Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 4445751, 14 pages
https://doi.org/10.1155/2022/4445751

Research Article

Gathering Contextual Data with Power Information Using Smartphones in Internet of Everything

Umar Mahmud,[1] **Shariq Hussain**,[1] **and Ibrahima Kalil Toure**[2]

[1]*Department of Software Engineering, Foundation University Islamabad (FUI), Pakistan*
[2]*Department of Computer Science, Gamal Abdel Nasser University, Conakry, Guinea*

Correspondence should be addressed to Ibrahima Kalil Toure; ikalil@msn.com

Received 30 January 2022; Revised 15 April 2022; Accepted 16 April 2022; Published 6 May 2022

Academic Editor: Alireza Souri

The advent of smart devices, interacting with each other as well as remote services, has paved the way for the Internet of Everything (IoE). IoE is a direct successor of Internet of Things (IoT), composed of smart devices interacting with remote services. The devices in an IoE environment are power constrained. At the core of an IoE environment, there is a context-aware system that gathers the context and classifies it. Various datasets have been published by authors for context-aware systems. This paper presents a mechanism that gathers a dataset of contextual information along with power information using smartphones. An Android application "PowerIpsum" is developed for gathering contextual information, power information, and user input activity labels. The dataset includes the sensor data as the contextual data, timestamps, average current, and average voltage as well as user activity labels. Time elapsed and power consumption is forecasted using Monte Carlo method. The results provide useful insights and demonstrate the advantages of power information within a context-aware system.

1. Introduction

The fourth industrial revolution has paved the way for establishing smart environments in mundane places [1]. Statista estimates that in 2019, almost 27 billion devices were connected globally [2]. This number reduced to 12.3 billion in 2021, due to COVID, with over a $160 billion industry [3]. Statista forecasted the total number of devices to be more than 75 billion in 2025. Cisco estimates there would be a trillion devices by 2025 [4, 5]. The International Data Corporation (IDC) predicts 40 billion connected devices generating 80 zettabytes of data by 2025 [6].

The dawn of smart, inexpensive, handheld, seamlessly integrated, and universally available devices have expanded the horizons of human potential [7, 8]. Over the years, the devices have become smaller, have more computation power, communicate seamlessly, and facilitate human activity recognition [9, 10]. Smart devices communicate not only among themselves but with remote services over the cloud. The smart devices gather sensor data for a users' activity classification. This helps in facilitating service discovery, delivery, and adaptation. Activity classification requires gathering contextual information about the users, remote services, and the environment [11, 12]. The contextual information includes both the sensor data as well as deduced data from remote services. The context is gathered, transformed, stored, and classified for activity [13, 14]. The prime challenge of a smart environment is the ability of the system to react appropriately to changing perceptions. For example, a change in brightness and orientation of a smartphone with the change in orientation and ambient light is an example of context-awareness. This is possible using an ambient light sensor and accelerometer in a smartphone. In an IoE environment, data is

gathered from the sensors and remote services, stored, and inferred based on historical information. The inference is termed as context inference engine (CIE) and is used to classify the current context as an activity [15].

A context-aware system gathers sensor data and interacts with remote services for service discovery or delivery [12]. A context-aware system is composed of interacting modules. These modules include context gathering module; which gathers data from intrinsic sensors and sensor services as well as deduced data, context history module; that records the history of interactions, a context inference module; which classifies the current context with a degree of confidence using a machine learning algorithm, and an adaptation module; that interacts with remote services.

The task of a context-aware system is to gather context and map it to a correct activity label. This requires communication, storage, and classification process. The proliferation of devices increases connectivity and introduces new challenges in establishing a context-aware system. The prime concern is power as the handheld devices are battery operated. The effect of battery use has been profiled by researchers. A simple 1-2 mW sensor can consume up to 180-300 mW power when the data is sensed and processed [16, 17] showing inefficiency of power consumption. Furthermore, the power consumed is dependent on the combustion of fossil fuels which leads to an increase in carbon emissions.

Context is the collection of attributes describing a state or situation [11]. These attributes are gathered from sensors and sensor services. The sensor data can be deduced to generate more information as part of the context [7]. Context is given activity labels that correspond to the activity or situation. Power information includes the voltage and current used by a context-aware system. This voltage and current can be used to deduce power consumption [18]. The power information includes all attributes that describe the power consumption.

Thus, there is a need to explore a power-conserving context-aware system that provides acceptable activity classification while consuming less power. A power-conserving context-aware system is also termed as green context-awareness (GCx). To develop this system, an appropriate data set is required for the training of classification algorithms. Many datasets are available for activity recognition; however, there is a need to include the power or battery drain information in the data sets. The authors have compiled a data set that includes context information as well as power information by using a smartphone interacting in IoE environments. This also includes the deduced contextual information via remote services and the activity label set by the user. An Android app is developed using MIT AI2 App Inventor [19]. The dataset compiled is useful for measuring the effectiveness of a power conserving context aware system and enables GCx.

The rest of the paper is organized as follows. Section 2 presents contemporary related work and highlights the motivation of this work. Section 3 introduces the concept of smartphone and its relationship with remote services within an IoE environment. Section 4 presents the concept of power awareness and its phases. Section 5 outlines how power and energy are measured. Section 6 presents how power can be profiled using models and real-time measurement as found in the literature. Section 7 presents how context and power information is gathered using a smartphone as a prelude to the compilation of a dataset. Section 8 discusses the results, and the paper concludes in Section 9.

2. Related Work

Over the years, many researchers have gathered and published data sets that aid in developing context-aware applications. These data sets have been based on wearable devices as well as smartphones. The recent advancements in the design and development of smartphones have embedded a large variety of sensors previously worn on the body. The authors of this paper have selected recent data sets to identify their limitations thus highlighting the significance of the current work. These are discussed in subsequent paragraphs.

Ilarri et al. have reviewed contemporary data sets for context-aware systems [20]. The contextual information gathered in the focus datasets is used by content delivering services including Netflix and Amazon. The datasets reviewed by Ilarri et al. include contextual information that can be harvested using a smartphone, history of interaction, and remote services. Different subsets of these datasets are used for trip planning, movie recommendation, and music recommendation. Some of the contextual attributes include location, time, mood, temperature, sleepiness, crowd information, and nearby family and friends. However, none of these data sets maintain power consumption information or even the battery level of the smartphone. While these datasets are useful in developing smart applications, the application would not consider power as a constraint.

Recently, the concept of the smart grid has been advanced to include energy systems. This has resulted in energy 4.0, a standard where renewable energy resources are managed in smart grids. Shahinzadeh et al. have proposed the concept of Internet of Energy where smart grids are developed for energy management [21]. This is a useful concept for managing energy resources from the perspective of smart cities [22].

Sariaslani has published a Python-based context-aware system for tourists who want to visit attractions in a city [23]. This system uses location through latitude and longitude, history of visits, and the user's context to recommend visit locations in a city. The attractions are selected using an image set of attractions and tourist locations in London. The images are used to extract the owner, occupation, tags, number of favorites, and location. This information is matched with the contextual data to generate recommendations for tourists. This dataset uses the clock and GPS sensor to compile contextual information but lacks power-related information that could aid in the development of a power-conserving context-aware system.

Morgan et al. have developed a prefiltering and heuristic-based context recommendation system tested on LDOS-CoMoDa and DePaul datasets [24]. However, there

is no power information in the datasets selected to test the effectiveness of the proposed techniques.

Chang et al. have developed a tree-based recommender model as a context-aware system using AmazonBooks and Taobao User Behavior dataset [25]. The mechanism for context-based recommendation is promising; however, the datasets used neither consider power information nor the proposed technique consider power as a constraint.

Jeong and Kim have recently used deep learning to enhance the performance of context-aware systems while minimizing the effect of data sparsity [26]. The deep learning model learns contextual information based on multiple data sets. However, the deep learning model still does not consider power information as a contextual feature, mainly because it is beyond the scope of work by the authors.

The closest power information is part of the CARS dataset compiled by Unger et al. using smartphones to gather contextual information [27]. The datasets include battery level, battery temperature, and battery status, i.e., plugged or unplugged. However, this battery information is not suitable to ascertain the amount of power consumed during context gathering. The battery level is just the remaining percentage at the time of context gathering. This cannot be used to develop power constraint context-aware systems. Furthermore, the battery level is dependent on the type, capacity, and wear-and-tear of the battery and is not a good measure of power consumption during context gathering. The newer smartphones do not show a change in battery level when the context is gathered because the process takes fractions of a second. It is necessary to measure average voltage and average current to effectively know the power consumed as wattage.

