Contributing to the Energy Efficiency of Smart Homes : An Automated Management Framework

Houssam Kanso

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

ÉCOLE DOCTORALE DES SCIENCES EXACTES ET LEURS APPLICATIONS (ED 211 SEA)

DOCTORAL THESIS

Contributing to the Energy Efficiency of Smart Homes: An Automated Management Framework

---

by

Houssam KANSO

Advisors: Ernesto EXPOSITO, *Université de Pau et des Pays de l'Adour - France*
Adel NOUREDDINE, *Université de Pau et des Pays de l'Adour - France*

Reviewers: Romain ROUVOY, Université de Lille / INRIA / IUF - France
Jean-Marc MENAUD, IMT Atlantique - France

Examiners: Congduc PHAM (**Jury president**), Université de Pau et des Pays de l'Adour - France
Chantal TACONET, Télécom SudParis - France

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

Defended on December 9, 2022

"Our greatest weakness lies in giving up.
The most certain way to succeed is always to try just one more time."

- Thomas A. Edison

Acknowledgements

I would like to express my gratitude to all the people who have contributed to the realization of my doctoral thesis, which would not have been possible without their support and guidance.

In particular, I would like to offer my special thanks to my supervisors Ernesto Exposito and Adel Noureddine for their valuable guidance and helpful suggestions during all three years of this work. Thank you for encouraging me to do all the scientific communications, following my progress, and providing me with insightful feedback.

Besides my advisors, I would like to thank my colleagues at the LIUPPA research laboratory for the suitable work environment and scientific gatherings. I would like to thank the SEA doctoral school and the Université de Pau des Pays de l'Adour. I would also like to thank all the members of the computer science department and the administrative staff at the STEE college of Anglet.

I would like to acknowledge the E2S UPPA project for their financial support and interest in promoting science.

I would also like to thank my friends in France and Lebanon for the support and laughs you brought in difficult moments, especially Reine Abou Sleiman, Ana Torres, Jean-Raphael Richa, Fouad Al Achkouty, Elie Al Chicha, Mamadou Lamin Gueye, Houssam Noureddine, Hassan Kanso, Lina Dergham, and Mohammad Hussein Nasser.

Last but not least, I would like to thank my beloved family. Words cannot express my gratitude to my mother Zahia and my father Jihad. I am deeply grateful for your sacrifices, unconditional love, and consistent support to reach my dreams. As for my two adorable sisters Zeina and Rim, thank you for your enduring support and encouragement. I would also like to thank my brother-in-law Hadi and my newborn nephew Karam.

Abstract

In recent years, the power consumption of Cyber-Physical Systems (CPS) has been increasing due to the increasing number of connected devices (e.g., smart appliances, plug-and-play IoT devices...), mainly in the residential sector. A large number of devices integrate sensors allowing them to produce data describing the state of a device or the behavior of a person (e.g., temperature sensor, presence sensor...). In addition, numerous devices are equipped with actuators capable of accomplishing tasks impacting the environment (e.g., light control, heating, ventilation, air-conditioning system...). These devices have the potential of collecting a large amount of data that can be useful for power estimation and management. However, current energy management approaches are mostly applied to limited types of devices in specific domains and are difficult to implement in other scenarios. They fail when it comes to their level of autonomy, flexibility, genericity, monitored metrics, and heterogeneity of studied devices.

To address these shortcomings, we present, in this thesis, an energy management approach for connected environments based on generating power estimation models, representing a formal description of energy-related knowledge, and using reinforcement learning (RL) techniques to accomplish energy-efficient actions. We illustrate our proposal in the smart home domain. We first present an automated power modeling approach used to generate accurate real-time power estimation models for any type of devices in heterogeneous environments. Then, we present an energy-oriented extension for a reference ontology. The latter aims to represent useful concepts used for energy management purposes in connected environments. Furthermore, we develop algorithms that exploit knowledge from both the power estimator and the ontology, to generate the corresponding RL agent and environment. We also present different reward functions based on user preferences and power consumption. The proposed approach performs well given the low convergence period, the high level of user preferences satisfaction, and the significant decrease in energy consumption.

The main contribution of this thesis is to guarantee autonomic management of energy consumption. It also provides visibility on energy drains by estimating the power consumption of devices in an automated manner. It lays out a way to represent energy-related knowledge. Finally, it ensures that energy-efficient actions are executed in heterogeneous environments.

Résumé

Ces dernières années, la consommation électrique des systèmes cyber-physiques (CPS) a augmenté en raison du nombre croissant d'équipements connectés (par exemple, appareils intelligents, équipements IoT...), principalement dans le secteur résidentiel. Un grand nombre d'appareils intègrent des capteurs leur permettant de produire des données décrivant l'état d'un appareil ou le comportement d'une personne (par exemple, capteur de température, capteur de présence...). En outre, de nombreux appareils sont équipés d'actionneurs capables d'accomplir des tâches ayant un impact sur l'environnement (par exemple, le contrôle de la lumière, le système de chauffage, de ventilation, de climatisation...). Ces dispositifs ont le potentiel de collecter une grande quantité de données qui peuvent être utiles pour l'estimation et la gestion de l'énergie. Cependant, les approches actuelles de gestion de l'énergie sont principalement appliquées à des modèles limités d'appareils dans des domaines spécifiques et sont difficiles à mettre en œuvre dans d'autres scénarios. Ces approches actuelles présentent des limites en termes de leur niveau d'autonomie, leur flexibilité, leur généricité, les métriques contrôlées et l'hétérogénéité des dispositifs étudiés.

Afin de répondre à ces lacunes, nous présentons, dans cette thèse, une approche de gestion de l'énergie pour les environnements connectés basée sur la génération de modèles d'estimation de puissance, la représentation d'une description formelle des connaissances liées à l'énergie et à l'utilisation des techniques d'apprentissage par renforcement (RL) pour accomplir des actions énergétiques efficaces. Nous illustrons notre proposition dans le domaine de la maison intelligente. Nous présentons d'abord une approche de modélisation autonomique de la puissance utilisée pour générer des modèles précis d'estimation de la puissance en temps réel pour tout type de dispositif dans des environnements hétérogènes. Ensuite, nous présentons une extension orientée énergie pour une ontologie de référence. Elle vise à représenter les concepts utiles impliqués dans la gestion de l'énergie dans les environnements connectés. De plus, nous développons les algorithmes de notre approche RL qui exploitent les connaissances de l'estimateur de puissance et de l'ontologie pour générer l'agent RL et des environnements d'apprentissage correspondants. Nous présentons également différentes fonctions de récompense basées sur les préférences des utilisateurs et la consommation d'énergie. L'approche proposée donne de bons résultats compte tenu de la faible période de convergence, du niveau de satisfaction des préférences des utilisateurs et de la diminution significative de la consommation d'énergie.

La principale contribution de cette thèse est de garantir une gestion autonome de la consommation d'énergie. Elle fournit également une visibilité sur les pertes d'énergie en estimant la consommation d'énergie des dispositifs de manière automatisée. Elle propose également une manière de représenter les connaissances liées à l'énergie. Enfin, elle garantit que des actions efficaces sur le plan énergétique sont exécutées dans des environnements hétérogènes.

Contents

List of Figures

List of Tables

Acronyms

IoT Internet of Things

CPS Cyber-Physical Systems

ICT Information and Communications Technologies

OSI Open Systems Interconnection

M2M Machine-to-machine

M2H Machine-to-Human

RPi Raspberry Pi

SSN Semantic Sensor Network

SAREF Smart Applications REFerence

RO Research Objective

RL Reinforcement Learning

WSN Wireless Sensor Network

HVAC Heating, Ventilation, and Air Conditioning

ARCADIA Architecture Analysis and Design Integrated Approach

MBSE Model-Based System Engineering

EMS Energy Management Systems

Chapter 1

Introduction

Contents

During the second industrial revolution, electricity was introduced as a practical and easy way of transmitting power to consumers such as buildings and industries. The process of electrification that took place during this period was the main driver of the invention of electrical appliances used in our daily life. The widespread usage of these appliances led to a continuous increase in energy consumption. Since then, researchers have been interested in energy optimizations to maximize the efficiency of devices.

Two industrial revolutions followed: (1) the third revolution characterized by the introduction of automation, information technologies, and electronics revolution, and (2) the ongoing fourth revolution introducing Cyber-Physical Systems (CPS), Internet of Things (IoT), and networks revolution. Internet is a fundamental technology of these revolutions, providing a networking infrastructure for transmitting data and communicating between people and devices. Moreover, electronic devices' capabilities increased exponentially following Moore's law and allowed the ability to communicate, stay connected, and perform complex tasks. These two factors are the base for the design and development of CPS and IoT.

As seen in figure 1.1, a significant increase of connected devices is currently observed and will continue in the future, in particular, for IoT and smart home devices [1]. Cisco estimates that 500 billion devices will be connected to the internet by 2030 [2]. With this increase, the energy consumption of these devices is growing and may challenge the capacity of energy production. For instance, in 2019, over 25 000 TWh of final energy consumption was electrical, with the majority being consumed by the industrial and residential sectors [3]. The energy impact of CPS goes beyond the use phase, with important energy and resources spent in its entire life cycle (Exploitation of natural resources, production, transportation, use, and end-of-life phases) [4]. Our research work focuses on the energy consumption of CPS in the use phase. These devices consume energy directly from the electrical grid (e.g. television) or by storing energy in an internal battery (e.g. mobile phones or laptops). Optimizing a large number of small devices can have a big impact. A saving of 1.0 Mega-Watt per hour worldwide by saving 0.25 W per device for 4 million users and these numbers increase as we go to larger scales, as estimated by Hindle in [5]. Energy consumption is also not limited to the energy impact of the device itself, but the entire ecosystem impacted by the usage of CPS (such as networking, servers and data centers, database storage, etc.).

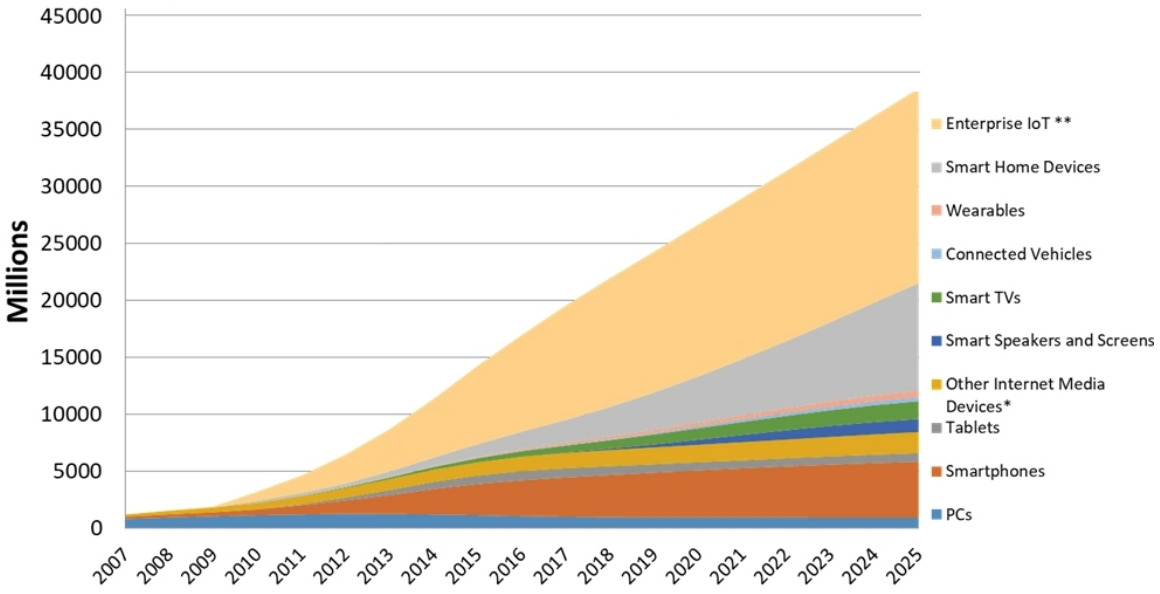


Figure 1.1: Global number of connected and IoT devices installed base forecast [1]

1.1 Problem Statement

The research community considers energy as one of the major concerns driven by the increasing number of connected devices and appliances. Many studies focused on using

more green energy sources, while others focused on making changes in the design phase, scheduling tasks, or encouraging users to accomplish energy-efficient actions through recommendations. However, we argue that previous literature suffers from certain weaknesses by being applied in specific scenarios or environments, on a limited number of devices, or a limited number of metrics. In the following, the different problems identified throughout this thesis and the motivation to target them are detailed.

Increasing power consumption in CPS

In recent years, the power consumption of CPS has been increasing due to the increasing number of connected devices. In addition, traditional appliances and devices became ”smart” or connected by adding circuits that allowed them to compute, communicate, and store data. Despite the efforts to reduce the energy consumption of some devices by introducing energy-conserving modes (e.g. televisions). The savings are minor and limited to the device itself without considering the full environment on a holistic approach.

Non-consideration of several contextual green metrics in CPS management

A large number of devices in a CPS produces a large amount of metrics that could potentially help to achieve energy efficiency of the entire system. This large amount of data is produced by Machine-to-machine (M2M) or Machine-to-Human (M2H) interactions. This data is collected from different layers of a CPS. Contextual data is not only collected from sensors but also by other layers and components of the system, namely, communication, processing, control, and application. For example, occupancy detection can be done in many different ways based on the available data, namely using a presence sensor, camera, mobile phone location, WiFi network utilization or more than one data source are combined to deduce the information. A variety of the data generated by different smart devices have the potential to be exploited from an energetic point of view.

Lack of flexible power management approaches for heterogeneous environments

Beside the limited exploitation of contextual green metrics, the computing community does not share a common approach allowing to consider the heterogeneity of devices in CPS environments in order to deal with power management. Different types of devices are deployed in these environments such as sensors, actuators, embedded systems, appliances, and others. Each of these devices is unique regarding the task it is intended to accomplish. They use different communication protocols, architectures, and components. In addition, they are usually provided by different vendors. Flexibility is also an essential characteristic because new devices are introduced, others are removed, or updated depending on the needs which can also be variable. In addition, different users could have different behaviors in the

same environment. Therefore, any considered solution for power management should be easily extensible to deal with the heterogeneity and flexibility present in CPS environments.

1.2 Research Objectives

The aim of our research work is to develop a power management approach that reduces the total energy consumption in CPS. We focus in this research on collecting, sharing, and analyzing data from the system in order to adapt accordingly. The main three research objectives (RO) addressed throughout this thesis are summarised as follows:

- **RO1. Identify metrics affecting energy consumption from different layers of a CPS environment**
  In a smart environment, a large variety of connected devices (e.g. Smart appliances, Information and Communications Technologies (ICT) devices, IoT devices, ventilation systems, lighting systems, networking devices ...) produce a large quantity of data. These are present in many layers of the environment. For instance, some data are contextual and others are related to the system. Contextual data are produced as a result of the interaction of actors with the environment. System data are usually collected from the system itself (processing, networking, sensing, actuating, applications running on devices ...). The definition of relationships between these metrics creates a new level of knowledge for an advanced contextual understanding of the environment.

- **RO2. Estimate real-time power consumption of any device in a connected environment accurately**
  The knowledge of devices' power consumption is an essential need to quantify energy leakage and saving. However, heterogeneity of connected environments leads to a challenging power consumption estimation due to the variety of metrics and energy concerns per device. Nowadays, the real-time power consumption of a device in a connected environment is unknown unless a hardware wattmeter has been connected to this device or a power model has been previously generated for it. However, changes in the environment by the introduction of new devices and changes in existing devices (such as hardware or software updates) have an impact on the power of each device. Hardware approaches are costly financially, scale poorly, and require time-consuming physical setups for each device. Software approaches are quickly out-of-date due to updates and scale poorly. For these reasons, it is necessary to estimate the real-time power consumption of devices using a method that will be able to

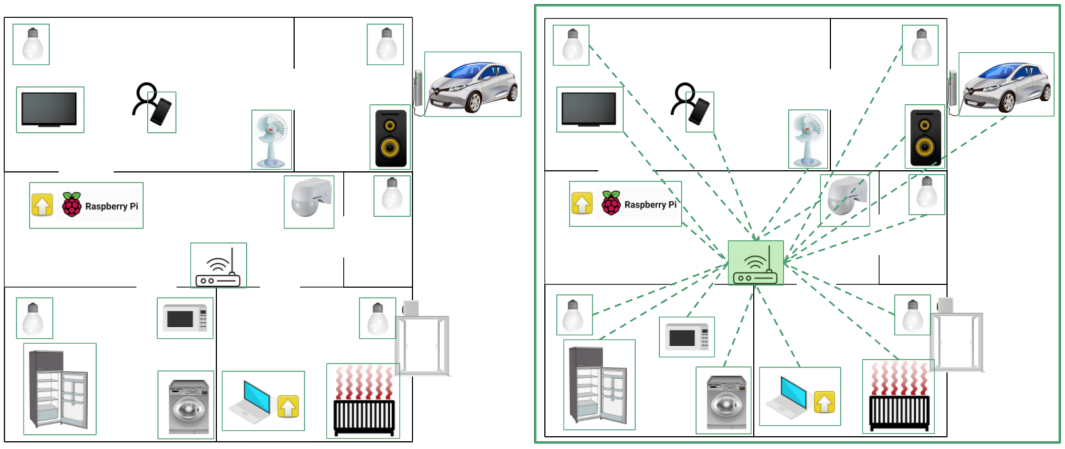
understand changes in the environment. A such technique will allow the calibration or generation of power estimation models when necessary.

- **RO3. Identify actions that can be applied depending on devices states and metrics to optimize the total energy consumption**
  A large number of devices integrate sensors allowing them to produce data describing the state of a device or the behavior of a person. In addition, a large number of these devices are equipped with actuators accomplishing tasks impacting the environment. These elements could allow the implementation of autonomous feedback control loops for power management allowing easy integration and understanding between devices in order to achieve higher energy savings. This approach for power management should guarantee the application of flexible adaptations and re-configurations during run-time to reduce energy consumption at holistic level. It could use the actions as outputs of a smart agent and data (including states and power measurements) to be the feedback from the environment.

