan, and Jian-Ling Sun. 2016. What Security Questions Do Developers Ask? A Large-Scale Study of Stack Overflow Posts. *Journal of Computer Science and Technology* 31 (September 2016), 910–924.

[112] Zelig. 2012. Reducing power consumption. https://stackoverflow.com/questions/10468305

[113] Heng Zeng, Carla S. Ellis, Alvin R. Lebeck, and Amin Vahdat. 2002. ECOSystem: managing energy as a first class operating system resource. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems* (San Jose, California) *(ASPLOS X)*. Association for Computing Machinery, New York, NY, USA, 123–132.

[114] Zhenyun Zhuang, Kyu-Han Kim, and Jatinder Pal Singh. 2010. Improving energy efficiency of location sensing on smartphones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services* (San Francisco, California, USA) *(MobiSys '10)*. Association for Computing Machinery, New York, NY, USA, 315–330.

[115] Étienne. 2018. Is there a way to stop Qt rendering temporarily? https://stackoverflow.com/questions/48766582