The related work shows that the main focus has been gathering sensor data and deduced data. The complete power information is missing in the compiled data sets. The authors have compiled a dataset that overcomes this limitation by recording power information including voltage, current, and power with each record. This not only introduces the concept of power as a part of the context but also provides a means to establish the effectiveness of a power-conserving context-aware system, interacting in an IoE environment.

2.1. Comparison of Contemporary Datasets. Table 1 presents a comparison of contemporary datasets based on context information, use of smartphones to collect data, and power information. The statistics of the dataset are not presented as this paper is concerned with the type of data rather than the quantity of data.

2.2. Motivation. While the contemporary datasets are rich in contextual information, they simply ignore the power information. This entails that the context-aware systems developed using these datasets would assume that context awareness is not power constrained. To learn the effect of power on context-aware processing, the power information must be logged with each record in the dataset. Logging battery level is not suitable since this is a misleading ratio. In newer batteries, the level does not change even when the smartphone is in use. It is appropriate to measure the voltage and current to calculate power consumption.

Table 1: Comparison of datasets based on contextual and power information.

Authors	Year	Smart phone	Contextual information	Power information		
				Battery level	Current	Voltage
[20]	2018	✓	✓	×	×	×
[27]	2018	✓	✓	✓	×	×
[23]	2020	✓	✓	×	×	×
[24]	2020	✓	✓	×	×	×
[25]	2021	✓	✓	×	×	×
[26]	2022	✓	✓	×	×	×

3. Smartphones in an IoE Environment

A smart environment is composed of sensors, devices, and remote services. At the heart of a smart ecosystem is a smart device that polls sensors and communicates with remote services. A smart device polls the sensors and remote services and gathers contextual data or context. This context is then transformed, stored, and recognized which is termed as context processing.

A smart personal space is a collection of devices owned or temporarily controlled by a user. A user may own multiple smart devices including wearable and handheld devices. Since these devices are controlled by a user, they are termed as a smart personal space. In addition, a smart personal space includes the private and public data of the user. This data can be maintained within the space or remotely accessed via the Internet.

A smart environment consists of multiple smart spaces interacting with the appliances and remote services [28]. This is analogous to multiple patients in a hospital, multiple persons in a house, engineers in a lab, or colleagues in an office. Within a smart environment, multiple smart personal spaces interact with each other as well as appliances and remote services. Since multiple personal spaces interact with the services using a common channel, conflicts in access can arise. A conflict resolution mechanism is provided in a distributed fashion where remote services and appliances can prioritize service delivery. Figure 1 shows a model of a smart ecosystem.

With the development of system on chip (SoC) based fog environments, IoT has become IoE where devices, sensors, and appliances are all connected, and computation is carried out within the fog. With a higher dependency on fog for computation, mapping, or learning tasks, the IoT world has transformed into artificial intelligence of things (AIoT). The difference between IoE and AIoT is the localized computation and decreased use of bandwidth for cloud storage. AIoT is relatively smarter and can provide abstractions for machine learning and big-data. Hassani et al. have proposed the context to be implemented as a service within a smart environment [29]. The context is processed by a context

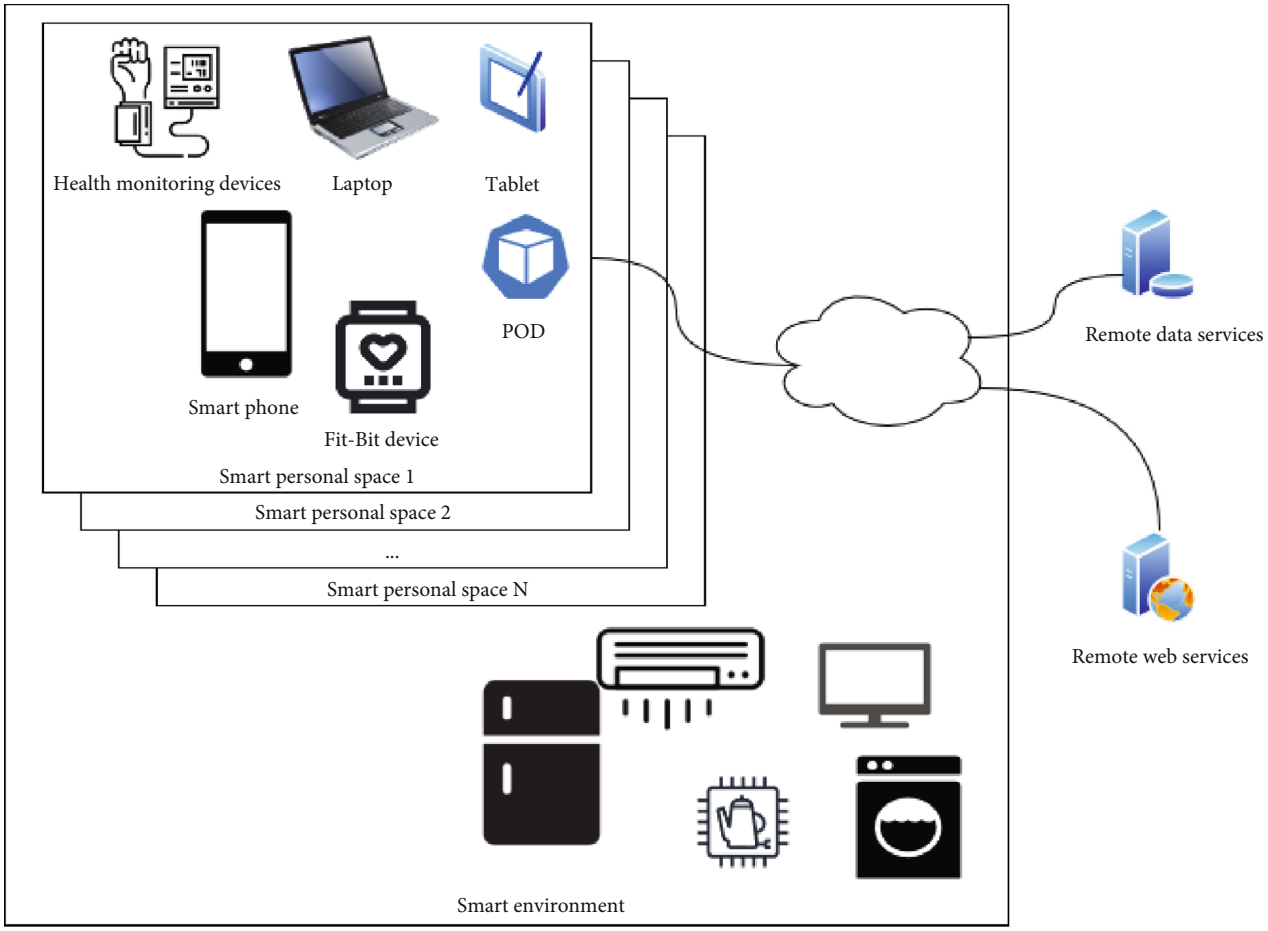

Figure 1: A smartphone in an IoE environment.

management platform (CMP), which is responsible for context processing and its provision to context consumers.

4. Power Awareness

Power awareness is the ability of a computer system to provide the same service albeit reduced performance while conserving power. An efficient system might not be a power-aware system as the efficiency parameters would not consider power conservation. Power awareness can also be viewed as a metric for cost of context (CoC) in an IoE environment [30]. The CoC is based on quality of context (QoC) which provides quality parameters to each contextual information. Power consumption can be considered as a cost parameter that could be used for triggering context processing. Power awareness is a combination of two closely linked concepts discussed below:

4.1. Power Consumption and Profiling. Power consumption is the energy consumption by a computation machine to provide a service. The energy is in the form of electrical power distributed through AC mains or DC batteries. A battery-based system needs to be charged regularly while a mains-based system is dependent on the supply of electrical current. Smart handheld devices are mostly battery operated and use lithium-ion (Li-ion) as the implementation technology [31, 32].

The energy capacity of a battery is represented in watt-hours (Wh) while the operating voltage is 3.8 V. Another method is to use milliampere-hour (mAh) to represent the energy stored in the battery. It is interesting to note that a discharging battery would also lose voltage though for most of the operation it remains above 3.5 V.