1.3 Motivating Scenario

1.3.1 Context of a Smart Home

A smart home is a rich environment with different types of energy consuming connected devices and it represents a suitable motivating scenario for this research work, as seen in figure 1.2. The focus of this thesis is to pass from local energy optimization to holistic energy optimization in flexible and heterogeneous environments. In such context, several heterogeneous devices are connected using different protocols and sharing all kinds of data. They interact with each other to accomplish specific tasks, in addition to their interactions with the humans which adds an extra level of complexity to the system. This environment contains devices that produce various data types ranging from physically measures values (e.g. temperature) to software-related data (e.g. CPU utilization by a specific software). The smart home environments are scalable as the number of devices is limited, even if they can be quite heterogeneous having different types, interfaces, vendors, etc. However, these environments are dynamic as changes can occur to existing devices such as software updates, hardware updates, location changes, etc. Within this scenario, each device has its power consumption that needs to be estimated or measured in order to be able to estimate global energy consumption and implement the appropriate power management strategies.



(a) (b)

Figure 1.2: Smart Home Environment (a) local energy optimization, (b) holistic energy optimization

1.3.2 Smart Home Scenario

The number of possible scenarios and combinations of devices running together in the smart home environment is unlimited. Therefore, we propose a concrete scenario around equipment in smart homes : we assume an environments with a variety of connected devices including a smart TV, an media box, LED light bulbs, a home occupant wearing a smartwatch (such a device can provide significant information mainly about the user activity), and a light sensor mounted on a Raspberry Pi (RPi). Energy-saving modes are present in some of these devices. For example, the smartwatch screen turns off unless the user is looking at his watch or turns his wrist, and the TV turns off or goes to sleep after a period of inactivity. In such scenario, devices do not exchange data to seek higher energy savings and the power consumption of these devices is unknown unless connected to a hardware measuring equipment.

However, we know that power is affected by many metrics (e.g. TV power is affected by its brightness, LED light bulb power is affected by brightness and color, and a media box power is affected by decoding high quality streaming videos). In addition, when a device exchanges data collected from its sensors to the entire system, power can be estimated for each device and more interesting adaptations are accomplished on a fine-grained level leading to holistic energy optimization. Power management is accomplished while ensuring user comfort: reducing LED and TV brightness, reducing the streaming quality, checking user's behavior and acting accordingly, and estimating and share the power consumption of a device.

1.4 State of the Art of Related Work

A large number of existing studies have examined energy management in ICT and CPS. These studies have been thoroughly reviewed and the detailed state of the art review is presented in chapter 2. In order to better analyse these proposals, power management solutions in CPS and ICT have been categorized as follows:

Design and simulations: These techniques are based on reducing energy consumption during the design phases of a device or system. It can take the form of using newer technologies, making changes in the architecture or design of the system [6, 7, 8, 9, 10, 11], or using simulators [12, 13, 14] to study the behavior of a component before putting it on the market. It aims to raise a device's efficiency while reducing its power consumption.

Scheduling: That consists of changing the time slot when a specific task is executed. The change in the time of execution of a task is usually done to benefit from energy production when it is greener [15], better energy efficient execution time slot [16], or during the absence of users [17, 18]. In addition, scheduling could have a target of detecting unused devices to change their operating mode [19, 20, 21]. This approach is usually implemented in data centers [22, 23, 24], Wireless Sensor Network (WSN) [25, 26], smart homes [27, 28], and IoT [29].

User choice and recommendation: As seen in [27], user behavior has a direct impact on energy consumption. These approaches are based on encouraging users and recommending them to take actions that reduce energy consumption. It also includes raising awareness among users about their consumption by showing them visual information and notifying them. Most of viewed solutions focus at individual level [30], enterprise level [31], and community level [32]. In addition, others used games to show users their energy consumption [33, 34, 35].

Adaptation and reconfiguration: These approaches are based on the ability of a system to change its initial configuration by tuning some parameters allowing it to adapt to changes that take place during run-time. It's usually based on a finite number of cases that were predefined during design time [36] or multiple predefined operating modes [37, 38]. Other solutions propose autonomous systems that can use machine learning techniques and build knowledge bases [39].

Power supply/demand: Power demand changes regularly with time, and so does the power supply, especially with renewable energy such as photovoltaic cells and wind turbines. Providers need to match their power production with the demand to be as efficient as possible [40]. Many methods are used such as predicting the power needed

at a certain time based on historical records [41], balancing available power supply and demand [42, 43] or storing energy when it is available that is the cape for electric vehicles [44, 45].

Migration: These techniques are usually used alongside virtualization in large distributed data centers [46, 47, 48, 49], distributed services in a WSN [50], or even corporate environments [22]. Virtual environments can be created to migrate easily an application from one server to another that is less loaded. In short, it is based on changing the place of execution of a task.

In order to evaluate the previous solutions, several comparison criteria have been defined including the level of autonomy, system flexibility, general-purpose of an approach, and heterogeneity. These criteria can be defined as follows:

Level of autonomy: The five maturity levels of autonomy defined in the Autonomic Computing framework proposed by IBM [51] will be used to classify related work. *Level 1:* Manual level: users perform all the management functions. *Level 2:* Monitor level: data from the managed resources is collected helping users to minimize the time needed to collect and understand information. *Level 3:* Analysis level: technologies are used to provide a correlation between metrics. It recognizes patterns, predicts the best configuration, and offers advice about potential actions to the user. *Level 4:* Closed loop level: environment can automatically take actions based on the available information and the knowledge about what is happening in the environment. *Level 5:* Closed loop with objectives level: high-level user-oriented business policies control the infrastructure operation in an automated manner.

System flexibility: System flexibility is the ability of the system to handle a dynamic environments. This includes the addition, update, and removal of devices or software/hardware components changes, as well as in the network topology.

General-purpose: Determines the genericity or specificity of solutions regarding potential application domains. It provides a glance if the approach can be easily applied to any domain and if it is still efficient under different circumstances and different devices.

Monitored layers and metrics: The structural levels of the CPS where the solutions are developed and where the metrics are collected. It provides an understanding of the layers on which the optimization is achieved and if it is limited to these layers.

Heterogeneity: It is defined by the presence of a variety of hardware and software frameworks that use different protocols, components, and architectures and that may be provided by different vendors.

Table 1.1 compares the state-of-the-art categories using the previously mentioned criteria. As summarized in this table, none of the studied solutions could be classified as belonging to the 5^{th} level of autonomy and the ones categorized as levels 3 and 4 do not satisfy all the properties of these levels. Moreover, the flexibility is highly limited to the devices and the environment in which they were developed, therefore, they are mostly domain specific and seem complex to implement in another domain. Monitored layers seem also quite limited and plenty of metrics are not considered. Beside the absence of flexibility in the aforementioned works, implementations were conducted on specific types of devices. Therefore, a lack of heterogeneity could be observed. These drawbacks motivate the needs to continue the research in order to find a better approach that can consider the previously mentioned problems while respecting the defined criteria.

Table 1.1: State of the art categories comparison

Approach	*Level of autonomy*	*System flexibility*	*General-purpose*	*Monitored layers*	*Environment heterogeneity*
Modeling	*1*	*None*	*Low*	*Low*	*Low*
Scheduling	*3-4*	*Medium*	*Low*	*Low*	*Low*
User recommendation	*1*	*Low*	*Low*	*Low*	*Medium*
Adaptation	*3-4*	*Medium*	*Medium*	*Low*	*Medium*
Migration	*3-4*	*Medium*	*Low*	*Low*	*Low*
Supply/demand	*2-3*	*Low*	*Low*	*None*	*Medium*

1.5 Proposal

Our proposal intends to facilitate power management in CPS by providing (1) an architecture based on the MAPE-K (Monitor, Analyse, Plan, Execute, Knowledge base) autonomic computing approach [52], as seen in figure 1.3, (2) an automated approach to empirically generate power estimation models for a large set of devices, and (3) an automated knowledge-based reinforcement learning agent generator to choose the optimal actions to be executed. Any changes or updates that occur in the environment are considered leading to the calibration of estimation models and reinforcement learning agents. Moreover, our proposal considers all the previously defined criteria. The adoption of a MAPE-K approach guarantees non-functional capabilities including self-management, self-configuration, self-optimizing, self-healing, and self-protection. In addition, it guarantees functional capabilities covering the data collection, monitoring, analyzing, planning, executing, and actuating. Our architecture approach is divided into 5 main components defined as follows:

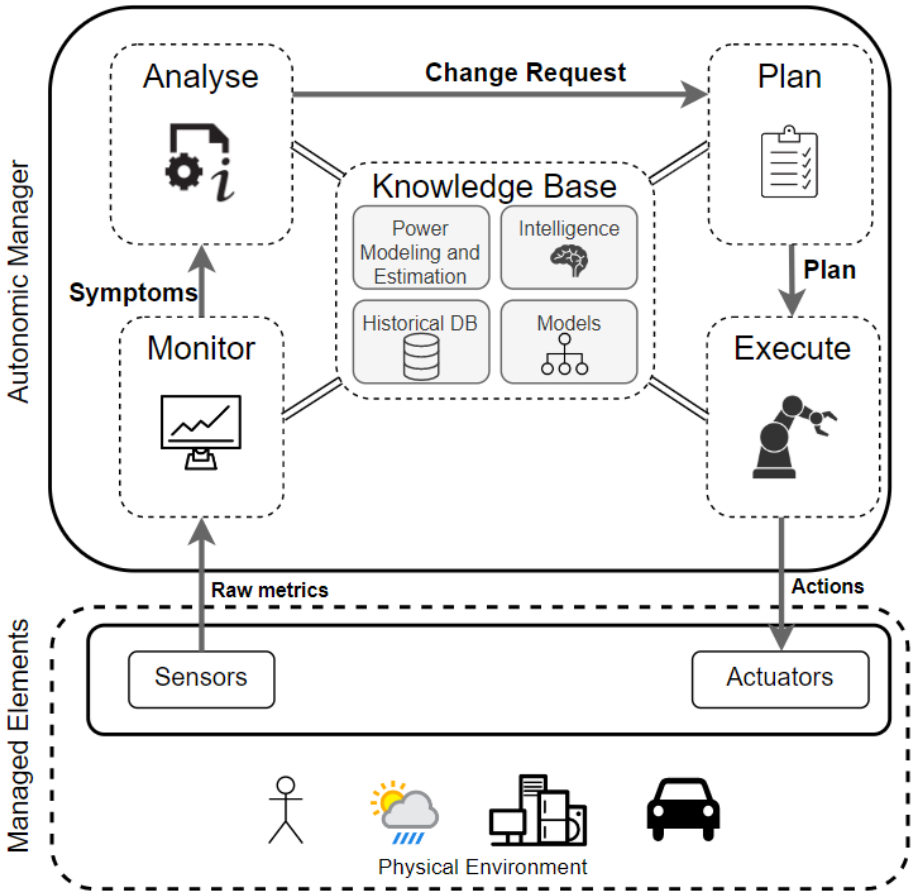


Figure 1.3: MAPE-K architecture

Monitor: Collects a collection of useful data from each of the previously enumerated layers in the CPS, to identify the energy-related leakage. Data can be captured using APIs, software estimation tools, physical sensors, or other means. In the monitoring component, data is cleaned, filtered, and aggregated. Collected data could be related to the context, communication, processing, software, or states of the devices. It selects the significant features, correlates, and organizes them as symptoms that are sent to the analysis entity.

Analyze: It is the entity responsible for exploiting and processing the collected data received from the monitor as symptoms. Furthermore, finds what is making the system consume more and what can be done to reduce this consumption, based on high-level management strategies. It classifies symptoms and processes them to deduce a diagnosis.

Plan: Proposes a set of actions that needs to be achieved to accomplish the high-level goal related to energy consumption. It is responsible for planning what should be changed to reduce the energy of the system and sending this feedback signal back to the CPS to execute it. Short-term and long-term plans are elaborated. They are highly based on the

change request sent by the analyzer and the knowledge repository.

Execute: Will receive the set of actions and sends them to the effectors. Effectors will accomplish specific tasks in the environment. Actions accomplished on a device can be either behavioral (e.g. changing the frequency of sharing data) or structural (e.g. changing the operating mode of a device to sleep).

Knowledge-Base: Exchanges with all the previously cited entities with awareness abilities. It is where most of the contributions of this research work were developed. It includes models to estimate the energy consumption of the devices, metrics ontology model (which provides cognitive capabilities that allow finding the relationship between the metrics), power consumption, and actions. It also included machine learning techniques that find the most suitable adaptation to minimize energy consumption.

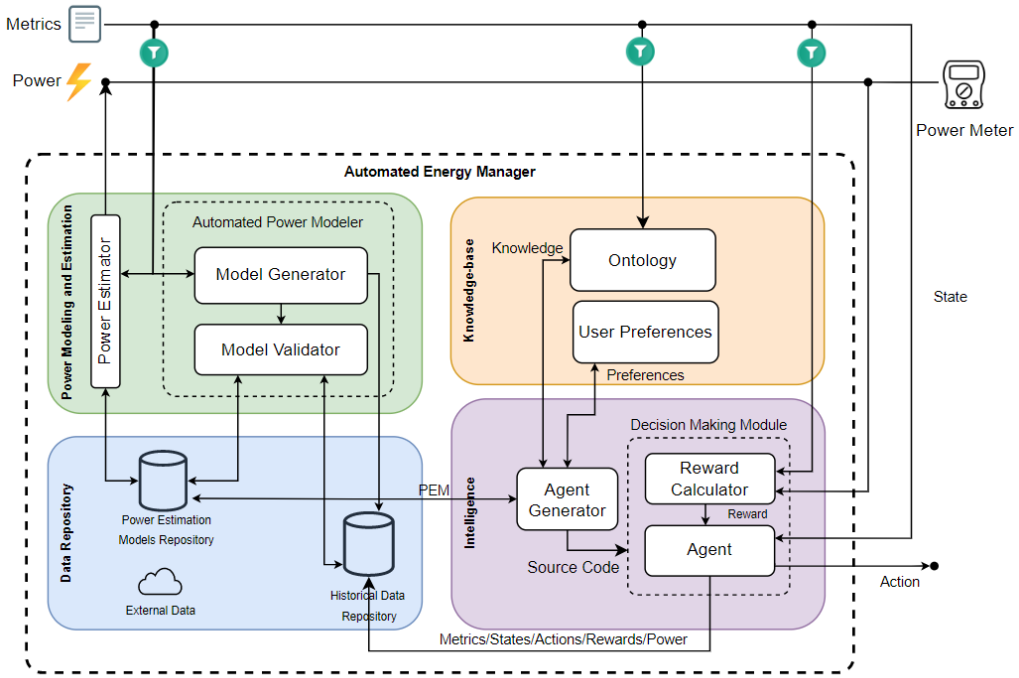


Figure 1.4: Knowledge-base architecture

In particular, our research work focuses on the shared knowledge base of the autonomic computing architecture. Figure 1.4 shows its architecture divided into 4 main components:

Power Modeling and Estimation: Automatically estimates the power consumption of a device based on the collected metrics without the constant need for measurement hardware.

Knowledge-base: It is used to understand the relationship between metrics from different devices, power consumption, and actions that can be applied. This model correlates the metrics and the potential actions to be carried out by any new device introduced to the environment. The use of an ontology guarantees the inclusion of heterogeneous devices and the general-purpose of the approach.

Intelligence: It has the central role of identifying the best energy-efficient action that should be taken and its impact on the entire system while respecting user satisfaction. It is composed of (1) an agent generator that creates a Reinforcement Learning (RL) agent based on the knowledge received from the knowledge-base component, and (2) a decision-making model that receives the metrics and the power consumption to choose the best action that can be taken to save energy. Together, they guarantee high flexibility and automation levels.

Data Repository: A collection of databases that includes previously collected metrics, applied actions, power consumption, power estimation models, and potential external data.

The detailed solution proposal and architecture are presented in chapter 3

1.6 Contributions

The following paragraphs summarize the three main contributions of this thesis:

1.6.1 Energy Management Approach Architecture

The first contribution of this thesis is the architectural design of our energy management approach using a Model-Based Systems Engineering (MBSE) methodology. The proposed architecture for energy management ensures that our approach is holistic and can manage a variety of devices in an evolving environment. This contribution allows the definitions of functional and non-functional requirements in a traceable manner for proper validation at the end of the development process. It documents all decisions and choices made during the development, in particular, the logic behind using ontology and reinforcement learning techniques. We start with an operational analysis defining the needs of different actors and the system. The system architecture level aims to define the functionalities required the satisfy the operational needs identified in the previous phase. The third step focuses on a logical architecture building a detailed component-level architecture of the system. Finally, the physical architecture aims to pass from an abstract logical architecture to a representation of how the system will be developed and built on the implementation level.

1.6.2 Automated Power Estimation of Heterogeneous Devices

The second contribution of this thesis is an automated architecture and approach to empirically generate power estimation models for a large set of devices. This contribution allows conducting automated benchmarks to collect power data and metrics, generating or updating accurate power models, and allowing software tools to query and retrieve the most accurate and up-to-date power model of a specific device configuration. It is divided into two types of components: clients and servers. The client's role is to collect software, hardware, and power metrics of run-time and real-world workloads. The collected data will then be shared with the server. The latter's role is to generate (train and validate) accurate and always up-to-date power estimation models using machine learning algorithms. The proposed approach to generate power estimation models is implemented, tested, and discussed for multiple device configurations.

1.6.3 Automated Energy Management Framework

The third contribution of this thesis is an automated knowledge-based reinforcement learning energy management framework. It is based on the extension of the Smart Applications REFerence (SAREF) ontology to allow the integration of a variety of devices in the reinforcement learning loop. It defines metrics, states, actions, power consumption, and other information needed to apply reinforcement learning. Relations between devices is also defined to increase the context understanding ability of the system.

In addition, this knowledge is used by the automated reinforcement learning agent generator to create a compatible decision-making agent. It can adapt to any environment and uses the ontology to provide the required knowledge to understand what are the metrics, power consumption models, and actions associated with each device, in addition to the understanding of the relations between different devices. The generated agent then takes as an input the metrics of any environment and chooses the best actions to reduce the power consumption at holistic level based on energy optimization policies and rewards.

1.7 Structure of the Thesis

The remainder of this thesis manuscript is organized as follows:

Chapter 2: State of the Art: This chapter reviews current research approaches and directions in energy-aware CPS. A systematic review to analyze and compare state-of-the-art approaches, based on their architectural design and energy-related factors, is proposed.

Chapter 3: Energy Management Approach Architecture: This chapter presents a requirement analysis of the proposed problem. The design and architecture of the proposed approach, aimed to reduce energy in CPS are also presented.