The total consumption measured over a period is termed power profiling. The profile can be measured as a simulation as well as in real-time measurement [33]. Power profiling is mostly measured at the device or the OS level but can be fine-grained at the application level. Specialized hardware installed on the chip allows fine-grained measurement; however, the mundane device can use battery drain measures over a period to identify power consumed.

As the first step in power awareness, the power profile needs to be measured over a period [34, 35]. This allows the identification of the power consumption characteristics of a device or an application [36, 37]. Within a smart device, the power is consumed in the communication, storage, and computation phases of a context-aware application. Depending on the environment and the application focus, the profiles can be different. However, a general profile can be formed which can be fine-tuned over time.

Table 2: Description of tasks in the BPMN model of PowerIpsum.

Task	Actor	Description
Poll sensors and record timestamp, current, and battery level.	Smart device	This task records the current timestamp and polls the sensor embedded on the device. For the missing sensors, a null value or zero value is recorded. Furthermore, the average current, voltage, and battery level are also recorded.
Access weather service		The weather service is accessed using latitude and longitude provided by the GPS sensor.
Receive weather data		The weather information is received and appended to the recorded sensor data. This includes the current weather report.
Access map service		The location service is accessed using latitude and longitude provided by the GPS sensor.
Receive map data		The location information is received and appended to the recorded contextual information. This includes the current location.
Record timestamp, current, and battery level		The timestamp is recorded again so that elapsed time can be measured. The average current and battery level are again recorded. It has been observed that for the latest smartphones and using newer batteries, the battery level remains unchanged. Also, the average current is the same because the elapsed time is a fraction of seconds.
Get user label		The user types in the activity label to complete the context record. The record is then appended to the context file, which can be shared through email, messaging, or file transfer for postprocessing.
Weather service	OpenWeatherMap	This is a free and online web service that provides the current weather report based on latitude and longitude. The weather report includes weather outlook, temperature, precipitation chances forecast, and other attributes. The service is accessed using the HTTPS/GET method.
MAP service	MAPQUEST	This is a free and online web service that provides location information based on latitude and longitude. The information includes nearby places as well as location tags including street name, locality, and city. This service is accessed using the HTTPS/GET method.

4.2. Power Conservation. The final step in power awareness is power conservation. Power conservation is the ability of a system or application to provide the same service while conserving power. Power conservation is a nonfunctional requirement like time and space efficiency. Greener systems are power efficient in addition to time and space efficiency. Consider a case of a supercomputer that is time and space-efficient and consumes power of the order of megawatts (MW). Such computation does not conserve power and increases the carbon footprint as a byproduct.

A system that can measure the power profile and predict battery life needs to provide conservation to increase battery life while providing the same service. Researchers have proposed many techniques that are presented in the next section. The means to measure conservation also needs to be established to ascertain the effectiveness of a power conservation technique.

5. Power and Energy Measurement

Power and energy have been used interchangeably in the literature. Power is the product of voltage and current, measured in watts while energy is the capacity of doing work measured in Joules. Energy can also be measured in watt-hours, by multiplying power with elapsed time. Since it is easier to measure power using a multimeter, and through power profiling apps, this work considers power in watts and energy in watt-hours. Equation (1) gives the power equation with power (P) measured in watts (W) as a product of voltage (V) and current (I).

$$P = V \times A. \tag{1}$$

Equation (2) represents the electrical energy (E) measured in watt-hours (Wh) as a product of power (P) and time (t).

$$E = P \times t. \tag{2}$$

A watt in terms of energy is when the energy of a Joule (J) passes through an electrical device for a second. Watts can then be measured in Joules per second (J/s). Energy potential is measured in Joules by replacing P with J/s in equation (2). A variation of electrical energy can be represented by combining Ohm's Law in Equation (2). Equation (3) shows the dissipated electrical energy as a product of the square of the current (I), time (t), and resistance (R). This is also measured in Wh.

$$E = I^2 \times R \times t. \tag{3}$$

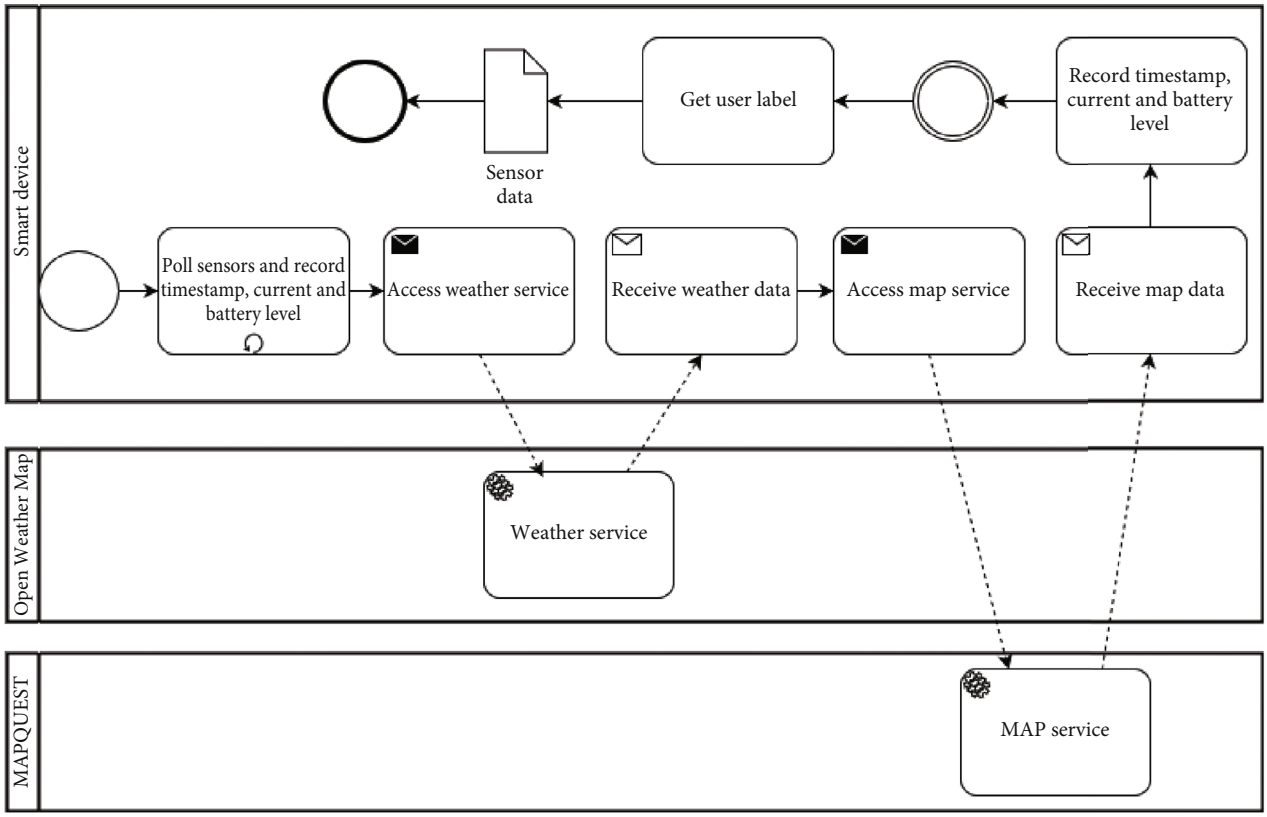

Figure 2: Design of PowerIpsum App using BPMN.

6. Power Profiling and Optimization

Power profiling refers to the measurement of power consumption over time. Power profiling is a useful tool to monitor the battery usage of mobile devices. All modern OS support power profiling at the OS level and rank power-draining applications [38]. A human can then tweak applications based on their power profile. Profilers measure the current draw over time and share their results in the form of plots. Jofri et al. have surveyed profiler applications for smartphones [39].

Tsao et al. have modeled the power consumption of I/O operations by implementing a power profiling [40]. This system monitors I/O operations on communication modules and touch screens. Sathyamoorthy et al. have focused on the power profiling of data communication by smartphones in a smart environment [41]. Both authors have reported that the communication modules consume more power, especially in noisy networks. Bolla et al. and Celenlioglu et al. have surveyed power consumption in networks, independently [42, 43]. Brienza et al. carried out power consumption of file-sharing P2P networks [44]. The major power consumer is the communication module. Ismail et al. point out that multimedia applications and screens consume a high amount of power [45]. This can be adjusted based on battery drain. Asnani et al. have recommended that contrast enhancement can reduce power consumption by OLED-based screens [46]. Vasile et al. have reported that the front-end applications consume more power when compared with back-end components and external web services that are invoked by the back-end components [47]. The authors have profiled power consumption in Windows phones.

Power is a constraint for battery-operated devices. Contemporary technology has paved the way for reducing the size and weight of smart devices. However, power consumption remains a pressing issue in the design and development of smart devices. Both hardware and software need to be optimized for power consumption [48]. While energy-efficient hardware has been designed, there is a need to address power constraints in algorithm design and software development [49]. The optimization of software applications for power usage is a requirement for green computing [50, 51].

Duan et al. have outlined a power estimation model across three layers. These include the hardware layer, the OS layer, and the application that is in the running state [52].

Chang et al. have proposed residual power as the unit of forecast and optimization. Residual power gives the estimated battery life of smartphones and power banks [53]. Lee et al. have profiled the power consumption of context-aware applications [54]. The authors report power is consumed in two stages. First, when the context is gathered through the sensor and then when it is processed. Context processing is more power-intensive based on the algorithm used to classify context. Rault et al. point out that the continuous sampling of real-time sensor data drains the battery quickly [55].

```
STEP #      pseudocode
1   int numberOfAttributes; //total number of attributes
2   double sensorData[numberOfAttributes]; //array for sensor data
3   URL weatherURL;
4   URL mapURL;
5   file contextFile;
6   procedure init() {
7      mapURL = new URL("https://developer.mapquest.com/);
8      weatherURL = new URL("https://api.openweathermap.org");
9      contextFile = new file("contextualData.Txt");
10  }
11  procedure powerIpsum (){
12     double timestampBefore = getCLK();
13     double avgCurrentBefore = getAverageCurrent();
14     double batteryLevelBefore = getBatteryLevel();
15     for (int i =0; i < numberOfAttributes; i++) {
16        sensorData[i] = attributeValue; //value of attribute
17     }
18     HttpURLConnection con1 = weatherURL.openConnection();
19     string weatherRep = con1.setRequestMethod("GET", lat, long);
20     HttpURLConnection con2 = mapURL.openConnection();
21     string mapRep = con2.setRequestMethod("GET", lat, long);
22     double timestampAfter = getCLK();
23     double avgCurrentAfter = getAverageCurrent();
24     double batteryLevelAfter = getBatteryLevel();
25     string activityLabel = inputUserLabel(); //user types data
26     contextFile.Append(timestampBefore, avgCurrentbefore, batterylevelbefore, sensordata[], weatherRep, mapRep, timestamp-
After, avgCurrentafter, batteryLevelafter, activityLabel, "endLine");
27  }
```

Pseudocode 1: Pseudocode of PowerIpsum process.

Abkenar et al. have proposed a power measurement method during group activity recognition. This is carried out by measuring the current draw of a smartphone from the battery [56]. Bernal et al. have outlined energy consumption reduction strategies based on power profiling [57]. These strategies can potentially optimize battery life in context-aware applications. Similar techniques can be utilized for BEV [58].

Power profiling can be used to optimize device behavior and conserve power. Devices can reduce screen intensity, turn off Wi-Fi/5G/4G modules, and shift to power-saving modes as required. Devices can also inform the users on which applications are consuming more power and can be suspended based on users' input. The optimization is carried out at the device and OS level and has little adaptability. This scheme lacks application-level power conservation. The measurement of power consumption is the basis of power profiling [33].

6.1. Model-Based Profiling. In model-based profiling, software models are used to measure power consumption. This method is useful at design times where the power considerations can be incorporated at the device or OS level. Application development can also utilize this method to improve the development constraints.

An implication of model-based profiling is the comparison of different algorithm approaches based on their power profiling, like time and space complexity. An algorithm that consumes less power and produces acceptable results is greener. A greener algorithm can be selected for development by engineers to reduce power consumption.

6.2. Real-Time Measurement-Based Profiling. In contrast to model-based power profiling, real-time measurement-based profiling, measure power consumption in real-time. This method has the added advantage of live measurements and can be used by the OS to identify applications that consume less power. González-Pérez et al. have listed guidelines for reliable measurements using Android-based devices [59].

UPower is used in Linux distributions to measure power consumption, voltage level, energy, and battery level [60]. The battery manager extension is provided for use on Android devices [61]. Battery manager extension provides voltage, current, health, temperature, and level information. Both these mechanisms are software-based. This method requires may require hardware support to accurately measure the power consumption. Datta et al. have developed an application that monitors the power consumption of smartphones and logs the usage patterns [62, 63].

The hardware-based measurement uses Summit SMB347 or Maxim MAX17050 for Nexus devices. An external power monitor can also be attached to a Nexus device for real-time profiling [64]. Mukherjee and Chantam have developed a device-level mechanism to measure the total power

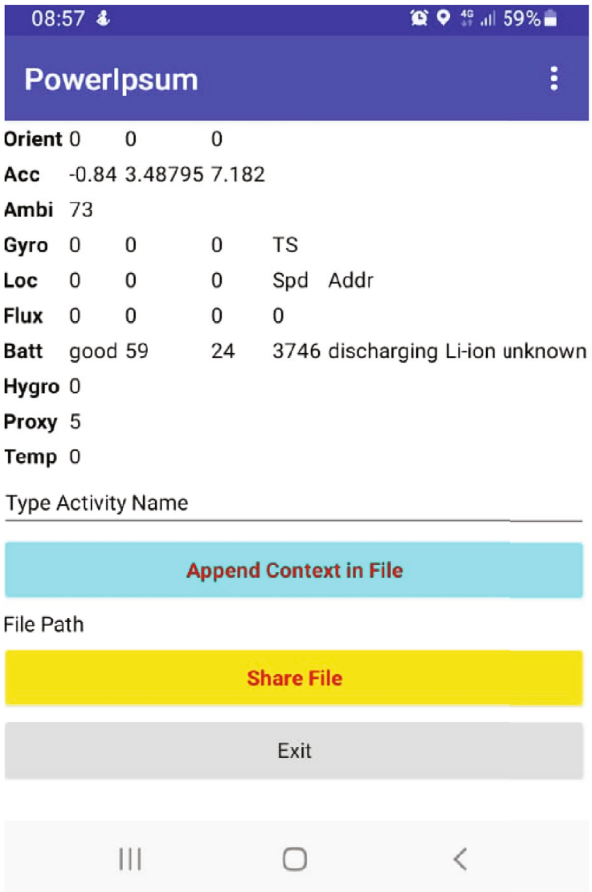

Figure 3: PowerIpsum application.

consumption of a Nexus 6 smartphone [65]. Hardware-based approaches are expensive and not available in mundane and widely distributed devices. Tsao et al. focused on I/O operations in mobile devices and profiled power consumption during these operations [40]. In contrast, a software-based mechanism approximates power consumption and is easily available in mundane devices [66].

Schwartz et al. have proposed the use of floating-point operations (FPO) as the measurement method for power consumption [67]. This mechanism is expensive and requires fine-tuned measurements.

7. Gathering Contextual Data and Power Information

The context data is first gathered from the sensor services present in the environment. The data is acquired by accessing the internal sensors as well as remote sensor services. For example, a smartphone can use the internal pedometer, gyroscope, accelerometer, light sensor, etc., as internal sensors while it could access a weather service over the cloud as a remote sensor service. Traditionally, the context congregator module or the context aggregator module of a context-aware system gathers and stores the contextual data [14]. The contextual data can be stored using RDF, XML, and OWL and can be organized in many ways [13]. There is no context processing at this level. Context processing classifies the activity of the contextual data and history of context using a machine learning algorithm [12, 15]. The first step is to gather data and store it on a device. The user should supply the context label, to aid in classification. The context label of context is the user's label assigned to the context. The label could be on a macro level or micro level, e.g., "Driving" is the context label at the macro level, while "Driving to Carrefour to Buy Groceries" is a label at the microlevel.

7.1. An Example Case. An example case of using PowerIpsum is the data collection of mundane and daily tasks performed by a user. The device captures the sensor data through the sensors embedded in the users' smartphones. This information can be used to access remote services. For example, GPS data is used to get weather and location information through two web services. In addition to contextual data through the sensors, the device also records the elapsed time, average voltage, and average power. For example, a user when going to work uses the app to record the contextual information. The sensor data is recorded, and the remote services are accessed through 4G connectivity. The power consumption is dependent on the signal strength of the network. Remote services are also accessible through Wi-Fi; however, more power is consumed in retransmissions due to low signal strength or noisy channels. The power is calculated in a postprocessing fashion.