Chapter 4: Automated Power Estimation of Heterogeneous Devices: This chapter presents the second contribution accomplished during this thesis. It presents a study of power measurement and estimation models in computing devices. It also proposes an automated architecture and approach to empirically generate up-to-date power models for a large set of devices.

Chapter 5: Automated Energy Management Framework: This chapter presents the third contribution accomplished during this thesis. It presents the models used to define the relationship between devices by defining their metrics, states, actions, and feedback forms. In addition, a proposal for an automated reinforcement learning agent generation is presented, having the goal of increasing energy efficiency in an environment with a large number of heterogeneous devices.

Chapter 6: Validation and Discussions: This chapter validates the proposed approach with the development of two real-world scenarios. The experimental setup is described, then, the results and the validity of the approach are discussed.

Chapter 7: Conclusion and Perspectives: This chapter summarizes and concludes the thesis work and contributions previously presented. Finally, the vision for upcoming research directions is shared.

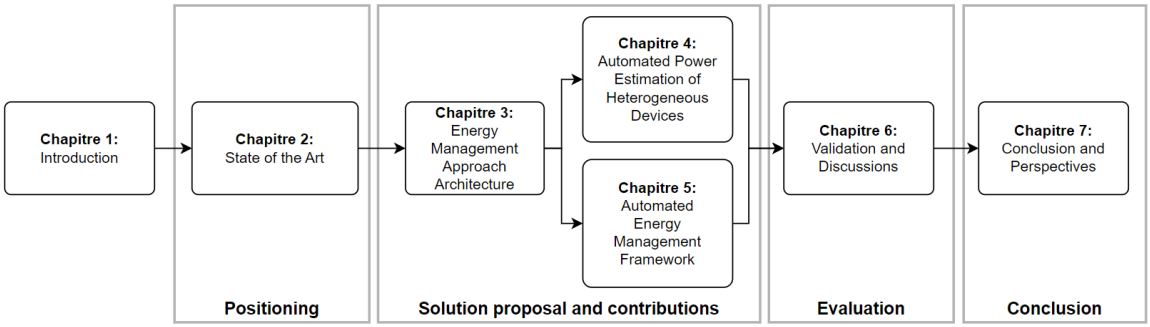


Figure 1.5: Structure of the thesis

Chapter 2

State of the Art

Contents

2.1 Introduction

Recently, research interest in IoT and CPS approaches aiming to satisfy non-functional properties is increasing, in particular around energy consumption and sustainability of devices usages. More sustainable solutions are proposed lately (such as recycled appliances, using renewable energy sources, or promoting energy efficiency standards [53]). Even though sustainable and renewable energy sources are more and more used, such as wind, solar, and ocean energies, the resulting total consumption is continuously increasing. This is causing a significant ecological impact mainly due to its carbon emissions. Meanwhile, some connected objects could have the potential to monitor and control the appliances through a variety of manual and automated approaches, allowing the control and optimization of their energy consumption. As CPS integrate computing devices (such as sensors and actuators, IoT devices, mobile, and connected devices and appliances), and interact with the physical environment (such as homes and buildings, human end-users, animals, and natural phenomenons), the assessment of their energy consumption and its optimization is a challenging task. Our understanding of energy consumption in CPS is limited to electrical

devices, and their efficiency is mostly studied independently of their usage and interaction with the environment. However, these devices could be considered smart as they could take decisions based on the knowledge they have of the CPS itself and its environment. This knowledge can be pre-configured by users, administrators, or manufacturers, and can be acquired and improved using machine learning techniques such as reinforcement learning. In such environments, the energy consumption of devices is, therefore, more complex to manage because of the interactions between different components. This complexity is due to additional layers of information that need to be addressed and taken into consideration when applying energy optimization techniques. A lot of data can be collected from these extra layers of the system. Thus, a higher understanding of the environment and potential for energy optimization per layer is required. Different layers and architectures of CPS are reviewed in the next section.

2.2 Cyber-Physical Systems

In this next section, we provide a definition of the Cyber-Physical Systems (CPS). Then, we identify different architectures and layers that compose a CPS. After that, we identify the impact of energy in ICT and CPS. Finally, we summarize some limitations of existing CPS architectures.

2.2.1 Cyber-Physical Systems Definition

A CPS is defined as a smart system composed of interconnected devices defined by computing and physical processes [54]. These processes run over physical components, and over systems with limited resources in variable temporal and spatial conditions [55]. Computational systems main objective is to control and interact with physical processes via touch-points (sensors and actuators). CPS are characterized by the significant interaction between networks of physical and computational components. In addition, human interaction and behavior in such environments have an important impact on the system. It is seen as an intelligent, real-time, adaptive, predictive, and distributed feedback system. It is characterized by its interoperability and scalability because it is implemented in heterogeneous environments as seen in figure 2.1. A CPS can be considered as a system of systems. These systems are usually self-managing and self-optimizing systems [56]. This kind of systems are cross-domain oriented applications, such as in the fields of healthcare, smart homes, autonomous automobiles, industry, and many more [57, 58, 59, 60]. An example of this kind of systems could be autonomous vehicles, equipped with sensors and actuators. In order to sense and understand its environment, make its own decisions, and execute these decisions

using the actuators. Healthcare robots, medical monitoring devices, industrial robots with the monitoring IoT devices in factories, and smart home management systems are all good examples of CPS. A CPS does not only have great economic and social impacts, but it can have an important energy footprint, and could be optimized in order to reduce the global consumption.

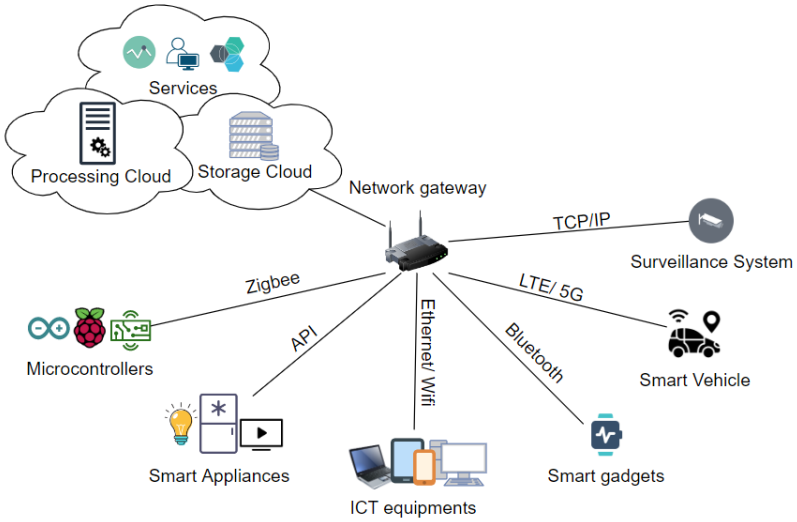


Figure 2.1: Example of devices found in a CPS

2.2.2 Cyber-Physical System Architecture Review

ICT are the different components and infrastructure that make modern computing possible, it includes hardware devices, communication technologies, applications and systems that shape the digital world [61]. ICT have been modeled and specified with standards to describe their functions such as the Open Systems Interconnection (OSI) model that provides an abstract framework to describe the communication functions of a computer system [62]. However, these standards do not cover all the specificities of heterogeneous physical and logical components of CPS. We believe that there is a need to provide a standardized referential model for CPS, in order to specify their functional and non functional properties and in particular integrating the energy dimension. In this section, we present previous research work offering architecture proposal for CPS from a structural point of view.

In [63], the authors presented a survey of system integration in the context of Industry 4.0 using the 5 C integration levels (Connection, Communication, Coordination, Cooperation, and Collaboration). They also identified the most common challenges around Industry 4.0 as follows: the complexity of planning, standardization, security, privacy, heterogene-

ity, and integration. Creating standardized CPS architecture may solve these challenges by allowing all entities to apply the 5 C easily.

The authors in [64] identified challenges facing the CPS. They mentioned the challenges of interoperability, system distribution, real-time concern, component modeling, testing, optimization algorithms, safety concern, and heterogeneous data. One of which is the need to develop standardized architectures while taking into consideration the high dependency between the software and hardware components. Any proposed architecture should respect the three abstract layers of a CPS: Computation, communication, and control (3C). The abstraction of these layers grants the integration and interoperability of heterogeneous systems together. The authors in [65, 66, 67] also considered CPS as an integration of computation, communication, and control (3C) technologies. The concept of 3C is considered by some researchers as the backbone architecture of CPS. In [65] a CPS is defined by three main components: a physical, network, and a distributed cyber systems, in addition, it operates on perception, transmission, and application layers. [68] proposed a service-based CPS to deal with the challenge of difficult computation capabilities on connected devices with limited resources. Three tiers architecture was essential to build this system. An environmental tier collects the information gathered from the physical world to the control tier that manages the physical components and services. It identifies the appropriate services, and ensures the dynamic composition of new real-time services based on the need. Finally, the service tier provides reusable functionality as cloud computing and micro-services. In [69], the authors presented an environmental monitoring CPS based on a three-layer architecture. The bottom layer is composed of several connected sensor nodes distributed over a wide area. The middle layer of the CPS receives data from the lower layer and stores it, analyzes and takes decisions. The top layer provides web-services for clients in the form of Software-as-a-Service to interact with the system. A framework for analyzing cyberattacks against CPS is presented in [70]. Vulnerabilities such as taking complete control, damaging, stopping the functionality, disrupting, and degrading the system are divided into three logical layers. Physical layer (Sensors, actuators, and system dynamics), control layer (Signals from physical layer and control algorithms) and cyber layer (traditional ICT components like bus, processor, memory ...)

The authors in [57] reviewed previous models and proposed a four layers architecture based on Service Oriented Architecture (SOA) that respects real-time control, security, and integrability characteristics. Perceive tier includes environment awareness and pre-processing, data tier stores and processes the data to make it homogeneous. Service tier controls the whole system by scheduling tasks, making decisions, and provide API for consumption. Finally, execution tier interacts with the environment directly using physical actuators or with the system itself. Intelligent transportation, agriculture, and medical ap-

plications were introduced as potential domains for this architecture. In [18], a distributed public street light adaptation system was proposed to save energy and took into consideration new performance indicators as power reduction, CO_2 emission reduction, and service usefulness. This system was based on the following layers: Capillary networks layer includes heterogeneous wireless sensors and actuators. Network backbone layer assures homogenization, reliability and data transformation. Enabling technologies layer provides data mining and infrastructure management. Services and applications layer exploits the information gathered with an abstraction level to reuse these services. The authors in [71] defined a general architecture of a medical CPS and discussed vulnerabilities regarding secure storage, networking, and computation. Data acquisition layer composed of battery-powered wireless wearable sensors for the collection of patients data. Data aggregation layer acts as a gateway that collects the data from the previous layer, aggregates them, and sends them to remote locations for further processing. Cloud processing and storage layer achieves storage, processing, and analytics for decision making and predictions. Action layer executes the decisions actively or passively (With or without physical interaction). The authors in [72] proposed an agricultural management system architecture based on their findings of the four major layers. The physical layer interacts with the environment and collects data. The networking layer ensure the transition of data throw wireless communication. The decision layer stores, and analyses the data to provide decision-making graphical tools in the application layer. This provides the knowledge to the experts throw dedicated applications and web services. This CPS is not autonomous, its functionality is highly dependent on the farmers' interpretations and interventions.

Table 2.1: Comparative table of distinct CPS layers

S:Sensing, C:Communication, A:Actuating, P:Processing, Se:Services, Co:Control, SA:Sensing and Actuating

Article	Domain	Layers	S	C	A	P	Se	Co	SA
[64]	General	3		x		x		x	
[68]	General	3					x	x	x
[69]	General	3	x			x	x		
[66]	General	3		x		x		x	
[65]	General	3					x	x	x
[70]	Security	3				x		x	x
[57]	General	4	x		x	x	x		
[18]	Smart city	4		x		x	x		x
[71]	Security	4	x	x	x	x			
[72]	Agriculture	4	x	x		x	x		

We identify the distinct layers of a CPS for each of the literature architectures. The main identified layers in CPS are: Sensing, Communication, Actuating, Processing, Ser-

vices, and Control. In some architectures sensing and actuating are combined into the same layer. Table 2.1 summarizes our layer categorization per approach, the domain in which the architecture was proposed, and the number of considered layers.

This section showed that many researchers created architectural models of CPS by dividing them into layers especially in the domain of security and attack protection [70, 71, 73]. Some adopt the approach of physical architecture separation of layers as in [71] where each layer is identified by one or many components that are physically independent. Most approaches adopt a higher abstraction with the separation of functionalities to identify the layers remarkably found in [18, 64] where the function accomplished on each layer makes it identifiable, thus, two tasks operated by the same component can lead to the identification of two layers.

As seen in figure 2.2, the most common layers of CPS are communication, processing, services, control, sensing, and actuating in most architectures. These layers are not fixed and can be changed or redefined as more granular layers, such as separating the software from the hardware it is running on in the processing layer. For example, the sensing layer is referred sometimes as perception layer.

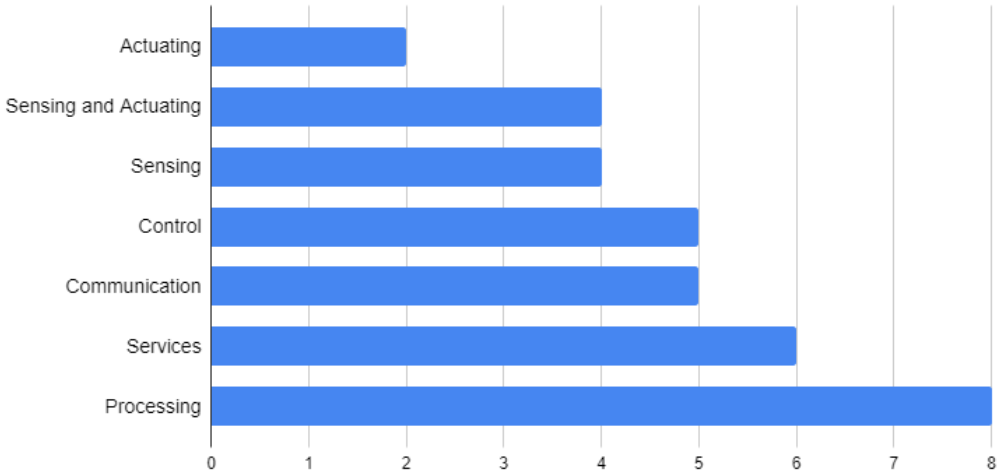


Figure 2.2: Identified CPS layers

The analysis of these works allows us to conclude that from a structural point of view, there is no unique architecture to identify a CPS. Moreover, previous research papers do not consider energy as a relevant property to be specified and taken into account. Similar to security concerns, we can identify energy properties by layer. We believe that this kind of non functional property needs to be considered in order to be able to optimize the structure and behavior of CPS in an effort to cope with energy consumption reduction goals.

2.2.3 Energy in ICT and CPS

Although developers do not usually think about energy while designing a system, energy management has always been a concern in ICT. Power saving solutions are found on individual devices such as smartphones and laptops notably since they run over batteries. In these devices, energy affect directly user experience but when users are not directly impacted, energy is seen as a non-relevant concern. Most of these solutions are based on making devices more efficient and reducing idle power consumption by using various power plans that make actions as turning off their screen, CPU frequency modification, and others. They also include different operating modes such as active, sleep, hibernate, and off. Energy management is also found in data centers due to the rapid growth of cloud computing and micro-services especially with the increasing need for new infrastructure, higher storage, and networking capabilities. For instance, in some countries such as Denmark, data center electricity consumption is expected to increase to 15% of total electricity consumption by 2030 [74].

The increasing number of ICT and CPS devices used by a person in everyday life leads to concerns about saving the energy on each system that can lead to better performance and reduce energy on a large scale. At the same time, this could offer an opportunity to gather information and use it for automating energy management. In ICT, energy optimization is limited to a certain type of devices, such as computers and smart phones, that usually have a power manager acting locally. Their adaptations are logical and do not directly impact the physical world and are not directly impacted by the physical world. With the integration of the CPS in our daily life, physical activities can be detected and predicted uncovering potentials of energy optimization in CPS. Energy management in CPS could be even more efficient than traditional ICT due to the understanding of the environment where they are implemented. Processing collected data allows them to reduce the consumption of devices that used unnecessary energy. It is important to note that the energy needed for accomplishing a task in the physical world needs usually significant amounts of energy. ICT and CPS can also work accordingly to ensure the best energy optimization. For example, mining the data produced by a user's laptop or phone leads to the knowledge of his geolocation, hence, turning off his home lights when he is away.

In the previous section, we defined CPS and reviewed the state of art of different architectures of CPS. We found that many papers defined the CPS layers, but until today there is no general unified architecture or standard framework to define it. Therefore, we consider processing, services, communication, control, sensing, and actuating as the main layers of CPS.

The remainder of this chapter presents the research protocol and summarizes different energy optimization approaches in CPS. Some of them optimize the energy consumed by

the CPS itself, while others use the CPS to reduce the energy of a wider entity such as buildings. We include both design-time and run-time approaches. We also include some potential software solutions that are already applied in ICT but not yet widely used in CPS, as we believe they have an impact on its overall energy consumption.

2.3 Energy-aware Cyber-Physical System Approaches

2.3.1 Survey Protocol

The research method adopted in this survey follows the process defined in [75]. First, we identify the research questions. Then, we perform a manual and an automated search. Next, we remove the duplicates and apply inclusion and exclusion criteria. After that, we make our analysis and comparison based on the CPS layers and defined comparison criteria. Finally, we discuss the found results. The research method process is presented in figure 2.3.