7.2. Context Gathering App: "PowerIpsum." To gather contextual data, the authors of this paper have developed an application for Android platforms using MIT App Inventor. MIT App Inventor provides a quick way of developing Android applications and provides a visual component-based interface. A developer needs to drag and drop components and develop their logic around the components. MIT App Inventor has reduced development time, is suitable for use by all ages, and is free to use. The developed application is called as PowerIpsum and is available for download on GitHub under GNU GPL v3.0 License for public use [68]. MIT App Inventor provides a wide array of sensors that may be installed on Android phones. These include clock, gyroscope, accelerometer, ambient light sensor, proximity sensor, GPS, temperature, battery info, orientation, and magnetometer. The testbed is Samsung A03s. Google Nexus 6, Samsung A51, and Samsung S20 are used for application testing purposes, and limited data is collected from these devices. All phones run Android OS. Google Nexus 6 has a built-in power management chip installed that facilitates real-time power measurement, while Samsung A03s, and A51, approximate the power usage using software tools. The data from some of the sensors is raw and needs to be deduced using an online web service to make it meaningful. For example, the GPS sensor returns GPS coordinates in the form of the latitude and longitude of the device. This GPS data can be used to access a weather service and a map service to get the weather information and address information simultaneously. OpenWeather and MapQuest are used as web

Table 3: Description of sensor data attributes.

Sensor	Attribute	Value type	Description
CLK	Timestamp	Long	This timestamp is a label to record the time at the start of the process. This timestamp is a long number depicting a nanosecond. The timestamp is also recorded after the processing has been completed and the elapsed time is measured in picoseconds, postprocessing.
Orientation sensor	Azimuth	+ve double	It is used to measure the azimuth for the orientation of the device.
	Pitch	Double	It is used to measure the pitch for orientation of the device.
	Roll	Double	It is used to measure the roll for orientation of the device.
Accelerometer	X acceleration	Double	It measures the acceleration along the x-axis.
	Y acceleration	Double	It measures the acceleration along the y-axis.
	Z acceleration	Double	It measures the acceleration along the z-axis.
Ambiance	Lux	+ve integer	It measures the ambient illuminance in lux.
Gyroscope	X angular velocity	Double	It measures the angular velocity of the device's rotation along the x-axis.
	Y angular velocity	Double	It measures the angular velocity of the device's rotation along the y-axis.
	Z angular velocity	Double	It measures the angular velocity of the device's rotation along the z-axis.
Location sensor	Latitude	+ve double	Gives the latitude of the device.
	Longitude	+ve double	Gives the longitude of the device.
	Altitude	+ve integer	Gives the altitude of the device.
Magnetometer	X strength	Double	Gives the magnetic field strength along the x-axis.
	Y strength	Double	Gives the magnetic field strength along the y-axis.
	Z strength	Double	Gives the magnetic field strength along the z-axis.
	Absolute strength	Double	Gives the absolute strength of the magnetic field.
Battery extension	Health	String	The health status of the battery.
	Level	+ve integer	Gives the battery level as a percentage of total capacity. The before and after battery level is recorded.
	Temperature	Integer	Gives the temperature in Celsius.
	Voltage	+ve integer	Gives the voltage in millivolts (mV).
	Status	String	Gives the status, i.e., charging or discharging.
	Technology	String	The technology of the battery, e.g., lithium-ion, or nickel cadmium.
	Current draw	Integer	Measures the current draw in milliAmperes (mA). The current draw is negative when the device is discharging. This is recorded for both before and after processing.
Hygrometer	Humidity	Integer	Measures the humidity in the environment.
Near field communication sensor	Distance	+ve integer	Measures the near field distance of a compatible device.
Weather web service	Weather	String	Gives the outlook of the weather. The location is the latitude and longitude measured using a location sensor.
Location web service	Address	String	Gives the address of the location. The location is the latitude and longitude measured using a location sensor.

services since both are free for use [69, 70]. Table 2 describes the tasks, shown as a rounded rectangle, of the BPMN model of PowerIpsum. Pseudocode 1 shows the pseudocode of the PowerIpsum process. Since each sensor provides multiple attribute-value pairs, there could be more attributes than the total number of sensors embedded in a smart device. If the number of attributes is n, then, the time complexity of the pseudo code given in Table 2 is $O(n)$. This shows that the data gathering process has a linear time complexity.

The design is presented using business process model and notation (BPMN 2.0) in Figure 2. BPMN is a standard used to design a business model of interacting services. Since PowerIpsum is envisaged as a service that uses other services, a BPMN is suitable for design. Figure 2 shows that there are three interacting entities or services. The smart

Table 4: Statistics of Monte Carlo method on time elapsed and power consumed on different smart phones.

	A03s		A51		S20	
	Time (psec)	P (μW)	Time (psec)	P (μW)	Time (psec)	P (μW)
Average	11.84	30.09	6.79	2.97	6.5	1.57
Median	12	30	7	3	6.5	2
Mode	11	27	7	3	7	2
Max	26	55	12	6	10	3
Min	0	12	3	0	4	1
Std dev	4.83	6.58	1.41	0.95	0.91	0.5

device is the user's device and interacts with two web services. At the start, the timestamp and battery information are taken and then the internal sensors are polled. After that, the web services are called sequentially via message passing using the HTTP GET method. MIT App Inventor is a single-threaded, sequential app development system. There are no threads and no concurrent execution. The application developed is executed sequentially, as a constraint. The response by the services and the sensor data is stored as a record temporarily, as an intermediate event. The user then inputs the label in the application, and the record is stored in a comma-separated value (CSV) file. The process then completes. The saved file can also be shared via social applications. This aids in data collection purposes.

Figure 3 shows the front end of the PowerIpsum application. The sensor data is arranged in a grid and shown in a row for each sensor. The columns correspond to data values. This is shown to confirm that data is read by the sensors. Once the user types the activity name and presses append context in file, the sensor data is read and appended to a text file. The file can then be shared using social applications by the user. A user can exit the application by pressing the exit button. The application can run in the background with the measurements taken and stored only when the append context in file button is pressed. Depending on the sensor availability, the data is flushed on the application as well as appended to the file. However, it is not easy to confirm whether the data appended in the file is the same as the one flushed on the screen, as some sensors are sensitive and keep on reading data constantly. The timestamps and battery information are recorded when the user presses append context in file. This button gathers and stores context as well as a timestamp, battery information, and average current. Battery information includes battery level and voltage. Voltage and average current draw can be used to measure power as shown in equation (1). Timestamps can be used to find elapsed time and energy can be measured using power and elapsed time as shown in equation (2).

7.3. Data Fields of Contextual Data Set. The data fields or attributes of contextual data include the complete data provided by sensors, APIs, and web services that are accessed by the PowerIpsum application. Table 3 describes the attributes of the contextual data set and the sensors associated with it. The absence of a sensor on a smartphone shows a null value or 0 in the data record.

The postprocessing computations are carried out using both equations (1) and (2). Time elapsed is measured by subtracting the timestamps. The recorded data is exported to a spreadsheet using MS Excel. The postprocessing of the data set is performed using MS Excel and on a desktop. The time and power used for processing are not considered as this paper is concerned with the time and power consumed in data sensing and storing. The computations required are the time elapsed, power, and energy consumed. Among the targets is the correlation between time elapsed and power conservation. It is interesting to note that time elapsed and power consumption are both random variables that occur with an underlying and undiscovered probability distribution in a context-aware world. Similarly, the context label is also a random variable assigned by a user to a context.

7.4. Using Monte Carlo Method on Contextual Data. The data collection of sensor data for context purposes is a tiresome and time-consuming process. It also depends on the availability of test subjects who would wholeheartedly participate in the project. The privacy concerns of users become a hindrance in data collection. Furthermore, noise can also distort the data and lead to an incorrect conclusion. Since the variables of interest are random variables, an obvious question is what would be the correlation among them? Furthermore, can the power consumption be predicted, based on the time elapsed, with a degree of probability? In the absence of abundant and noise-free data, and a requirement to forecast the outcome, Monte Carlo method can be utilized on the gathered contextual data [71, 72]. The mean and standard deviation of time elapsed, power consumption, and energy computation are taken, and Monte Carlo Method is run to 500 and more instances.

8. Results and Discussion

The sensor data collection is an ongoing task. It is envisaged that the data would be gathered by willing subjects for over a year. Given that a subject provides 10 instances per day, this would collect 3500+ records in a year. An increase of subjects would simply increase the dataset. The collected data set has been uploaded on our institute web page and is regularly updated. Each record is checked for missing attributes, duplication, or null values, and all such records are discarded. Currently, the data set has records of a willing person who uses Samsung A03s for data collection. The same person has used multiple smartphones for testing and evaluation purposes; however, only one is used for extended data collection.

Monte Carlo method is selected for forecasting and modeling the behavior of stochastic processes. Monte Carlo method is widely used, is reliable, and computationally inexpensive [73]. It is also suitable when there is limited empirical data [74]. The collected data set is subject to Monte Carlo method to forecast time elapsed and power consumed. Mean, standard deviation, mode, max, and min after

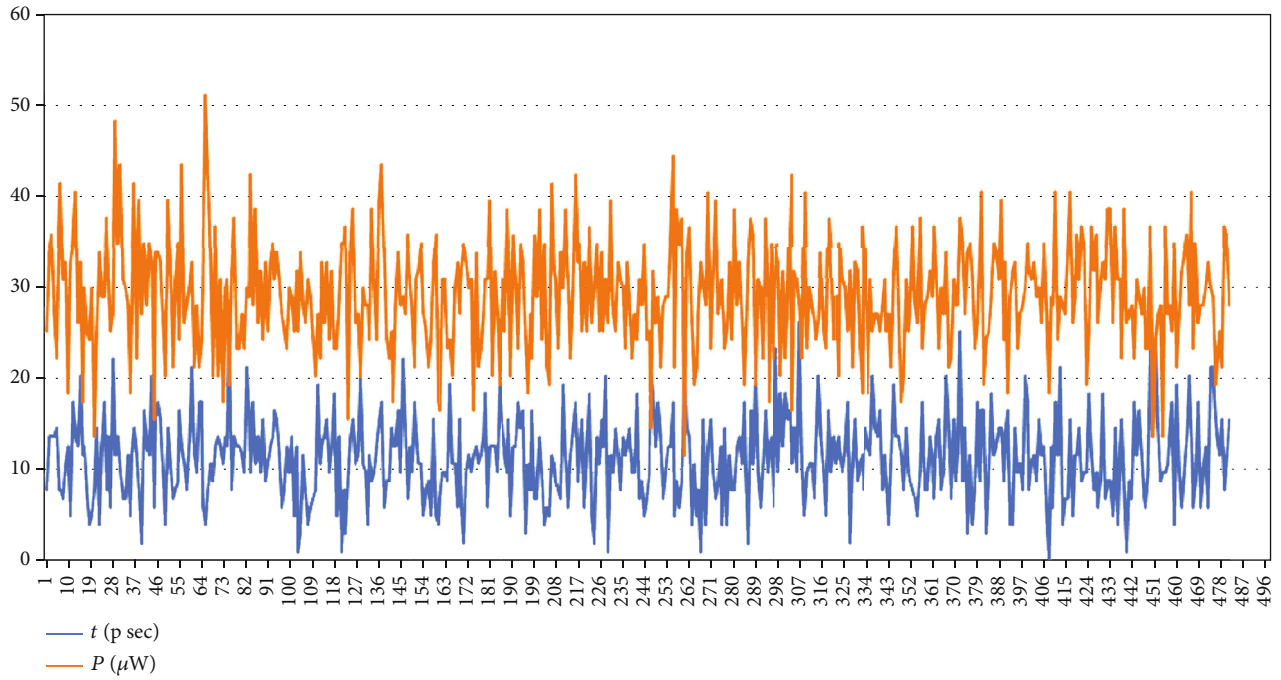

Figure 4: Plot of time elapsed and power consumed using Monte Carlo method using Samsung A03s.

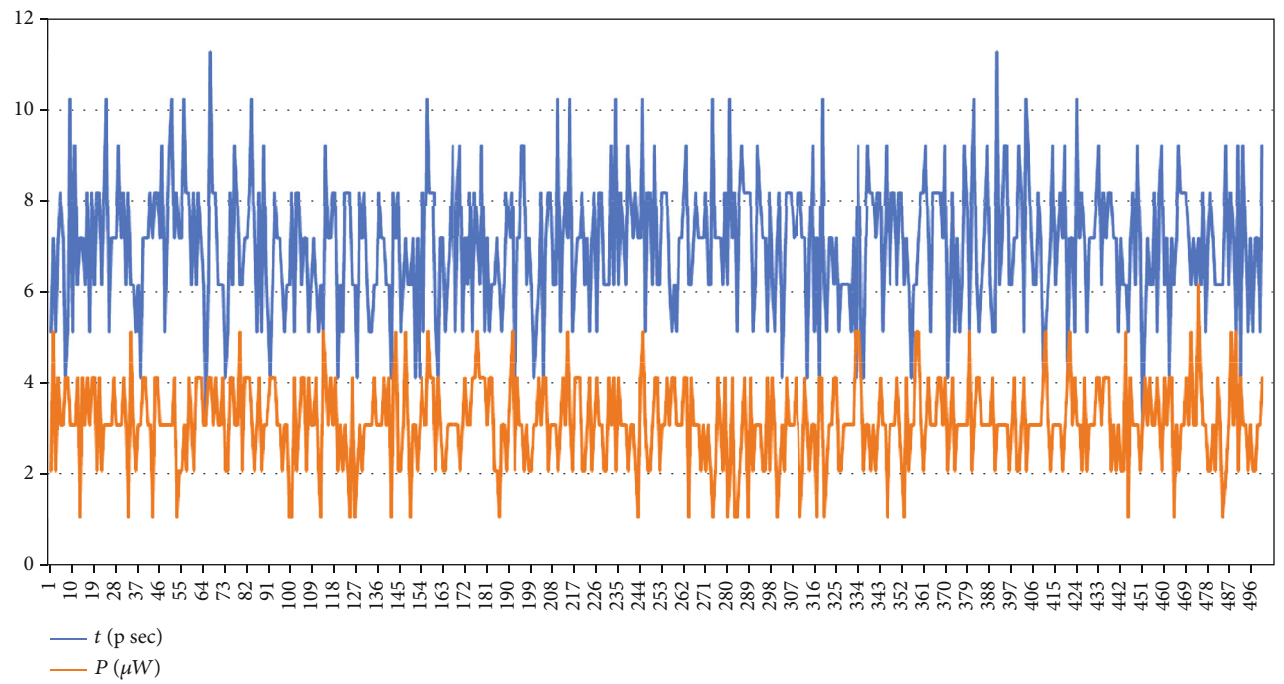

Figure 5: Plot of time elapsed and power consumed using Monte Carlo method using Samsung A51.

executing Monte Carlo method are recorded as well. The Monte Carlo method can produce negative numbers which are illogical in this case as time elapsed and power consumption are positive numbers. Absolute value is used to control illogical cases. Table 4 shows some statistics performed on running Monte Carlo method for time elapsed and power consumed, using 200+ records on Samsung A03s, and 20+ records on A51 and S20.

Figure 4 shows the plot of time elapsed and power consumed after running Monte Carlo method on data collected using A03s. Figure 5 shows the plot of time elapsed and power consumed after running Monte Carlo method on data

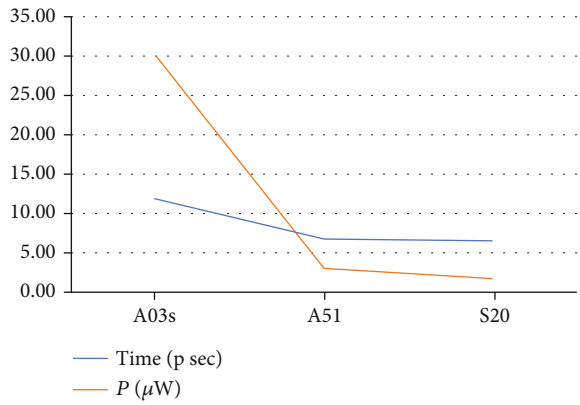

Figure 6: Comparison of time elapsed and power consumed after running Monte Carlo method for different smart phones.

collected using A51. While the data collection is less, the Monte Carlo method is executed 500 times in both cases.