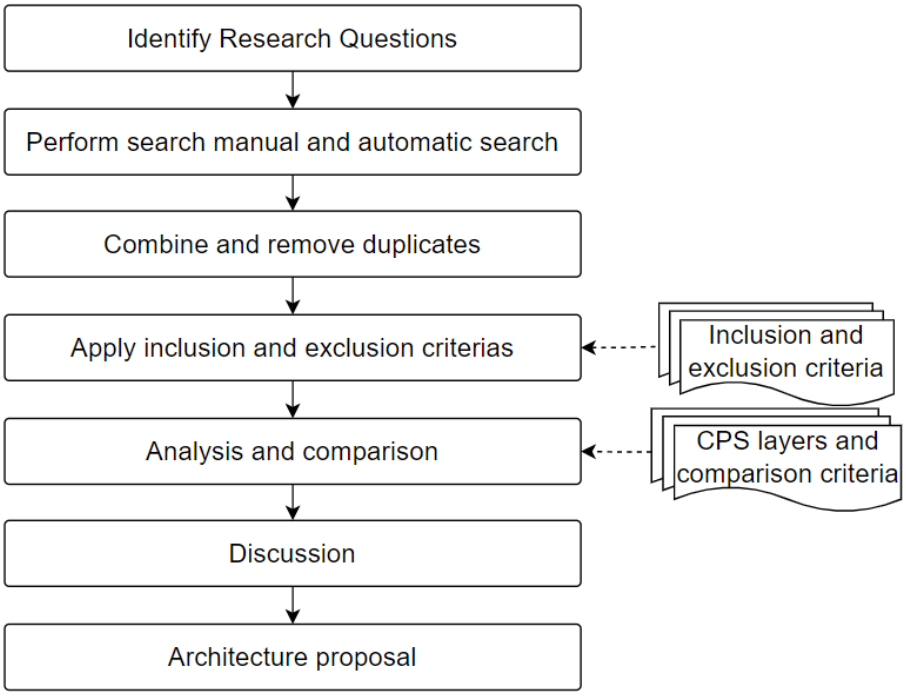


Figure 2.3: Survey protocol process

Research Questions: The main objective of our review is to identify the existing research approaches dealing with energy concerns in CPS. In order to achieve this goal, we first formulated the following research questions:

- *RQ1: What type of approaches is used to deal with energy in CPS?*
- *RQ2: What are the studied metrics that have an impact on energy consumption?*
- *RQ3: Studied metrics belong to which CPS layer?*

Search Strategy: The search strategy is based on an automated search using a search string and a manual one in order to include relevant research papers in our survey.

- Automated search: We used SCOPUS, one of the world's largest and most comprehensive scientific databases, to conduct automated searches using the Harzing Publish or Perish Tool[1]. To accomplish that we used the following search string:

  ```
  (cps OR cyber physical system OR cyber physical
  systems) AND (green OR energy OR ((energy OR power)
  AND (efficient OR consumption OR saving OR management)))
  ```

  We have limited the number of studied papers to the first 100 found articles while doing the search query[2].

- Manual search: We manually choose research papers published in relevant journals, conference proceedings, and cited in prior thematic related surveys.

  The output of this stage was **128** articles.

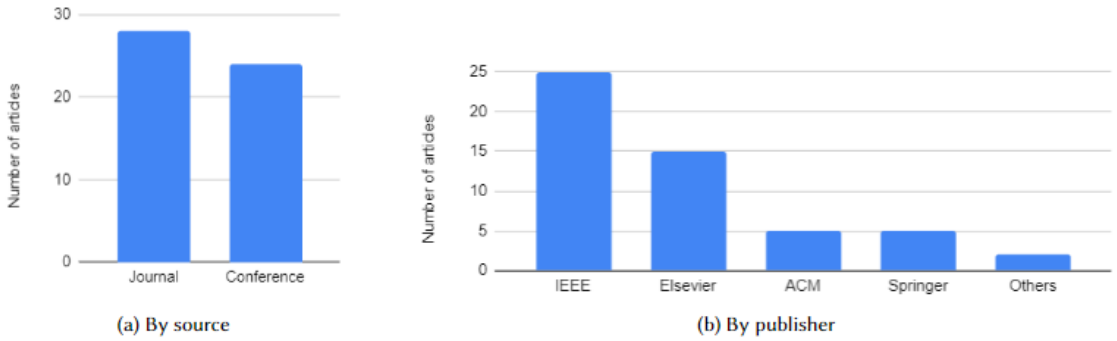


Figure 2.4: Article sources and the number of articles found

Study Selection: A study selection was conducted to ensure that the findings are reliable and informative regarding our research questions. The following describes the selection process:

- Combination and duplicates removal

[1]Harzing, A.W. (2007) Publish or Perish, available from `https://harzing.com/resources/publish-or-perish`

[2]The search was executed on October 20, 2021.

- Inclusion Criteria: i) studies proposing an approach for reducing energy consumption in CPS, ii) studies published after 2010, iii) studies cited at least one time (except studies published between 2018 and 2021)

- Exclusion Criteria: i) studies that propose the use of CPS without providing a energy reduction approach in the CPS itself (especially in power plants), ii) studies that were not available in full-text or we could not access, iii) studies written in language different than English.

Inclusion and exclusion criteria can be formulated in as follows:

```
[(YEAR >= 2010 AND CITATIONS >= 1) OR (YEAR >= 2018
AND CITATIONS = 0)] AND DUPLICATION = FALSE AND LANGUAGE =
"ENGLISH" AND AVAILABLE_IN_FULL_TEXT = TRUE AND SUITABLE_TITLE
= TRUE AND SUITABLE_ABSTRACT = TRUE
```

The last stage resulted in **52** primary studies, which we have used for our analysis. The average number of citations per research paper is equal to 48,5. This list of studies can be found online[3].

Figure 2.5 shows that in recent years there have been an increasing interest in energy optimization in CPS. Figure 2.4 shows the trend and statistics regarding the sources of the reviewed publications on energy CPS. Figure 2.6 shows the papers application domains frequencies. Chosen research papers belong to a variety of domains.

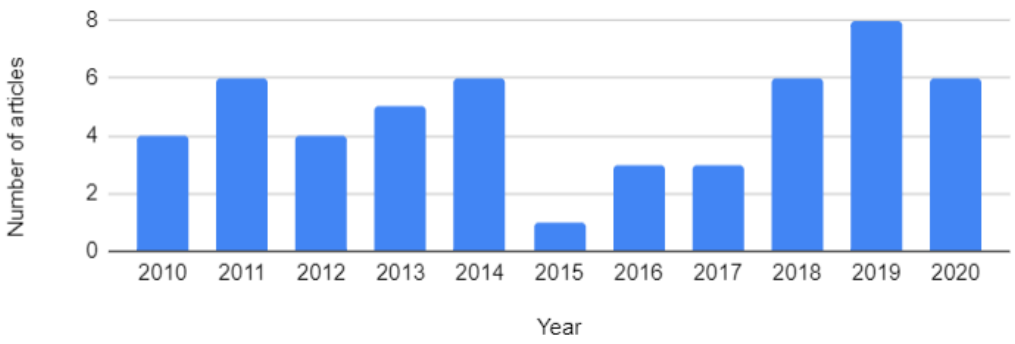


Figure 2.5: The number of articles found by year

[3]The list of the studies: https://t.ly/rbKe

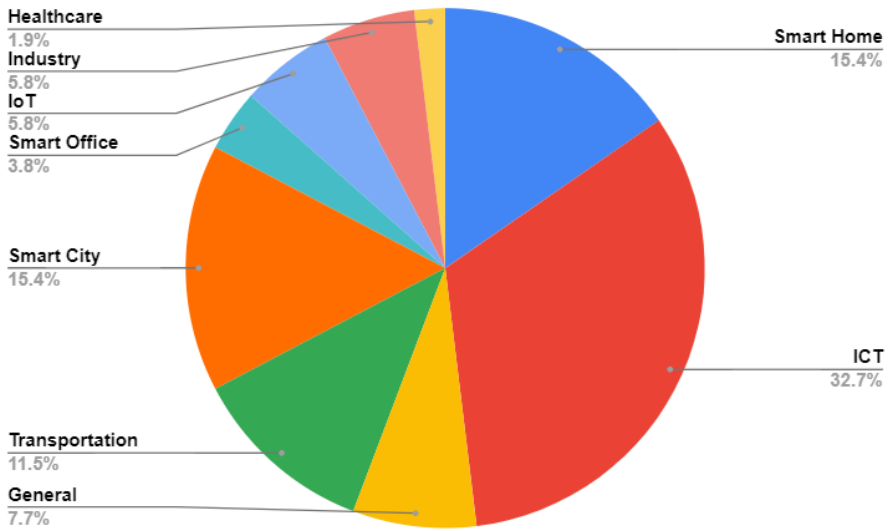


Figure 2.6: Studied papers application domains

2.3.2 Energy-aware Cyber-Physical System Review

In this section, we present a literature review of existing research approaches aiming to optimize and reduce the energy consumption in CPS. During our examination, we could identify the most common concepts of energy optimization in CPS. The section is divided into sub-sections based on the behavior of a CPS regarding the energy concern. In each sub-section, we define the concept and cite previous efforts based on it. The most common approaches in CPS found in the literature review are scheduling, reconfiguration, contextual-awareness, and user recommendation. Some of these solutions are related to the general ICT domain but have potential benefits in the CPS environment, in particular those related to migration, profiling, and software.

Scheduling

Scheduling is one of the solutions to optimize energy consumption in CPS. It consists of changing the time slot during which a specific task is executed [15]. The change of time is usually done to benefit from the energy when it is more green, for example, solar energy during the day. In addition, scheduling has a target of detecting unused devices and changing their mode between active, idle, sleep, off, and other modes.

IoT devices are usually powered by limited capacity batteries wherefore many ideas were developed to allow these devices to last longer. The authors in [29] propose that activity scheduling can increase energy efficiency while maintaining the reliability in addition to

the transmission protocol choice and the transmission power in an IoT environment. Each node can choose to switch between standby or active mode as long as it will not affect the efficiency of the whole system and that it will inform the other nodes. This solution is based on optimizing energy at the communication layer and at the entire device level while maintaining sensing.

With the increasing number of connected appliances in homes, the evolution of smart homes and smart cities, Home Energy Management System (HEMS) were introduced. A HEMS using ZigBee communication, smart outlets, and lights is proposed in [19]. The proposed approach minimizes the energy consumed by home appliances during standby mode, where the smart outlet cuts-off the current from a device consuming below a certain threshold. Their solution is simultaneous and the waiting time usually needed before turning off the outlet was eliminated.

The authors in [20] present an architecture for energy monitoring and saving functions that will encourage users to save electric energy by themselves and also will reduce standby power automatically. The home energy-saving system is divided into clients that measure electrical power and powers-off the devices connected to them if necessary and a server that has the role of monitoring and controlling these clients. This solution operates on the entire device by cutting-off electricity for unused devices.

Scheduling solutions also exist in ICT environments due to the high amount of resources compared to variable demand. They are proposed to solve the problems of efficiency and availability. SleepServers [22] is a software-based solution that creates virtual instances of hosts in a work environment. It allows hosts to run in a low power sleep mode but still be able to respond on the network and even run some applications. This approach allows to raise the availability and increase usability in sleep states. It is implemented in an ICT environment and acts on the service and communication layers.

The authors in [23, 16], proposed three control strategies for data centers depending on the level of coordination between the cyber (computational) and physical (thermal) components of the environment. They criticize that thermal properties are managed as pure physical, however, they have cyber and physical characteristics. A baseline, uncoordinated (divided into two separate optimizing problems one for the cyber component and one for the physical one), and coordinated (one optimizing problem) strategies are compared. It also introduced a cyber-physical index (CPI) that helps to choose a coordinated or an uncoordinated strategy depending on the amount of workload, servers, and cooling distribution in the data center. The data center was divided into server, zone, and data center levels.

In [24], a QoS-aware virtual machine scheduling method in a cloud environment named QVMS. The scheduling problem was defined as a multi-objective optimization problem: (1) minimize energy consumption, (2) minimize downtime, and (3) maximize

the resource utilization rate. QVMS uses a Non-dominated Sorting Genetic Algorithm III (NSGA-III) is used to find the optimal scheduling policy however their optimization resulted in slightly better resource utilization and downtime but the energy was almost not affected compared to other scheduling methods such as: (1) Benchmarking: move the application to the closest physical server using the shortest path algorithm, (2) Energy-aware virtual machine (VM) scheduling method: move the application to a server that has a higher performance or lower energy consumption [76].

In [77], an energy optimization approach for manufacturing environments was proposed. It was based on collecting real-time data. This data was used by an energy modeling artificial neural network and a monitoring one the output of these two modules are compared for scheduling and rescheduling tasks.

The authors in [25] proposed an agent-based routing approach for WSN. It achieved data aggregation and processing locally, in addition to the selection of the best routing path for mobile nodes to reduce unnecessary flows between the nodes and optimize energy. [26] also proposed an approach for reducing energy consumption by considering a data cleansing algorithm, an energy-saving scheduling algorithm, and a low-power protocol for communication in a WSN.

In [21], an approach that consists of compromising between the energy efficiency and the reliability of a system in both shutdown and scale-down scheduling techniques was proposed. It was divided into two techniques reducing the energy consumption of CPUs: (1) make a copy of the scaled-down task as a non-scaled task to recover it if necessary, (2) Specify minimum reliability constant for each task and make sure this value is respected.

Financial cost-effectiveness is the main concern in [28]. A problem of controlling a pump and scheduling a dishwasher and a clothes washer to minimize the energy cost, while preserving the desired water level, was solved using a Particle Swarm Optimization (PSO). It creates models for water and energy demands, water tank, energy production, and prices. It takes into consideration different variables (like the water need in an average household and the energy cost in the USA). Finally, it tuned the parameters using PSO to find the optimal time slot for running each of the appliances.

In [27], an energy management system that creates profiles for the home residents by taking into consideration factors like their gender, the number of occupants. It also takes into consideration the human behavior, flexibility region for each task in addition to the acceptance for the scheduling of each task. The goals of this research were electricity cost and peak power minimization while satisfying the comfort of the home residents. In [28, 27, 20, 19] scheduling is done on the entire device as one entity where they are changing the operating mode of each device based on its usage.

The authors in [17] proposed an occupancy detection and prediction framework using

a WiFi probe-based ensemble classifier in energy-CPS to minimize the energy consumed by the cooling and ventilation demands. This system can detect the presence of individuals in a room or even predict occupancy patterns.Devices schedules followed and were influenced by the schedules predicted for occupants. Their solution acts mainly on the communication layer of a CPS.

Occupancy is an important factor in human interaction with the heating, ventilation, and air conditioning (HVAC) systems because it determines directly the desired mode of the system as a real-time scheduling. It could affect also other CPS environments like the city infrastructure as in [18]. IoT devices were implemented to sense the environment and make smarter decisions by detecting the presence of people or cars and controlling an infrastructure of street lighting.

Reconfiguration

Reconfiguration is the ability of a system to change its initial configuration by tuning some parameters allowing it to adapt to changes that take place during run-time [78]. It is usually based on a finite number of states predefined during design time in contrast to autonomous systems that can accumulate knowledge with time.

The authors in [37] proposed multi-operating modes for each layer of the system to increases energy efficiency (33% energy gain). In this approach, the optimization of each layer is done by the layer itself and there is no interaction with the other layers to share information about their modes. Sensing layer optimization is done by having active mode and sensing mode where sensors are still detecting events but the transceiver that sends data is turned off. They found that the power consumption of the communication layer is high and can be divided into two ranges of transmission speed. For a speed lower than 10 Gbps, active and low power idle modes (Unable to transmit or receive data) are used. If the speed is higher than 10 Gbps, the energy cost of transition between modes is high. A fast wake intermediary mode was introduced to solve this issue (able to transmit but unable to receive data). Computation and control unit changes its frequency to create 4 operating modes: high-performance (highest frequency), low-performance, idle, and napping modes (Lowest frequency). Their paper did not include the used frequency values. Actuators are in sleep mode by default, they switch temporarily to active for accomplishing an action. Their proposed solution optimized the energy on the sensing, communication, computation, and actuating layers individually.

The authors in [36] present an energy-efficient reconfiguration tool built on Raspberry Pi in an intelligent transportation scenario. They identify energy-consuming concerns at design time, analyze configurations and variants, create a file containing all the possible configurations and the consumption of each of them, selecting initial configuration, and

reconfigure at run-time according to the context. It is based on the aspect-oriented programming by using the separation of concerns and adapting the configuration of each concern. For example, the compression algorithm can change during the run-time based on the size of the message that needs to be sent. In this solution, energy can be optimized on many layers by changing parameters based on the concerns mainly on the sensing layer (monitoring concern) such as sampling frequency. Other parameters also affect the power consumption on the services layer (software concern) such as compression protocols and on the communication layer (networking concern) such as exchange protocols.

The authors in [38] ensured energy optimization while satisfying the user comfort by proposing a smart zoning multiple-mode feedback system in a smart building based on the usage of each room, its orientation, and its occupancy. Each zone can be in normal, pre-cooling, or power off mode. The HVAC set points are configured dynamically in each room considering the indoor state (temperatures, occupancy schedule, availability of renewable energy) and outdoor state (weather conditions, weather forecasts). Their solution was able to save 15% of the energy while testing it on EnergyPlus [79].

In [80], an energy-efficient large-scale data collection and correlation technique was proposed. Their approach is used in an environment having sinks and correlation regions, it consists of changing the size of the region based on the distance between the latter and the sink based on the residual energy of the node. If the region is close to the sink then the region is increased, on the contrary, if the distance is far, the region is decreased. The authors focused on the energy used by the network layer in large data exchange systems.

The authors in [81] proposed a cloud-terminal-based CPS to reduce energy consumption in machining processes. The proposed architecture contains 4 layers that summarize the functionalities of the approach: machine level (raw data is collected using sensors), data level (data storage/management and prediction analysis is done), decision support level (contains optimization services), and control level (data presentation and on-board physical execution using controllers). The decision-making process was based on the PSO using GPU.

In [39], a machine learning energy manager for hybrid electrical vehicles (has an internal combustion engine and an electric motor) was proposed. A nested reinforcement learning approach was adopted where an inner-loop was responsible of choosing the electric and fuel engines to minimized the fuel usage, while the outer loop was in charge of modulating the battery health degradation. However, the impact of the reinforcement learning technique used was not considered.

Migration

Large distributed data centers also tackled the problem of energy efficiency by the use of VMs alongside with migration techniques [46].

The authors in [47] proposed the use of workload consolidation and VM migration as a way to minimize the energy of ICT equipment in a smart building. Migration is usually done for the processing and storage layer. Alongside their solution, they also proposed the necessity to use sustainable equipment like new lighting technologies by using LED lights, and the energy source using Solar PhotoVoltaics.

In [48], VM resource utilization (CPU, memory, and network) are monitored and estimated, overloads and underloads situations are detected, and dynamic VM consolidation through live migration is performed.