The results show that while the time elapsed is similar, the power consumption is different for different data sets. This is because of the smartphone being used and the activities performed. The smartphones being used are of varying performance, which is evident in the power consumption. The power consumption is the average current draw from a smartphone. Since the gathering is performed for fraction of seconds, power as a measure is suitable. Figure 6 compares the time elapsed and power consumed for all three smartphones. It is interesting to note that the expensive and efficient phones took less amount of time and consumed less power. However, the smartphone that has lesser sensors and weaker performance took more time and consumed more power. One of the reasons is the underlying architecture and mobile usage. The data used in Figure 6 is derived from Table 2.

9. Conclusion

There is no information on power or battery usage in the datasets accessible on the UCI Machine Learning Repository or Google Research. Furthermore, the datasets that are used for context awareness are composed of sensor-based data and associated high-level activity. This research aims to collect contextual data as well as power information to help in the creation of a power-conserving context-aware system. The research domain is limited to users' mundane activities performed with a smartphone in an IoE setting. The authors used MIT App Inventor to create PowerIpsum, an Android application that captures sensor data as well as battery information including current and voltages. After that, a user tags the records with activity labels. The dataset is then inspected, compiled, and regularly updated. Monte Carlo method is used on the dataset to determine the correlation between time elapsed and power consumption when contextual data is gathered and stored. The results show that on average the power consumption is higher for older devices while the time elapsed is almost the same. Future work will entail increasing the dataset size and employing various machine learning algorithms for context awareness.

Data Availability

The authors confirm that the data generated or analyzed and supporting the findings of this study are available within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] World Economic Forum, "Fourth industrial revolution," 2021, https://www.weforum.org/focus/fourth-industrial-revolution.

[2] Statista, "Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025," 2020, https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[3] K. L. Lueth, "IoT analytics," 2022, https://www.mdpi.com/2078-2489/5/4/612.

[4] AlliedTelesis, "Designing networks for IoT," 2021, https://www.alliedtelesis.com/en/blog/designing-networks-iot.

[5] Cisco, "Internet of things," 2016, https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf.

[6] S. Rizvi, R. Pipetti, N. McIntyre, J. Todd, and I. Williams, "Threat model for securing internet of things (IoT) network at device-level," *Internet of Things*, vol. 11, pp. 100240–100252, 2020.

[7] N. A. Malik, U. Mahmud, and M. Y. Javed, *Future Challenges in Context Aware Computing*, VillaReal, 2007.

[8] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, pp. 85–90, Washington, DC, 1994.

[9] U. Mahmud, N. Iltaf, and F. Kamran, "Context congregator: gathering contextual information in CAPP," in *Proceedings of the International Conference on Frontiers of Information Technology*, Islamabad, Pakistan, 2007a.

[10] S. Hussain, U. Mahmud, and S. Yang, "Car e-talk: an IoT-enabled cloud-assisted smart fleet maintenance system," *IEEE Internet of Things Journal*, vol. 8, no. 12, 2021.

[11] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pp. 304–307, Karlsruhe, Germany, 1999.

[12] U. Mahmud and M. Y. Javed, "Context inference engine (CiE)," *International Journal of Advanced Pervasive and Ubiquitous Computing (IJAPUC)*, vol. 4, no. 3, pp. 13–41, 2012.

[13] U. Mahmud, "Organizing contextual data in context aware systems: a review," in *Handbook of Research on Human-Computer Interfaces, Developments, and Applications*, J. Rodrigues, P. Cardoso, J. Monteiro, and M. Figueiredo, Eds., pp. 273–303, IGI Global, Hershey, PA, 2016.

[14] U. Mahmud, N. Iltaf, A. Rehman, and F. Kamran, *Context-Aware Paradigm for a Pervasive Computing Environment (CAPP)*, VillaReal, Portugal, 2007.

[15] U. Mahmud and M. Y. Javed, "Context inference engine (CiE): classifying activity of context using Minkowski distance and standard deviation-based ranks," in *Systems and Software*

Development, Modeling, and Analysis: New Perspectives and Methodologies, pp. 65–112, IGI Global, 2014.

[16] S. L. Kiani, A. Anjum, N. Antonopoulos, and M. Knappmeyer, "Context-aware service utilisation in the clouds and energy conservation," *Journal of Ambient Intelligence and Humanized Computing*, vol. 5, no. 1, pp. 111–131, 2014.

[17] O. Yuryur, "Energy efficient context-aware framework in mobile sensing," in *University of South Florida, Electrical Engineering*, University of South Florida, Florida, 2013.

[18] U. Mahmud, S. Hussain, and S. Yang, "Power profiling of context aware systems: a contemporary analysis and framework for power conservation," *Wireless Communication and Mobile Computing*, vol. 2018, article 1347967, pp. 1–15, 2018.

[19] MIT, "MIT app inventor," 2012, https://appinventor.mit.edu/.

[20] S. Ilarri, R. Trillo-Lado, and R. Hermoso, "Datasets for Context-Aware Recommender Systems: Current Context and Possible Directions," in *2018 IEEE 34th International Conference on Data Engineering Workshops (ICDEW)*, pp. 25–28, Paris, 2018.

[21] H. Shahinzadeh, J. Moradi, G. B. Gharehpetian, H. Nafisi, and M. Abedi, "Internet of energy (IoE) in smart power systems," in *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, pp. 627–636, Tehran, 2019a.

[22] H. Shahinzadeh, J. Moradi, G. B. Gharehpetian, H. Nafisi, and M. Abedi, "IoT Architecture for Smart Grids," in *2019 International Conference on Protection and Automation of Power System (IPAPS)*, pp. 22–30, Iran, 2019b.

[23] A. Sariaslani, "Context-aware recommender," 2020, https://www.kaggle.com/amiralisa/context-aware-recommender/.

[24] C. Morgan, I. Paun, and N. Ntarmos, "Exploring Contextual Paradigms in Context-Aware Recommendations," in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 3079–3084, Atlanta, 2020.

[25] D. Chang, J. Liu, Z. Xu, H. Li, H. Zhu, and X. Zhu, "Context-aware tree-based deep model for recommender systems," DLP-KDD 2021, Singapore, 2021.

[26] S.-Y. Jeong and Y.-K. Kim, "Deep learning-based context-aware recommender system considering contextual features," *Applied Sciences*, vol. 12, no. 1, pp. 45–52, 2022.

[27] M. Unger, B. Shapira, L. Rokach, and A. Livne, "Inferring contextual preferences using deep encoder-decoder learners," *New Review of Hypermedia and Multimedia*, vol. 24, no. 3, pp. 262–290, 2018.

[28] U. Mahmud, S. Hussain, A. J. Malik, S. Farooqui, and N. A. Malik, "Realizing IoE for smart service delivery: case of museum tour guide," in *smart systems design, applications, and challenges*, J. M. Rodrigues, P. J. Cardoso, J. Monteiro, and C. M. Ramos, Eds., IGI global, 2020.

[29] A. Hassani, A. Medvedev, P. D. Haghighi et al., "Context-as-a-Service Platform: Exchange and Share Context in an IoT Ecosystem," in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 385–390, Athens, 2018.

[30] K. S. Jagarlamudi, A. Zaslavsky, S. W. Loke, A. Hassani, and A. Medvedev, "Towards measurable efficient and effective metrics for quality and cost of context," *Modeling and Using Context*, vol. 4, no. Special Issue, pp. 1–6, 2021.

[31] D. Ferreira, A. K. Dey, and V. Kostakos, "Understanding human-smartphone concerns: a study of battery life," in *pervasive computing. Pervasive 2011. LNCS*, K. Lyons, J. Hightower, and E. M. Huang, Eds., Springer, 2011.

[32] M. Ue, K. Sakaushi, and K. Uosaki, "Basic knowledge in battery research bridging the gap between academia and industry," *Materials Horizons*, vol. 7, no. 8, pp. 1937–1954, 2020.

[33] Y.-F. Chung, C.-Y. Lin, and C.-T. King, "ANEPROF: Energy Profiling for Android Java Virtual Machine and Applications," in *2011 IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 372–379, Tainan, Taiwan, 2011.

[34] J. Kulk and J. Welsh, "A low power walk for the NAO robot," in *Proceedings of the 2008 Australasian conference on Robotics & Automation (ARCA-2008)*, pp. 1–7, Canberra, Australia, 2008.

[35] C. Seo, G. Edwards, S. Malek, and N. Medvidovic, "A framework for estimating the impact of a distributed software system's architectural style on its energy consumption," in *Seventh Working IEEE/IFIP Conference on Software Architecture, 2008. WICSA 2008*, pp. 277–280, Vancouver, BC, Canada, 2008.

[36] Y. Li, H. Chen, and W. Shi, "Power behavior analysis of mobile applications using bugu," *Sustainable Computing: Informatics and Systems*, vol. 4, no. 3, pp. 183–195, 2014.

[37] N. Vallina-Rodriguez, J. Shah, A. Finamore et al., "Breaking for commercials: characterizing mobile advertising," in *Proceedings of the 2012 Internet Measurement Conference IMC '12*, pp. 343–356, Boston, 2012.

[38] J. Wang, L. Feng, W. Xue, and Z. Song, "A survey on energy-efficient data management," *ACM SIGMOD*, vol. 40, no. 2, pp. 17–23, 2011.

[39] M. H. Jofri, M. F. Fudzee, and N. M. Ismail, "A survey on energy-aware profiler for mobile devices," in *Computational Intelligence in Information Systems.*, pp. 295–305, Springer, 2015.

[40] S.-L. Tsao, C.-K. Yu, and Y.-H. Chang, "Profiling energy consumption of I/O functions in embedded applications," in *Architecture of Computing Systems – ARCS 2013*, H. Kubátová, C. Hochberger, M. Daněk, and B. Sick, Eds., pp. 195–206, Springer, 2013.

[41] P. Sathyamoorthy, E. C.-H. Ngai, X. Hu, and V. C. Leung, "Profiling energy efficiency and data communications for mobile internet of things," *Wireless Communications and Mobile Computing*, vol. 2017, Article ID 6562915, 15 pages, 2017.

[42] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future internet: a survey of existing approaches and trends in energy-aware fixed network infrastructures," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, pp. 223–244, 2011.

[43] M. R. Celenlioglu, D. Gözüpek, and H. A. Mantar, "A survey on the energy efficiency of vertical handover mechanisms," in *International Conference on Wireless and Mobile Networks (WiMoN)*, Lyon, 2013.

[44] S. Brienza, S. E. Cebeci, S. S. Masoumzadeh, H. Hlavacs, Ö. Özkasap, and G. Anastasi, "A survey on energy efficiency in P2P systems," *ACM Computing Surveys*, vol. 48, no. 3, pp. 1–37, 2016.

[45] M. N. Ismail, R. Ibrahim, and M. F. Fudzee, "A survey on content adaptation systems towards energy consumption awareness," *Advances in Multimedia*, vol. 2013, Article ID 871516, 8 pages, 2013.

[46] S. Asnani, M. G. Canu, L. Farinetti, and B. Montrucchio, "On producing energy-efficient and contrast-enhanced images for

OLED-based mobile devices," *Pervasive and Mobile Computing*, vol. 75, article 101384, 2021.

[47] C. V. Vasile, C. Pattinson, and A.-L. Kor, "Mobile phones and energy consumption," in *Green It Engineering: Social, Business and Industrial Applications*, V. Kharchenko, Y. Kondratenko, and J. Kacprzyk, Eds., pp. 243–271, Springer, 2018.

[48] J. Flinn, "Extending mobile computer battery life through energy-aware adaptation," in *Carnegie Melon University, Computer Science Department*, Carnegie Melon University, Pittsburgh, PA, 2001.

[49] O. Landsiedel, K. Wehrle, and S. Gotz, "Accurate prediction of power consumption in sensor networks," in *The second IEEE workshop on embedded networked sensors, 2005*, pp. 37–44, Sydney, 2005.

[50] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual Machine Power Metering and Provisioning," in *Proceedings of the 1st ACM symposium on Cloud computing SoCC*, vol. 10, pp. 39–50, Indianapolis, Indiana, 2010.

[51] A. Pathak, Y. C. Hu, and M. Zhang, "Where Is the Energy Spent inside my App?: Fine Grained Energy Accounting on Smartphones with Eprof," in *Proceedings of the 7th ACM european conference on Computer Systems EuroSys '12*, pp. 29–42, Bern, Switzerland, 2012.

[52] L.-T. Duan, B. Guo, Y. Shen, Y. Wang, and W.-L. Zhang, "Energy analysis and prediction for applications on smartphones," *Journal of Systems Architecture*, vol. 59, no. 10, pp. 1375–1382, 2013.

[53] H.-C. Chang, C.-K. Chung, M.-W. Hung, J.-Y. Lai, and Y.-S. Su, "A power-managed method for mobile devices," *Recent Patents on Computer Science*, vol. 6, no. 1, pp. 41–46, 2013.

[54] M. Lee, D.-K. Kim, and J.-W. Lee, "Analysis of characteristics of power consumption for context-aware mobile applications," *Information*, vol. 5, no. 4, pp. 612–621, 2014.

[55] T. Rault, A. Bouabdallah, Y. Challal, and F. Marin, "A survey of energy-efficient context recognition systems using wearable sensors for healthcare applications," *Pervasive and Mobile Computing*, vol. 37, no. 2017, pp. 23–44, 2017.

[56] A. B. Abkenar, S. W. Loke, W. Rahayu, and A. Zaslavsky, "Energy Considerations for Continuous Group Activity Recognition Using Mobile Devices: The Case of GroupSense," in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 479–486, Crans-Montana, Switzerland, 2016.

[57] J. F. Bernal, L. Ardito, M. Morisio, and P. Falcarin, "Towards an efficient context-aware system: problems and suggestions to reduce energy consumption in mobile devices," in *2010 Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR)*, Athens, 2010.

[58] Q. Fang, X. Wei, and H. Dai, "A remaining discharge energy prediction method for lithium-ion battery pack considering SOC and parameter inconsistency," *Energies*, vol. 12, no. 6, pp. 987–1024, 2019.

[59] A. González-Pérez, M. Matey-Sanz, C. Granell, and S. Casteleyn, "Using mobile devices as scientific measurement instruments: reliable android task scheduling," *Pervasive and Mobile Computing*, vol. 81, pp. 101550–101619, 2022.

[60] R. Hughes, "UPower," 2007, https://upower.freedesktop.org/.

[61] Pura Vida Apps, "Battery manager extenion," 2010, https://puravidaapps.com/battery.php.

[62] S. K. Datta, C. Bonnet, and N. Nikaein, "Minimizing Energy Expenditure in Smart Devices," in *2013 IEEE Conference on Information & Communication Technologies*, pp. 712–717, Thuckalay, India, 2013.

[63] S. K. Datta, C. Bonnet, and N. Nikaein, "Personalized power saving profiles generation analyzing smart device usage patterns," in *2014 7th IFIP Wireless and Mobile Networking Conference (WMNC)*, pp. 1–8, Vilamoura, Portugal, 2014.

[64] Mostly-Tech, "Power monitor," 2015, https://mostly-tech.com/tag/nexus-6/.

[65] A. Mukherjee and T. Chantem, "Energy management of applications with varying resource usage on smartphones," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2416–2427, 2018.

[66] K. Naik, *A Survey of Software Based Energy Saving Methodologies for Handheld Wireless Communication Devices*, ASO, 2010.

[67] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[68] U. Mahmud and S. Hussain, "PowerIpsum," 2022, https://github.com/umarmahmud5/PowerIpsum.

[69] MAPQUEST Developer Network, "MAPQUEST," 2021, https://developer.mapquest.com/.

[70] OpenWeather, "OpenWeather," 2012, https://openweathermap.org/.

[71] D.-G. Chen and J. D. Chen, *Monte-Carlo Simulation-Based Statistical Modeling*, Springer, 2017.

[72] N. Chopin and O. Papaspiliopoulos, *An Introduction to Sequential Monte Carlo*, Springer, 2020.

[73] M. Sabouri and M. Darbandi, "Numerical study of species separation in rarefied gas mixture flow through micronozzles using DSMC," *Physics of Fluids*, vol. 31, no. 4, article 042004, 2019.

[74] H. Y. Guo, Y. C. Ding, Q. Chang, C. P. Zhu, Y. Zhao, and Q. Zhang, "The coordinated development between the internal audit guidance by the government audit departments and internal audit units: applying the principal component analysis and Monte-Carlo simulation," *Journal of Physics: Conference Series*, vol. 1419, no. 1, p. 012040, 2019.