In [22], a light image of a computer plays its role on the network and maintains the functionality as if it is awake but consumes as much as if the PC is on sleep mode. This showed that 60% of the energy used by PCs in enterprise environments could be saved while maintaining user experience and computers' availability. Migration and resource sharing solutions may have potentials in CPS.

[49], proposed a live virtual machine scheduling in data centers in the form of a trade-off between performance and energy. A joint model for energy consumption and performance degradation of VM migrations was formulated. The method can be divided into three steps, (1) Identify the physical machine that has idle space for migration, (2) migration strategy searching to identify which VM to be migrated and to which machine, (3) identify the global scheduling strategy for all running VMs.

The authors in [50] proposed a relaxation-based algorithm for services replication in different nodes based on the stream flows and their occurrence rate aiming to reduce the communication energy between nodes in a WSN environment. The problem was formulated in the form of mixed-integer linear programming. Their algorithm chose which services should be deployed on which node and scheduled the flow in an optimal way.

Modeling and Simulation

System modeling is the process of conceptualizing abstract models and designs of a system. It allows respecting several requirements and dimensions of a system. It defines entities and the relations between them that give a comprehensible understanding of the functionality of the whole. System models aim to support analysis, specification, design, verification, and validation of a system [82]. A simulation is a method for implementing a model for validation, integration, verification, and limitations detection purposes [83]. Therefore, some studies aim to raise a device's efficiency while reducing its power consumption by

making changes in the design phase by modeling and simulating a system.

[84] showed that software engineers care about energy however, they are not successful due to lack of necessary information. It also proposed that the energy concern exists during different stages of software development. For instance, during the requirements specification, energy optimization is often seen as a threat to performance. During the design, developers do not consider energy scenarios due to the lack of scenario-aware tools. Ignorance of new energy techniques, the need for fine-grained tools for the whole system are the main reasons energy is forgotten during the construction of the software. Finally, when finding issues and making the maintenance of the application, developers are not concerned by the energy consumption unless it has a big impact, for example, battery drain.

[6] presented an energy-aware model-based approach to the development of a wearable medical device. It is based on the different layers compatible with this CPS. Design decisions were made after evaluating and redesigning different mechanical principles in the mechanical layer, evaluating different regulation algorithms (periodic and event-driven) in the computation layer, and evaluating the energy during different communication scenarios in the communication layer.

Authors in [12] presented a generic co-simulator for ICT and its power consumption. This simulator is distinguished by its architecture, modeling, and time management capabilities. They showed the important impact of ICT on the power consumption of a CPS (especially the networking layer). They also showed that energy-efficient choices of networking could result in lower functional efficiency.

In [7], a model-based design methodology for residential micro-grid was proposed. In addition to a simulation presenting the structural and behavioral parts of a residential micro-grid system. The model-based design was divided into four parts: (1) modeling the power grid, (2) modeling the grid management algorithms, (3) developing the cyber-physical co-simulator, and (4) verification and validation. Their simulator made possible the experimentation and comparison of electrical vehicle demand algorithms, residential demand response, and grid reliability.

In [8], an energy management framework used for autonomous electric vehicles in the smart grid was proposed. It is able to collect the real-time power consumption status and demand from autonomous electric vehicles and charging stations. Energy saving in this system was done during design (architecture and communication protocol). For the architecture part, path planning was made with respect to energy saving. For the communication protocol, an event-based control was used to reduce the communication between sensors, controllers, and actuators. This technique acts like the continuous state-feedback controller by establishing a communication between components after the occurrence of an event, therefore it reduced CPU and network bandwidth, thus energy reduction is guaranteed.

[85] proposed a domain-specific language for energy-efficient building modeling. It was based on the following elements: location, facility, sensor (including its unit and measured value), and rules. However, their modeling approach was missing actuators, energy feedback, and a more fine-grained description of the system.

In [13], a statistical model for simulating and validating different energy strategies in CPS in the railway CPS domain was presented. It was adopted to choose the most energy-efficient and reliable policies in such stochastic and critical environments.

The authors in [14] presented a framework to monitor, simulate, and analyze data center thermal performance and energy efficiency. It is based on the integration of WSN, building information modeling, and building management system. However, their proposed framework was not tested or detailed regarding the collected metrics and potential actions.

In [9], a multi-agent architecture for building energy management in CPS was proposed. It is based on 4 main layers: field layer, data acquisition, autonomic computing, and management. Collected data are contextual ones such as temperature, humidity, and light brightness. Actuators include switches, HVAC, and computers.

The authors in [10, 86] showed the impact of CPS workloads on its design and optimization. First, they identified the main complex characteristics of CPS workloads (such as self-similarity and nonstationarity), then, they proposed statistical equations for modeling these workloads. They also showed that better design optimization for some cost functions can be reached using the mathematical description of workloads such as resource allocation and power.

The authors in [11] proposed an energy cyber-physical model for cleaner manufacturing. Their architecture was divided into a physical-energy layer (IoT devices capturing energy-related data during the whole process of production), a cyber-energy layer (included data cleaning and data mining processes), and a data and knowledge-driven system layer (seen as the top-level layer that monitors, alarm, assess, and optimizes parameters). Their approach ensured not only the monitoring of electrical energy flow but also water and materials.

In [87], contextual collected data and machine learning techniques were used in the context of a cloth dyeing factory. Their goal was to identify and predict the process inefficiency from the functional point of view, therefore, lowering the possibility of repeating the dyeing process and energy waste. The process was adapted, parameters were tuned, and the laboratory pre-production phase was replaced by the real-time monitored data.

User Recommendation and Behavior

The human is a major component in the CPS. On one hand, he has continuous direct and indirect interaction with the environment where the system is implemented [88]. Some

researchers [89] include the human in the CPS loop as a service capability description model. They associated properties describing the role for each person, his capabilities, knowledge, availability, reliability, and interaction endpoints with the system for the CPS. On the other hand, user choices can be critical in the energy used by the CPS. For example, different applications performing the same task consume differently because of facts like synchronization with cloud, web-based energy inefficiency, heavy applications on startup before performing the task, continuous events are expensive (Spellchecker), user interface [30]. [27] showed that human behavior has a direct impact on energy consumption in residential CPS.

Encouraging users to change behaviors in a CPS is a new trend. CairnFORM aims at creating and encouraging new socially-shared practices by displaying energy data in collective and public spaces in the form of stackable interactive rings, where each ring corresponds to a defined time during the day. It is based on the current consumption of the tested area and the amount of renewable available energy [31].

[32] designed a community-scale energy feedback system having three main features: spatial, energy supply, and energy consumption. It is based on building power meters. It increased the visibility of building energy consumption using augmented reality for real-time and for historical data, in addition to the improvement of the power consumption over the years and the energy source for each building. This framework is considered as an open urban energy data because all these data can be shared from all the buildings to all the citizens of the community. The main drawback of this approach is the total lack of granularity.

A gamification approach that motivates the occupants of an apartment to take action and reduces their energy consumption by rewarding them by points was proposed in [33]. In addition to a framework that learns, models based on the players' decisions. An eco-feedback is proposed in [34]. It consists of mapping between the physical and virtual environment to motivate the user to take energy-efficient actions in a smart home. This is done by tracking the user in the house using sensors and knowing what are the activities of the user. Then, transforming the home to the form of a game and the energy-consuming appliances to enemies in this user interface. The authors in [35] presented a virtual pet game by reducing plug-loads in mid-size commercial offices. It showed that by using the game the workers reduced the energy consumption by 13% and even after using this game the workers had better energy-saving habits. Besides, findings showed that there was no rebound effect by the users of the game after they stopped playing it but it helped them build energy-oriented habits. This shows that this kind of solution could be useful for raising energy awareness regarding energy-aware behaviors. However, they are not a reliable solution because it takes a lot of effort and time to implement especially regarding

graphics and user interfaces.

Power Supply and Demand

Many approaches deal with the energy efficiency problem in CPS by changing the power supply source to renewable ones or matching demand to the supply in order to use the generated energy more efficiently [40].

In [41], a framework (called SG-CPS) capable of collecting ambient sensors values in order to predict the user demand and power supplied from renewable energy sources is proposed. It collects temperature, humidity, sunlight, and wind speed. It includes also two optimizers for purchasing energy decision-making (one uses linear programming and the other multi-stage stochastic programming). These two optimizers resulted in lower energy consumption but are computationally intensive. More investigations should be done on more metrics for prediction and lower resource-intensive optimizers.

The authors in [42] presented a framework for energy management in a building by considering different energy resources (especially renewable ones). They discussed also user choice and the use of a decentralized management system. However, their approach was based on production and demand and did not take into consideration a fine-grained view of the systems inside a building and its data.

In [43], an energy balancing system between different houses having solar panels and connected to the grid is proposed. This problem was solved using the minority game algorithm that deals with limited resources multi-suppliers issues. A multiple customer model is also proposed where a task that is done by the user can be inactive, active but unassigned to a supplier, active and assigned to the grid, active and assigned to battery or solar. It reduces the use of the grid and maximizes the use of green sources, however, the high ecological impact of batteries in the long term was not taken into consideration.

In the past few years, there has been an increasing interest in the research and development of Electric Vehicles (EVs) that are charged by connecting them to the grid to store energy. The essential advantage of these EVs is that the transition of electricity can be done in both directions, therefore a vehicle can push back electricity to power the grid from its battery. Their storage capabilities enable them to store electricity from renewable energy sources such as wind or solar and discharge when needed. This kind of vehicle is sometimes referred to as Gridable EV (GEV). They are considered as a distributed energy storage system. Vehicle-to-everything (V2X) is also a common research track. V2X covers a wide range of use cases, including vehicle-to-home (V2H), vehicle-to-building (V2B), as well as vehicle-to-grid (V2G) services.

[44] proposed the use of gridable vehicles in cyber-physical energy systems by storing the energy in these vehicles during off-peak and using it when needed. This approach was

intended to guarantee the minimization of cost and emissions using the PSO. Three cases where studies: (1) the vehicles charge and discharge randomly, (2) vehicles are charged from the conventional energy generation using load leveling, (3) vehicles are charged from renewable energy sources during off-peak hours and discharged during peak hours. Results showed that the smart grid model (using renewable energy) was the most convenient and resulted in lower costs and emissions.

[45] proposed the use of bidirectional energy exchange between vehicles and the grid in order to fill the gap between demand and supply. Their main contribution is the use of blockchain, contract theory, and edge computing to accomplish this task.

Software Related

Without any doubt, software has an impact on the power consumption of any system. Many research articles presented the importance of software development and the main issues faced by developers producing their low interest when it comes to energy [90].

The algorithm and complexity are the main factors that affect the energy drained because of the software. This kind of energy loss is due to the choices made by the developers while writing their code. [91], propose a model to estimate the energy consumption of a JAVA application during execution. The developed tool (TEEC) was tested on an optimized and un-optimized Java code and the results were validated by the Watts Up Pro Portable Power Meter. If found that memory should not always be neglected when compared to the CPU power consumption, whereas power consumption of hard disk could be neglected.

Another important choice while writing code is the choice of programming language. It is not only essential to develop in the most efficient, scalable, and compatible way, research shows that it also affects the power. [92], compare the energy efficiency and performance of the most commonly used approaches to develop applications (Java, JavaScript, C/C++) in Android mobile applications. It found that JavaScript saves more energy and is slower than the other approaches for benchmarks and that application hybridization may be a solution for application optimization, both in performance and energy consumption. For example, the difference between the most energy-consuming (Perl) and the most energy-efficient (C++ with O3) is equal to 25 463 joules which are considered quite high. An identical algorithm can have various energy consumption based on the language used to develop, based on the optimization level and the adopted design (recursive or iterative) [93].

In addition to the algorithms and the languages, it is possible by running the same software on different hardware to get different power consumption results [94]. This shows that the architecture on which the software is implemented also affects energy. For example, running an identical code on a micro-controller like Raspberry Pi consumes remarkably lower energy than on a laptop. [95] measured the power consumption of an application by

using different frequencies and changing the number of cores and found that using all the available cores with the highest frequency is the most efficient regarding energy consumption.

In the next section, we analyze and compare the reviewed approaches, and draw our observations of the current state of the art in energy-aware CPS.

2.4 Analysis and Discussions

In this section, we first introduce our comparison methodology, then compare and discuss the reviewed state-of-the-art approaches in two main categories: single-layer and cross-layer approaches.

2.4.1 Comparative Study

We define a list of comparative criteria that we use in table 2.2 to compare the reviewed CPS approaches:

Applied domain: CPS can be implemented in a variety of domains. It has applications in healthcare, transportation, smart cities, and many others. Most of the domains, where CPS are present, are highly dependent on electric energy to run and are essential to our daily life making us interested to identify in which domains researchers are thinking about energy as a concern.

General-purpose (GP): Determines the genericity or specificity of solutions regarding potential application domains. It provides a glance if the approach can be easily applied to any domain and if it is still efficient under different circumstances and different devices. Although most approaches are developed in a specific domain, some are easy to implement in another scenario.

Studied parameters: A set of energy consuming related metrics of each solution that researchers found significant and changing them leads to energy optimization. They can be found in all phases of building a system during modeling, developing, or operating phases. These metrics were studied by each of the research papers and it was found that they have an impact on the consumed energy.

Monitored layers: One of the most important comparison criteria because they show the levels of the CPS where these approaches were developed. It allows an understanding of which layers the optimization was done and if it is limited to it. Monitored layers can be any of the CPS layers defined in section 2.2.2. It can also be the entire device for solutions

that take all the layers of a device as one entity, entire system for the ones that consider the whole system as one entity, or contextual layer where the solution monitors only the changes in the environment.

Environment Heterogeneity (EH): It is defined by the presence of a variety of hardware and software frameworks that use different protocols, components, and architectures and that may be provided by different vendors. A heterogeneous computing system refers to a system that contains different types of computational units. In a CPS, the heterogeneity include but are not limited to architecture, functionality, components, vendor, protocol, and location.

System Flexibility (SF): System flexibility is the ability of the system to handle a dynamic environments. This includes the addition, update, and removal of devices or software/hardware components changes, as well as in the network topology. A flexible system is designed to adapt easily to changes and evolutions.

Level of Autonomy (LA): Autonomous systems are devices aware of their surroundings and can accomplish their tasks on their own without any intervention of the human. They can perceive, make decisions, and actuate a process in their environment using a control loop. We are also interested to know if the solutions are autonomous or not due to their adaptability with the changing surrounding. For the purpose of standardizing the comparison, we adopt the five maturity levels of autonomy defined in the Autonomic Computing framework proposed by IBM [51] will be used to classify related work. *Level 1:* Manual level: users perform all the management functions. *Level 2:* Monitor level: data from the managed resources is collected helping users to minimize the time needed to collect and understand information. *Level 3:* Analysis level: technologies are used to provide a correlation between metrics. It recognizes patterns, predicts the best configuration, and offers advice about potential actions to the user. *Level 4:* Closed loop level: environment can automatically take actions based on the available information and the knowledge about what is happening in the environment. *Level 5:* Closed loop with objectives level: high-level user-oriented business policies control the infrastructure operation in an automated manner. For example, non-autonomous solutions are usually predefined rule-based, need the intervention of the user, or are findings that can be implemented on the design time to make more power efficient CPS.

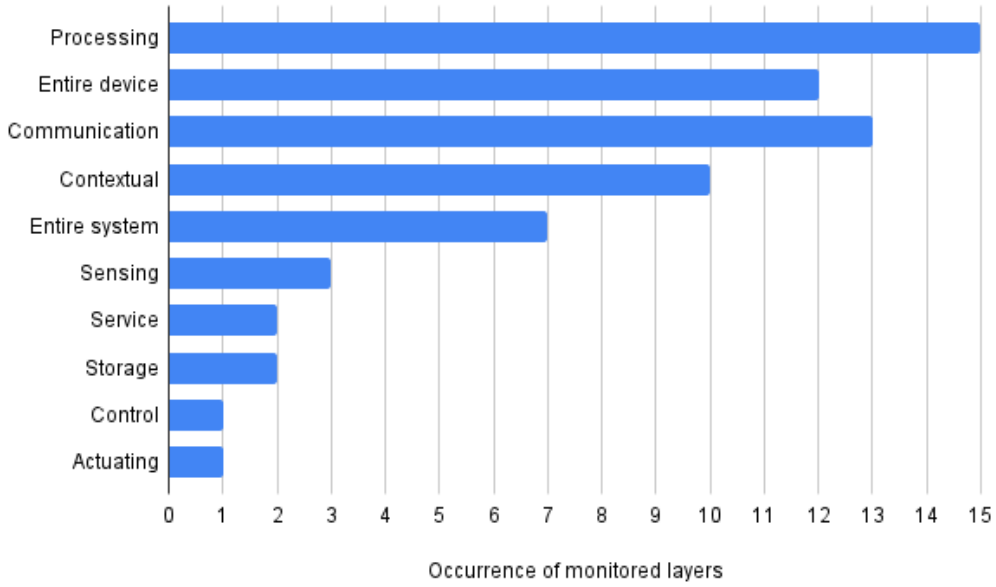


Figure 2.7: Quantitative comparison of monitored layers

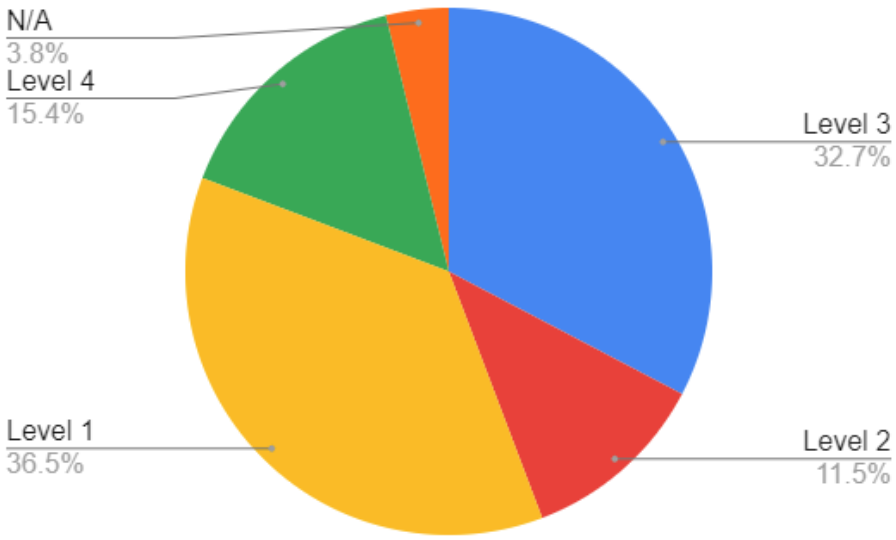


Figure 2.8: Quantitative comparison of level of autonomy

Table 2.2: Comparative study of energy CPS

GP: General Purpose, EH: Environment Heterogeneity, SF: System Flexibility, LA: Level of Autonomy *(H=High, M=Medium, and L=Low)*

Paper ID	**Applied domain**	**GP**	**Studied parameters**	**Monitored layers**	**EH**	**SF**	**LA**
[19]	Smart Home	M	Stand-by power	Entire device	M	M	3
[27]	Smart Home	L	Task schedule	Entire device	M	L	2
[22]	ICT	L	Entire system	- Communication - Service	L	M	1
[37]	N/A	H	- Operating Modes - CPU frequency - Networking speed	- Sensing - Actuating - Communication - Processing	M	L	3
[36]	Transportation	L	- Sampling frequency - Compression type - Archiving need	Entire Device (per concerns)	L	L	3
[96]	ICT	L	Software (Algorithm)	- Processing - Memory	L	L	1
[38]	Smart Home	L	Human presence	Entire System	L	M	3
[18]	Smart City	H	Human presence	Entire Device	H	L	3
[6]	Smart Home	H	Event-driven communication	- Processing - Communication	H	L	1
[32]	Smart City	M	Human behavior	Entire system	H	M	2
[33]	Smart City	M	Human behavior	Contextual	H	M	1
[34]	Smart Home	M	Human behavior	Contextual	H	M	2
[35]	Smart Office	M	Human behavior	Entire device	L	M	1
[29]	IoT	H	- Transmission protocol - Device state	Communication	M	M	4
[28]	Smart Home	L	Task schedule	Entire device	L	M	3
[17]	Smart Home	M	Wifi exchange	Communication	M	M	1
[47]	Smart Office	L	Task migration	- Processing - Storage	L	H	3
[48]	ICT	L	Task migration	- Processing - Storage - Communication	M	H	3
[86, 10]	N/A	H	Workload	- Processing - Communication - Control	H	H	1
[31]	Smart City	M	Data visualization	Contextual	M	H	2
[44]	Transportation	H	Energy sources	Entire device	M	M	3
[23, 16]	ICT	M	- CPU - Temperature	- Processing - Contextual	L	M	2
[24]	ICT	L	Task scheduling	Entire device	L	H	4
[45]	Transportation	H	Energy storage	Entire device	M	M	N/A
[12]	ICT	H	- Load - Communication protocol	Communication	H	L	1
[77]	Industry	H	Task schedule	Contextual	H	L	3

GP:General Purpose, EH:Environment Heterogeneity, SF:System Flexibility, LA: Level of Autonomy *(H=High, M=Medium, and L=Low)*

Paper ID	**Applied domain**	**GP**	**Studied parameters**	**Monitored layers**	**EH**	**SF**	**LA**
[8]	Transportation	M	Communication protocol	- Communication	L	L	1
[41]	Smart City	H	- Contextual data - Energy source	Contextual	H	M	4
[7]	N/A	M	Simulation	Entire system	L	L	1
[11]	Industry	M	Energy sources	Contextual	M	L	3
[80]	ICT	H	- Data correlation - Nodes distance - Residual energy	Communication	L	H	4
[49]	ICT	L	Task migration	Entire device	M	L	4
[21]	ICT	L	- CPU frequency - Task scheduling	Processing	M	M	4
[50]	IoT	M	Location	- Processing - Communication	L	M	3
[42]	Smart City	H	Production/Demand	Entire system	L	M	1
[26]	Healthcare	H	Data cleansing	- Communication - Processing	L	M	3
[87]	Industry	H	Contextual data (not mentioned)	Entire system	M	M	3
[14]	ICT	H	Temperature	Entire device	L	L	1
[85]	Smart City	M	Modeling	Sensing	L	L	1
[13]	Transportation	L	Modeling	Entire device	L	L	1
[25]	IoT	L	- Routing protocol - Task scheduling	Communication	M	H	3
[9]	Smart Home	M	Contextual data	- Sensing - Contextual	H	M	N/A
[43]	Smart City	H	Energy sources	Entire system	M	M	3
[81]	Industry	H	Contextual data (not mentioned)	Entire system	L	M	4
[39]	Transportation	L	Contextual data (not mentioned)	Contextual	L	L	4
[91]	ICT	L	Code Optimization	Service	L	L	1
[95]	ICT	L	- Frequency - CPU cores	Processing	M	L	1
[92, 93]	ICT	L	Prog. Language	Processing	M	L	1
[94]	ICT	L	Processor Arch.	Processing	M	L	1

The following two section analyze energy-related solutions, first, by identifying what are the solutions proposed to optimize the energy consumption of each layer, as seen in figure 2.7. Then, by finding cross-layer exchange potential to adapt and optimize the system from a holistic manner, in the next section. In both cases, we found that autonomous solutions are rare and that most solutions proposed for CPS are based on scheduling techniques, acting mainly on the entire device or system by changing its mode between active, sleep, and power off to minimize the energy [19, 20].

2.4.2 Independent Layer Approaches

A high number of recent research papers are based on user feedback but they are not autonomous. However, we find that most of the proposed approaches in the literature deal with energy issues on a limited number of layers. For each of these layers, we identify the potential energy hotspots and the applied energy optimization solutions, based on the comparative study in table 2.2. In this section, we discuss the energy management approaches by dividing them based on the previously identified CPS layers.

Entire device

Most solutions propose reducing the energy of an entire device, these solutions are based on scheduling and switching between various operating modes. Even if a high percentage of energy is consumed by devices in idle modes, these solutions optimize energy only when a device is unused. However, no optimizations are applied during its active state, making such approaches less than ideal. In [19, 20] no optimizations are being done while a device is running for example it is impossible to maintain the sensors of a device running while it is in a low power mode. A lack of autonomous flexibility is present in the last two solutions, due to the need for the user to change the configuration of the system each time a device gets connected to a different outlet. They also use hardware measurement tools which allows them to measure the whole device with high precision, however, making them more expensive and difficult to deploy. The main drawback in [27] is that not all tasks can be rescheduled automatically due to their diversity and the inconvenience for users to perform the tasks at an involuntary precise schedule. In [42], the entire system is considered as one entity, production and demand and did not take into consideration a fine-grained view of the systems. In other papers such as in [43, 45, 44], the long-term impact of batteries is not considered.

Services layer

The software can include potential parameters related to the code, for example, the complexity or the number of bugs and violations by the developers. On this layer, many research papers highlight the lack of tools and knowledge of the developers about the impact of their code [84]. Some real-time energy bugs scanners exist but are not commonly used across the community. We argue that energy awareness concerns should be introduced to would-be developers in their education curriculum. We also find that another energy concern is the choice of algorithms in software, as some algorithms are more optimized than others (for the same workload), thus leading to energy reductions [91]. The choice of the programming language also has an impact on the power [92, 93]. In [93] the same application developed using Perl programming language consumes more energy than developing it using C++. This kind of decision needs to be done during the design and the development of the applications running on a CPS but will have their impact during the run-time.

Processing layer

A collection of applications and services usually run over the processing layer. The processing layer is one of the main layers of a CPS, responsible for analyzing data captured by sensors and running the applications on the system. Many research shows that running the same application using different processors leads to different energy consumption due to various reasons processor architecture being the main one as in [94]. By changing the frequency [37, 21], the number of active cores [95] in a CPU, the algorithm [96], the workload [86, 10, 16] or the data cleansing [26], energy minimization can be reached. Generally, this is possible due to lower the time needed to accomplish a task, but this is not always the case because this can lead to a power increase even if the time decreased. Another important factor for software is the hardware on which it is running [48, 47]. Hence, making us think about the optimal device to run the analysis, in the case of a system-of-systems most of the processing is either done locally or remotely on a cloud. We believe that running the analysis can also be done in nearby devices in the same CPS if these devices consume less. Most software solutions are done on computers or servers in the domain of ICT. We argue that more research should be done in CPS due to the high potential of using software layer energy optimization for more energy-efficient systems.

Storage layer

The storage layer consists of the equipment responsible for storing the data and keeping them available in order to be processed. Data can be locally stored in SD cards and hard disks or remotely in the cloud. On one hand, storing data on the cloud creates sometimes

an illusion of zero energy but it is essential to mention that sending the data to the cloud may result in a more significant energy usage. Firstly, by using the networking medium, secondly, by keeping the data centers that host the cloud servers operational. On the other hand, green data centers use energy-efficient technologies such as low-power servers, modular data centers, free air cooling, renewable energy sources, and other sustainable technologies. Hence, it is essential for each piece of data to choose ether to store it locally or on the cloud.

Communication layer

The communication layer is the medium to exchange data and events between devices of the same system, between IoT devices (used as sensors or actuators), or between the system and external sources such as the cloud or web services. Reducing energy in the communication layer is also limited to a variety of operating modes based on the used bandwidth. We think scheduling should not only be used to change modes but also to accomplish tasks during the optimal time. For example, if a device needs to download or upload data through the network, it would be more convenient to perform the network transmission when the device is plugged in to an outlet, and the electricity produced is from renewable sources. Communication frequency and protocols are also found to have energy impacts. In [17], occupancy detection using the communication layer has one major advantage because there is no need to implement new sensors in the environment for the system to understand the context.

Sensing and actuating layer

The sensing layer monitors the environment of the CPS using physical and virtual sensors. The actuating layer is the medium used to execute the changes planned by the system. This layer can either be physical (*e.g.*, a hardware sensor) or virtual (*e.g.*, a logical or software sensor). The main concern on these layers is the optimization of idle power consumption. In [37], the authors proposed to have multiple operating modes where the transceiver is turned off because this part of the sensor consumes the most. This approach can be considered efficient if the sensor has a transceiver, but in many cases, the sensor is part of a connected object having one transceiver for its entire operations. Having flexible sampling frequencies is another solution [36]. However it is trickier to use as it needs to be well understood in order to increase or decrease the frequency, with some situations where a high sampling rate is necessary to obtain reliable information [97].

Contextual layer

The contextual layer is the knowledge and understanding of what is happening in the environment of the system. It is usually built upon the observations of the sensing layer, and also includes human activities and behavior [33, 34]. Others use contextual data such as power sources [41, 11]. During our literature review, we first found that the main contextual information collected for energy optimization is the detection or prediction of the presence of the user, also called occupancy. However, studying user behavior should not be limited to presence detection. To reach higher energy-saving it is essential to understand all the human actions in the environment (such as the time they use each device, their presence, their sleeping schedules, and others factors). This leads to an efficient reduction in CPS idle mode power. In [38], one drawback is that their solution takes into consideration occupancy but does not consider users' interactions with the building like opening a door or window. The environment where CPS are implemented is heterogeneous and rich with information that can be collected [39, 81, 9], however, it only focuses on the functioning loop of the system through its touch-points (sensors and actuators). For example, information such as: what is the user doing and what devices being used. This can be solved by a cognitive system that understands the holistic view of the system and all the aspects of the environment. Second, most solutions that consider the user focus on user recommendations and are not fully autonomous as seen in figure 2.8. Autonomous systems are needed as these solutions optimize energy automatically and in real-time without the need for continuous human intervention. Sharing all these user behavioral information raises a critical concern of security and privacy. Secure and encrypted communication should be used to share this information, with a priority to limit data sharing to local devices.

2.4.3 Cross-layer Approaches

Layered communication has a strong separation in the definition of each layer where the interaction between them is strictly controlled and limited to the necessary data to operate the system. In contrast, cross-layer approaches are protocols or architectures where the information of each layer is highly shareable with others. This way the information becomes more valuable and useful to accomplish better performances [98], in our case, lower energy consumption. CPS have a variety of energy-consuming concerns due to their various functionalities. These concerns can be found on an individual layer as seen in section 2.4.2 as well as on a combination of more than one layer .

First, the review of previously proposed CPS architectures showed a lack of energy concerns while defining the layers of the system. In particular, the optimization of software and contextual layers are not energy-aware. Many layers are used in a CPS and can be

energy-consuming. We argue that the way researchers and practitioners model a system without taking energy into consideration, is a major limitation for sustainable CPS. This might be due to a lack of understanding of energy in CPS. Hence, there is a need for an energy-aware CPS architecture able to identify all potential energy-consuming layers.

Second, as discussed in the previous section, solutions to optimize energy consumption mostly target single layers. We observe a lack of cross-layer energy optimizations. In [37], each layer has its operating modes but no data exchange between the layers to share their modes. The system does not exchange data between layers but optimizes each layer independently regardless of what is happening in the other layers. In [36], a huge amount of efforts is needed from developers to specify each of the energy potential consuming concerns, making it complex to implement. Then, they create a file containing a series of configurations and their energy consumption. These approaches are limited because they are not flexible if a new functionality is added to the system or if a new device joins the CPS. They are also limited by the managed or collected metrics and can not change on run-time.

We observe a lack of holistic solutions at the levels of the device and the entire system, which we argue is needed to build sustainable CPS. For example, each layer on a device can be optimized individually and then exchanging knowledge and information between different devices of the system leads to a holistic energy-efficient management and a decrease in energy consumption. This leads to a scalable solution in complex system-of-systems. For example, in the context of a smart home, bulbs can minimize their energy by detecting occupancy and the time of day. They can then share their status with the refrigerator constantly opened by the user. This is done in order to know if the fridge needs to to turn its light on or whether the light in the room is enough to illuminate the content of the fridge. In fact, a cross-layer exchange is already present in all CPS, as a CPS includes sensing, actuating, communication, storage, processing, and application/services layers. Most CPS sense information from the environment, analyze the data, the apply actions through actuators to the same environment. This is achieved in one direction and is oriented to satisfy the functionality of devices. Current shared data are limited and are not related to energy. Sharing energy-related data in addition to the operational one leads to a higher understanding of the environment and raises the ability of decision making. Shared information would include data captured from sensors, in addition to the state of each layer in a device, and contextual information.

2.4.4 Recommendations

Energy optimization in CPS is not yet fully mature, notably due to the lack of holistic and autonomous solutions. Therefore, we draw the following recommendations that we consider essential for energy-aware CPS solutions:

- Researchers should address the energy from a holistic point of view by studying the possibility of optimizing energy by collecting various data from different layers in a CPS. They should also investigate if collected data from a layer can help having an impact on reducing the energy of other layers.
- Large scale distributed CPS with hundreds of heterogeneous devices that can contain each tens of processors, in contrast with the simple ones with one processor, have a significantly bigger complexity, and need to be better studied. Especially that in this kind of constantly changing environment scalability is a major concern.
- The energy management in modern CPS needs to be fully automated or autonomous, and be able to adapt to changes in the environment with little or no human intervention. These autonomous adaptations also need to respect users' preferences and comfort levels, in order to avoid being ignored in favor of more energy-hungry solutions. Autonomous approaches can use artificial intelligence, machine learning techniques, or adaptive control systems in order to achieve their goals.
- CPS architectures need to take into consideration the contextual data and activities in its surrounding environment, in order to efficiently optimize energy consumption. The goal is to build a comprehensive and holistic approach that collects information from multiple layers of the system and apply cross-layer adaptations to reach optimal energy consumption.
- Researchers could improve the energy consumption of idle devices, either by reducing their energy to the minimum or by reusing their unused capabilities to lower the energy costs of other devices.

2.5 Summary

As CPS are gaining an increasing role in our daily activities, their energy impact is also rising and becoming a major concern for their sustainability. In this chapter, we reviewed the state-of-the-art energy-aware CPS approaches and compared them according to several criteria aimed at the studied parameters, monitored layers, and the level of autonomy of each of the approaches. We found that the most common layers of CPS are sensing, communication, actuating, processing, services, and control. We presented a literature review of the previously proposed solutions that deal with the issue of energy optimization in CPS and categorized them. Then, we explored energy approaches for each of the application, processing, storing, communication, contextual, sensing, and actuating layers. We also identified the approaches covering more than one layer and discussed cross-layer solutions and their advantages in order to build a holistic energy-efficient system.

In short, our findings are the following: 1) We have detected a lack of the studied metrics not covering all potential energy drain sources in a system. 2) Solutions deal with limited layers in a restricted number of devices mainly due to the challenges of exchanging data in a heterogeneous environment. 3) Most solutions are not autonomous. However, some could be considered to have a limited degree of autonomy. 4) Flexibility is highly limited to the devices and the environment in which they were developed, therefore, they are mostly domain specific.

These drawbacks motivate the needs to continue the research in order to find a better approach that can fill the previously mentioned gaps of state of the art. Therefore, in the next chapter, we present the architecture of our automated energy management framework based on power estimation, knowledge representation, and reinforcement learning techniques. Our contributions are detailed in the next part of this thesis in Chapters 3, 4, and 5.

Chapter 3

Energy Management Approach Architecture

Contents

3.1 Introduction

The results of the literature review on energy management in CPS reveal a number of shortcomings. Current approaches and commercial products fail when it comes to flexibility, level of autonomy, heterogeneity of devices, number of considered metrics, and genericity of the solutions. To fill this gap, we present in this chapter an automated energy management framework based on power estimation, knowledge representation, and reinforcement learning techniques. This chapter focuses on the design of a framework aimed at being deployed in CPS deployed in connected environments. Such a framework contains all the necessary elements to achieve higher energy management and has been designed to be easily enhanced in order to implement advanced and specialized management features.

In order to formally define the requirements of the system, we follow the Model-Based System Engineering (MBSE) methodology proposed by ARCADIA and develop the architecture of our approach using the open-source tool named Capella. This methodology allows to specify the various actors and stakeholders, as well as, to specify and analyse the requirements. It helps identifying well-adapted cyber-physical solutions in order to cope with our research goals.

In the remainder of this chapter, we take a look at the Architecture Analysis and Design Integrated Approach (ARCADIA) methodology. Then, we define functional and non-functional requirements. We also use Capella to formally model different components of our proposed approach on the operational, system, and logical levels. The physical architecture and implementation will be presented in chapters 4 and 5. Finally, we present some of the development choices that we made during the modeling of the system.

3.2 ARCADIA Method

ARCADIA is a structured model-based engineering method that aims to define and validate the architecture of complex systems [99]. Figure 3.1 shows the four different yet interconnected modeling levels of ARCADIA. The first two levels (operational and system analysis) ensure the needs understanding and the following two (logical and physical analysis) model the solution.

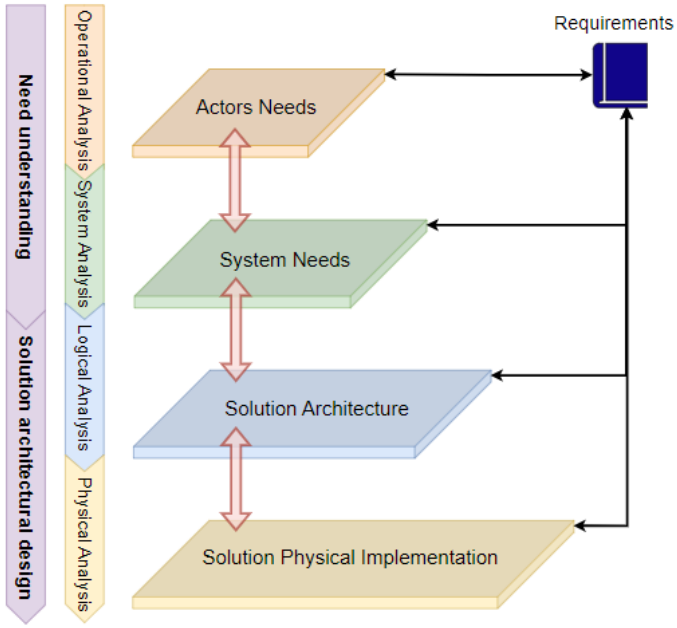


Figure 3.1: ARCADIA modeling levels

The operational analysis allows to specify actors and business processes, and to identify the needs of a system. The system analysis models the system as one component and the functions to be implemented in order to fulfill the operational needs. The logical analysis defines the logical components (independently from the physical technology that will implement the required system functionalities). It is considered the first level of internal components architecture and is unrestricted to one specific implementation. The physical architecture defines the concrete components of the system (e.g., hardware, software, interfaces, and connections). Different diagrams are used to model the system on each of these levels. Capella is a MBSE open-source solution that follows principles defined by the ARCADIA method. It is based on a graphical modeling workbench that provides rich methodological guidance for system architecture design.

3.3 Requirements Analysis

Requirement analysis represents the process of determining actors' expectations from a system. Requirements must be quantifiable, relevant and detailed. Based on the problem statement, functional and non-functional requirements are identified and will be respected during all modeling phases.

3.3.1 Functional Requirements

Functional requirements define what a system should be able to do. Each of these requirements should describe a function of the system or its components. The definition of functional requirements aims to make modeling choices during the modeling phase and to ensure the proper functionality of a system during the validation phase. In the following, the different functional requirements of our proposal are detailed:

- The system shall manage devices in a way that reduces power consumption of the environment on a holistic level. It is able to choose and execute proper actions in order to achieve this goal.

- The system shall be able to deal with a high variability of data types and formats collected from multiple heterogeneous systems and devices.

- The system shall represent and manage energy-related knowledge in a flexible and scalable environment. It is able to understand and exploit definitions and relations between devices, metrics, actions, and power consumption.

- Users shall be able to express their preferences and the system should respect these preferences in order to reach higher user satisfaction. It is also able to adapt to changes in the behavior of users with time.
- The system shall be able to estimate power consumption per device and generate power estimation models for new devices whenever existing ones are not accurate. It also stores and manages power estimation models for each collection of devices.
- The system shall share knowledge about power estimation models with other systems and actors. It also ensures higher visibility of power consumption per device.

3.3.2 Non-Functional Requirements

Non-functional requirements define desired properties and qualities of the system. The definition of non-functional requirements allows to specify the performance expected of a system. Figure 3.2 shows the systems' non-functional requirements. In the following, the different non-functional requirements of our approach are detailed:

- *Efficiency:* The system can execute a task successfully and accomplish its goals without wasting time or energy. The power consumption of an environment integrating our solution should be significantly lower than traditional solutions. The system also provides correct and real-time responses to occurring events.
- *Reliability:* The system performs accurately during the functioning period. It also ensures consistency under normal and abnormal circumstances. It responds intelligently to unexpected changes in the environment.
- *Flexibility:* The system is able to adapt to changes that take place in the environment, whether modifications occur in human behavior, environment topology, or device hardware/software updates.
- *Genericity:* The system is not limited in terms of potential application domains. It should be easily applied to any domain and maintain its efficiency under different circumstances.
- *Re-usability:* The system should be modular and machine-independent. Each component of the system should be reusable. Changes that occur to each component have a limited impact on the functioning of the rest of the system.
- *Scalability:* The system is not limited in terms of the number of devices. It can scale up or down with devices joining or leaving the environment. The system is also able to improve by adding new functionalities without disrupting existing ones.

- *Autonomy:* The system has self-managing capabilities allowing it to operate in complex, open-ended environments with a high level of freedom. It can perceive, learn and act with self-awareness of its surrounding.

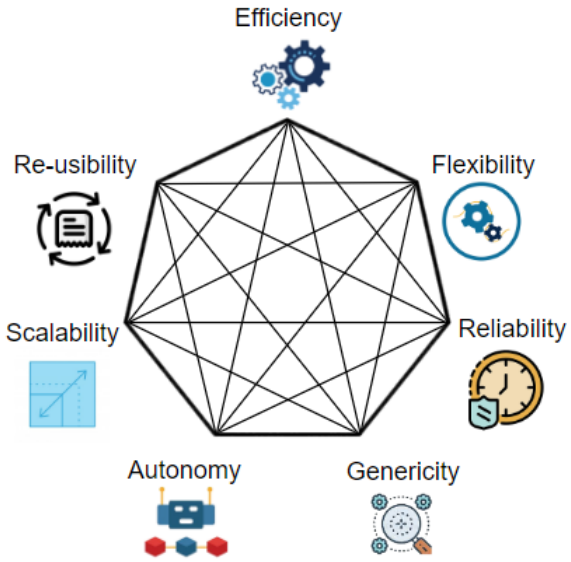


Figure 3.2: Non-functional requirements

3.4 Operational Analysis

The operational analysis focuses on defining the needs and goals that actors want to accomplish. It helps defining the problem. First, we defined the operational entities and actors used in all modeling phases. The composition of each of them is also defined. Actors and entities are external to the system and interact with each other as well as with the system. At the end of this phase, we describe the needs of actors and entities, their operational capabilities, and their activities.

Users are also actors and represent the main actors that interact with the system. A user is an individual that performs tasks affecting the environment. E.g., a home occupant in the case of a smart home scenario or a company worker in the case of an enterprise environment. Tasks are either applied directly to a device (e.g., a user changes the state of a device) or have consequences that are captured by sensors (e.g., a user leaves the room).

Devices are the main entities operating in the environment. All types of devices are modeled as devices. However, devices can have many types: sensors, actuators, appliances, HVAC, subsystems, meters, and intermediary devices. Figure 3.3 shows the different types of devices through an operational entity breakdown diagram. Sensors have sensing capabilities (e.g., temperature sensor, door sensor, and presence sensor). Sensors are not only

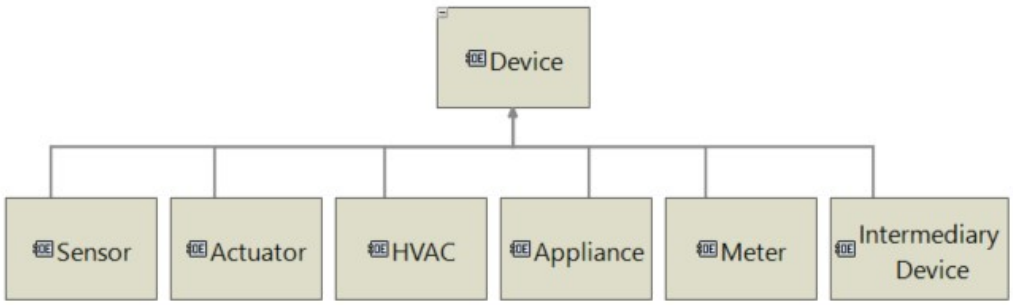


Figure 3.3: Operational entity breakdown diagram

hardware components, they can be software sensors (e.g., CPU utilization sensor, Wifi utilization collector). Actuators are devices that execute actions in the environment (e.g., light control, curtain control, and speakers). Heating, Ventilation, and Air Conditioning (HVAC) are systems used to control the temperature in a residential, industry, or corporate environment. Appliances are pieces of equipment designed to perform a specific task (e.g., refrigerator, washing machine, and vacuum cleaner). Meters are similar to sensors but are used to measure the power consumption of any device (e.g., watt-meters). Intermediary devices are used to transform a traditional device into a smart one. It does not provide all the functionality of smart devices. However, it guarantees a certain level of observation and control on the device (e.g., a smart plug that allows turning on and off an appliance).

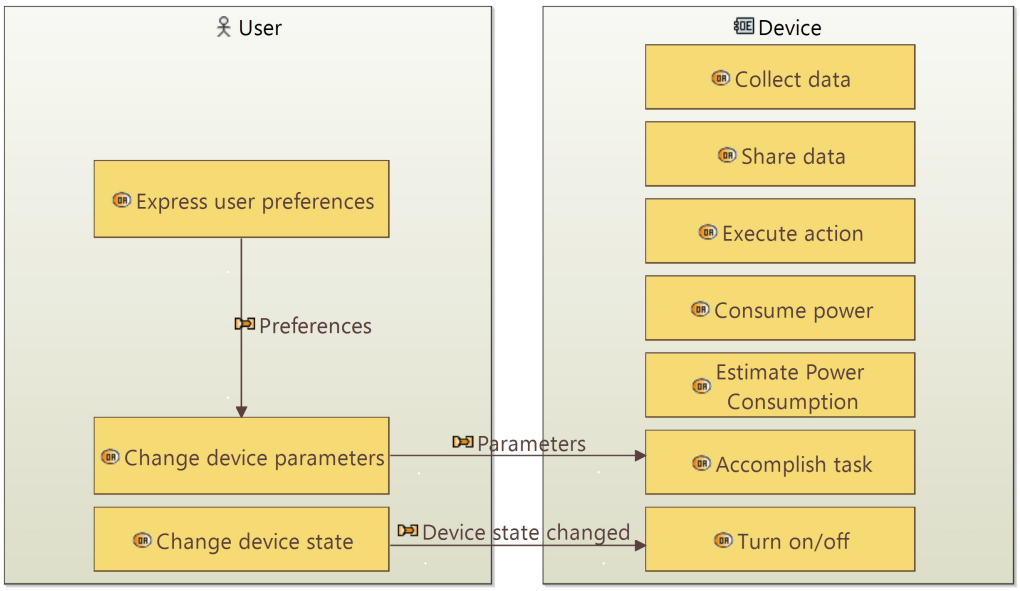


Figure 3.4: Operational architecture diagram

Figure 3.4 shows the operational architecture diagram that aims to identify the opera-

tional activities of actors and entities. In this analysis, we have specified the most relevant features related to our proposal. Users perform three main operational activities: (a) express preferences, (b) change device parameters according to preferences, and (c) change the states of devices. A device's main activity is to accomplish a particular task. They are also able to collect data internally or from their surrounding, share this data with other devices depending on their communication protocol, execute actions internally or on their surrounding, consume energy, and undergo state changes (e.g., turn ON/OFF) or parameter configuration (e.g., change the temperature of a heater). In addition, some devices are smart enough to estimate their power consumption.

3.5 System Analysis

System analysis focuses on the system itself and what it has to accomplish for the users. It consists of a functional analysis that begins by identifying the system's missions and capabilities in response to the needs of actors/entities based on the operational analysis. At the end of this phase, we describe functions, functional chains, and scenarios. In addition, it details the interactions with the users and external systems. Throughout this phase, the system is modeled as one entity (internal subsystems/components are not identified at this stage).

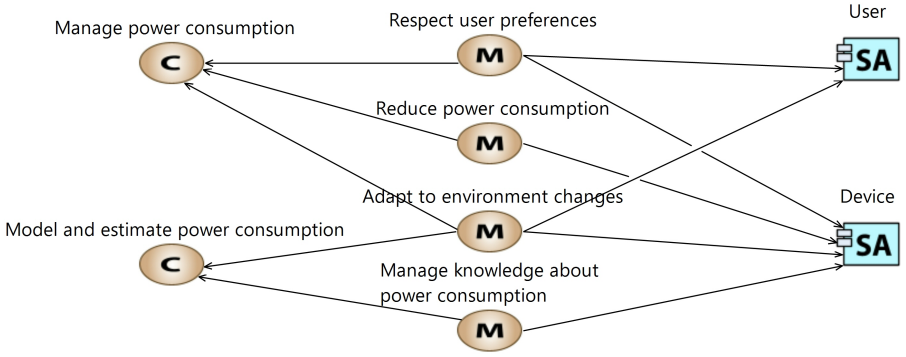


Figure 3.5: System missions and capabilities diagram

The main capabilities and missions of our system are illustrated in Figure 3.5. The principal missions of the framework are the ability to manage knowledge about power consumption per device, reduce this power consumption, respect user preferences, and adapt to environment changes. These four missions are mapped to two main capabilities: (1) model and estimate power consumption and (2) manage power consumption. The system should be capable of generating models and estimating the power consumption per device based

on collected data. It should also be capable of ensuring a power management functionality that respects user preferences.

Capella provides several scenario diagrams at each level of the ARCADIA method. One of the most important scenario diagrams is the system exchange scenario. It focuses on the lifelines of actors, entities, and the system that exchange messages and perform functions. Each exchange scenario defines a functional chain. In the following, we detail and model each of the aforementioned system capabilities using system exchange scenarios and system architecture diagrams.

3.5.1 Model and Estimate Power Consumption Capability

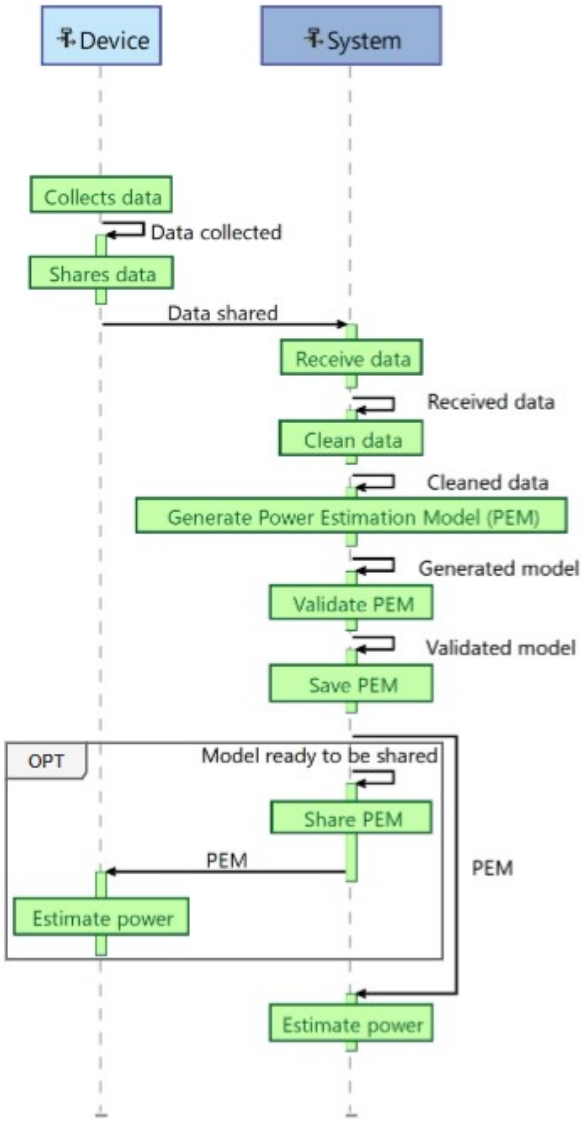


Figure 3.6: System exchange scenario diagram - Model and estimate power consumption capability

Accurate power measurement is the key to energy efficiency because it provides reliable quantification methods for power consumption. Measurement technologies should be reliable over the long term and adaptable to heterogeneous devices. However, current measurement and estimation methods are limited in terms of adaptability to changes. Therefore, the first capability of the system is to model and estimate power consumption for a large number of devices in an automated manner. In addition, share these energy estimation models across users and devices.

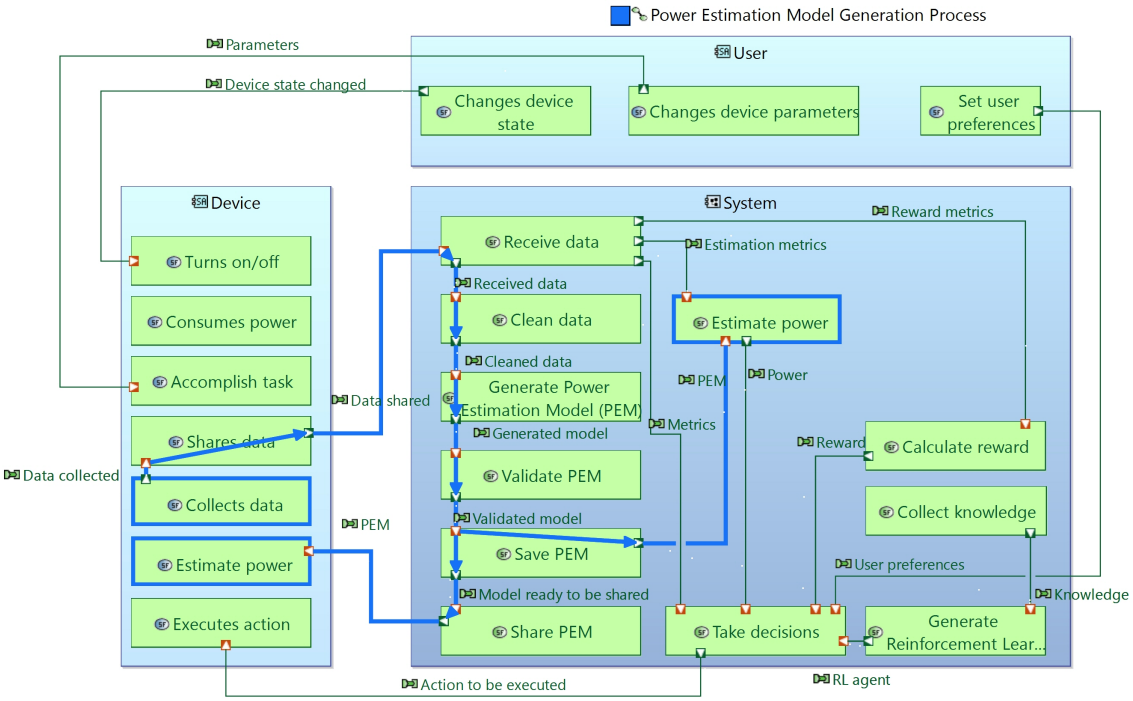


Figure 3.7: System architecture diagram - Power estimation model generation functional chain

Figure 3.6 shows the exchange scenario of the power consumption modeling and estimation capability. First, the device collects data and shares it with our system. Our system receives the raw data, modifies, and removes incorrect, incomplete, irrelevant, corrupted, and duplicated data. It generates a power estimation model using the cleaned data. Then, the system checks if the generated model is valid and that its error rate is under a certain threshold. If the model is valid, it is saved and used later. Otherwise, it will be discarded. The system can estimate the power consumption by itself or share the model with other devices/systems in order to estimate the power consumption of a specific device. The latter is an optional functionality for devices that are smart enough to exploit it. Capella generates automatically the functions defined in the scenario diagram in each corresponding actor, entity, and system. Figure 3.7 shows the generated functions and the functional chain of the power consumption modeling and estimation capability.

3.5.2 Manage Power Consumption Capability

Power management aims to reduce power consumption. This not only reduces the amount of electricity used, but also reduces the effort and concern of end-users around controlling devices. Reducing power consumption also extends the usefulness and the average lifespan of devices. The central goal of our approach is power management, therefore, we model this functionality by identifying exchanges between devices, users, and the system.

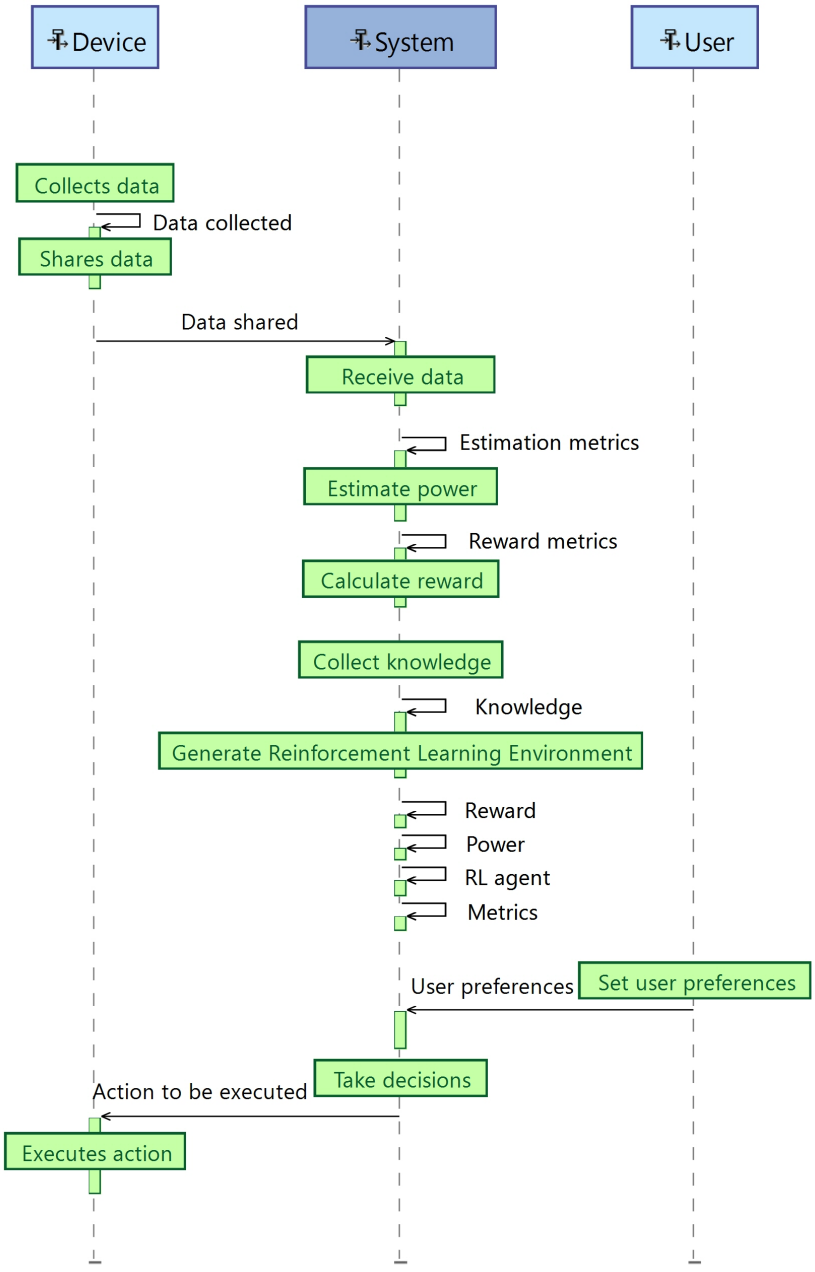


Figure 3.8: System exchange scenario diagram - Manage power consumption capability

Figure 3.8 shows the exchange scenario of the second system capability: manage power consumption. First, devices collect and share data with the system. The system receives data that will allow it to estimate the power consumption per device and calculate the reward for each accomplished action. It also collects knowledge and generates a reinforcement learning environment based on this knowledge. Then, user preferences are gathered. Finally, the system uses all of the above functions in order to take decisions and send the execution commands back to the devices.

Figure 3.9 shows the functions generated by Capella for the second system capability, power consumption management, in the form of a functional chain.

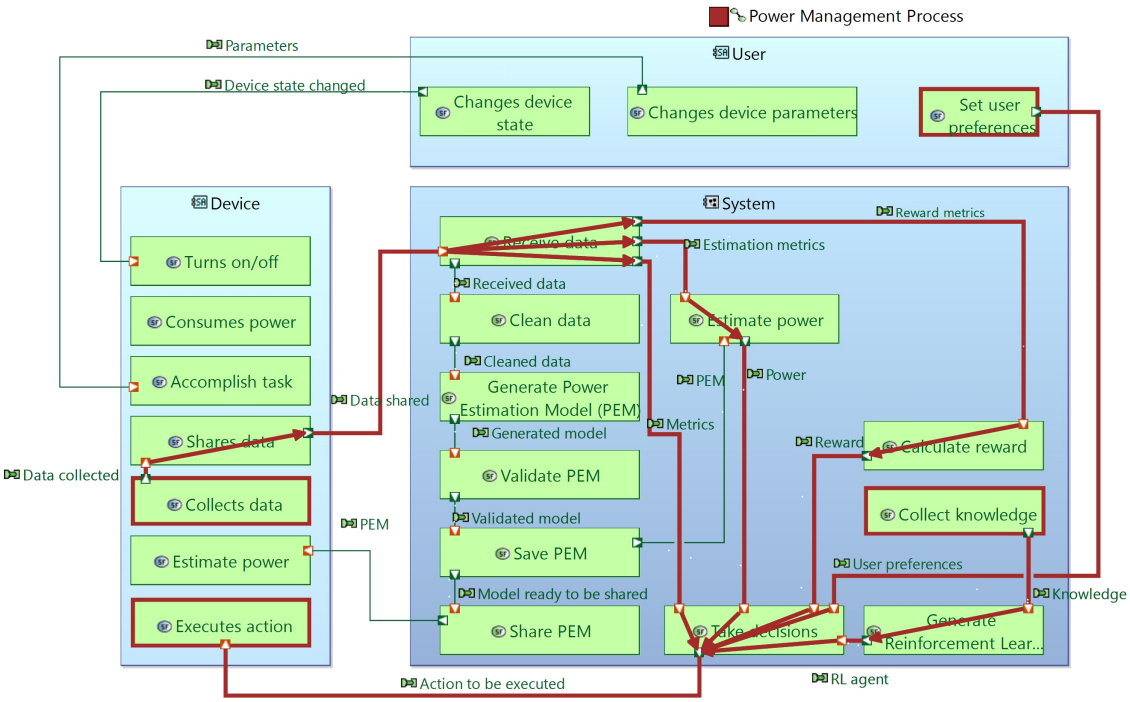


Figure 3.9: System architecture diagram - Power management functional chain

3.6 Logical Analysis

The logical analysis focuses on the decomposition of the system for development, integration, reuse, product, and configuration management. It defines how the system will operate to fulfill expectations. At the end of this phase, a logical architecture details functional, components, and interface descriptions, along with a formalization of the way they are integrated into the design of each component.

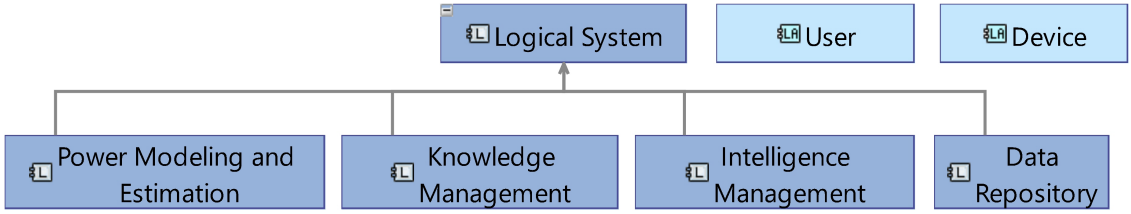


Figure 3.10: Logical component breakdown diagram

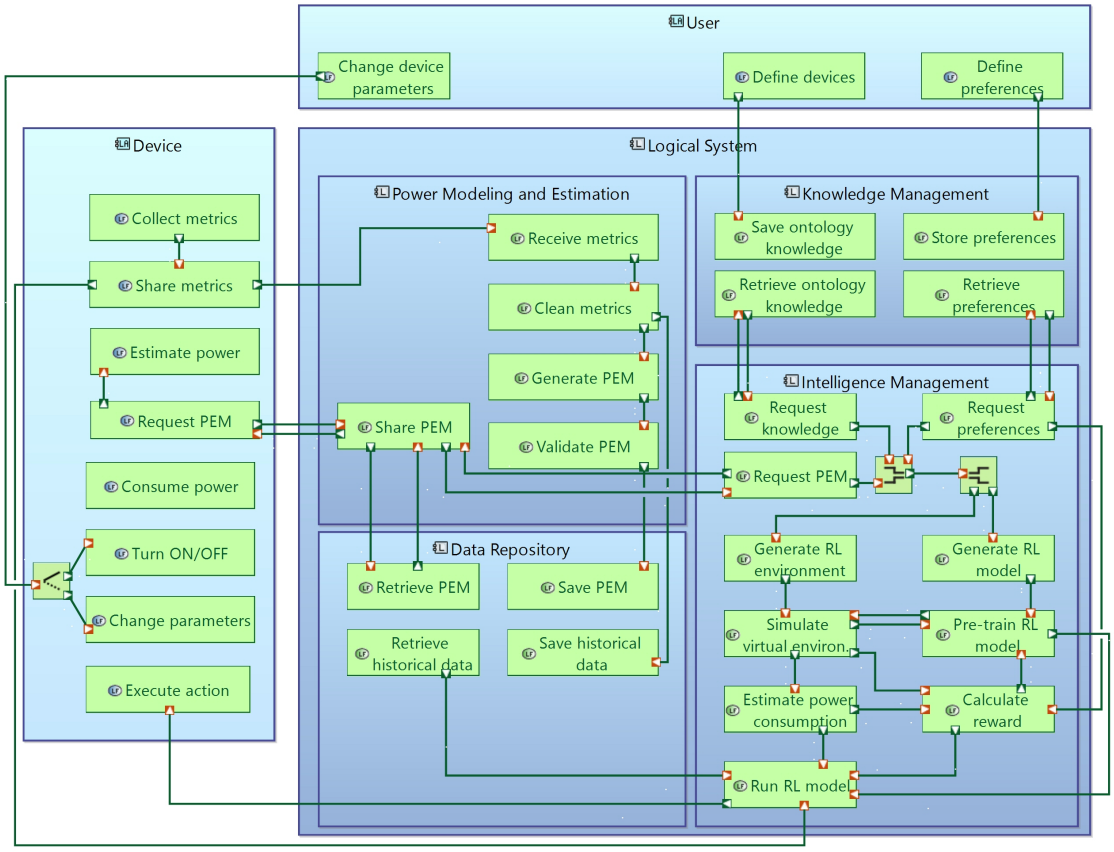


Figure 3.11: Logical architecture diagram

Figure 3.10 shows the logical component breakdown diagram. The four main components that constitute the system are defined as follows:

- A power modeling and estimation component aims to empirically generate power models for a large number of devices and configurations.
- A knowledge management component aims to provide a common representation and definition of concepts and the relations between them. In addition to the focus on power consumption while representing the knowledge.

- An intelligence management component aims to collect intelligence from the knowledge management component and create a virtual environment and a reinforcement learning agent based on the devices present in the environment and the relations between them. The virtual environment is used to train the agent to accomplish power optimizing action.
- A data repository component is formed by a collection of databases that include previously collected metrics, states, applied actions, rewards, power consumption, power estimation models, and potential external data.

Figure 3.11 shows the final logical architecture diagram that describes the system and its components. Each of the previously defined functionalities is divided into functions and allocated to their respective components. Exchanges between each of these functions is also identified.

3.7 Implementation Choices

Different development choices exist to conceive a physical implementation of our solution. Decisions are highly influenced by the requirements, needs, and goals of the system defined above. In this section, we propose the use of an ontology for knowledge representation as well as reinforcement learning algorithms for intelligence power management.

3.7.1 Knowledge Management Implementation Choices

We decided to use an ontology aiming to implement the knowledge representation of the environment. Ontologies are considered very well-adapted for knowledge modeling and information retrieval. They provide a common representation and definition of the concepts and the relationship between them. In addition, ontologies support inference to discover new relationships and knowledge. They are scalable, reusable, and extensible. While a database is used for the data repository because it is adequate and effective for storing large data sets (historical data and power estimation models).

Firstly, it would be interesting to define each of the following two concepts: a database and an ontology. On the one hand, an ontology is a defined set of concepts and relationships used to represent the meanings and a shared understanding in a certain domain [100]. On the other hand, a database is an organized collection of structured information defined by a database schema. It is used to store data without concerns about the meaning of this data [101]. Therefore, ontologies and databases have different purposes. The former aims to represent and define concepts and the relations between them. The latter goal is the

storage of data without the interest of their meaning. A database does not use taxonomy, which nevertheless constitutes the backbone for an ontology [102]. The differences between a database and an ontology have been explored in prior studies by [103, 104, 105]. A summary of these differences is outlined in Table 3.1.

Table 3.1: Differences between a database and an ontology

Criteria	Database	Ontology
World Assumption	Close	Open
Name Assumption	Unique	Not Unique
Implicit Information	Unavailable	Available
Represents	Data	Knowledge
Scalability	Low	High
Need of complete information	Yes	No
Focus on semantics	Minimal	Strong
Starting point	Scratch	Reuse is possible

Secondly, we justify our choice of using an ontology as our knowledge management component with the following arguments and advantages [102, 100, 106]:

- Allows the definition of a semantic model of the data combined with associated domain knowledge and defines links between different types of semantic knowledge. Therefore, advanced search strategies can be formulated.
- Shares common understanding of the information structure between heterogeneous components of the system allowing knowledge sharing. The resolution and prevention of communication issues between heterogeneous systems are guaranteed.
- Analyze the domain knowledge by using inference to discover deduced answers such as new relationships and new knowledge. Reasoners are also used to validate the logic and the consistency of the defined concepts and to endorse their structure.
- Facilitates information interoperability and integration between different knowledge and information sources while preserving a high level of abstraction and information fusion.
- Enables the reuse of the domain knowledge in different applications because of the separation between the domain knowledge and the operational knowledge.
- The world that we need to represent is significantly large and complex, in addition, it is impossible to assume the information about the domain is complete.

3.7.2 Machine Learning Implementation Choices

An intelligent system is a machine that incorporates a computer connected to its environment; that can collect and analyze data and communicate with other systems. It can analyze data using four different types of analytics, including descriptive, diagnostic, predictive, and prescriptive [107]. Furthermore, machine learning can provide cognitive capabilities to understand such large amounts of data [108]. There are three different machine learning techniques: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning adapted to problems where each input is labeled or belongs to a category (e.g., classification and regression). Unsupervised learning is useful for problems where each data is not labeled and does not belong to a category. This technique is good for grouping complex data into classes (e.g., clustering) [109]. Reinforcement learning is useful when future actions are based on the outcome of the current responses of the environment. A summary of the differences between machine learning techniques is outlined in Table 3.2.

Table 3.2: Differences between machine learning techniques

Criteria	Supervised	Unsupervised	Reinforcement
Aim	Predict outcome	Discover patterns	Learn from actions
Type of data	Labeled	Unlabeled	No predefined data
Supervision	Yes	No	No
Feedback	Yes	No	Yes
Flexibility	Low	Medium	High
Learning Type	Passive	Passive	Active

The use of supervised learning techniques for power modeling and estimation has many advantages: Power is a measurable value that is usually directly linked to other metrics values. Therefore, collecting and labeling this data (with the real-time power consumption) allows the easy creation of power estimation models using supervised learning algorithms.

The use of reinforcement learning for power management has many advantages: It does not require large labeled datasets, therefore, providing an advantage in terms of scalability and flexibility. It is adaptable, in contrast with supervised learning algorithms, RL can adapt to changes in the environments automatically. It is goal-oriented by defining the policies such as energy optimization, user comfort, etc. The agent will then try to find the sequence of actions based on collected metrics to satisfy the required objectives, in our case energy optimization and users satisfaction.

Based on the previous comparison, we chose to use supervised learning regression algorithms to generate power estimation models and reinforcement learning techniques for

power management functionality for the remainder of this thesis.

3.8 Summary

In this chapter, we present the design of our automated power management system for connected environments and CPS. We follow a MBSE methodology to specify the functional and non-functional requirements and to design structural and behavioral dimensions of our solution. We start by specifying and analysing the needs of actors/entities. Furthermore, we present the functionalities of systems and how to fulfill these needs. Finally, we identified components where each function will be developed.

We designed different architectures based on the layers defined in the ARCADIA methodology. An operational, system, and logical analysis are accomplished using the Capella open-source tool. Throughout these different modeling phases, we took architectural and development decisions impacting the implementation of the approach. The two identified system capabilities are: (1) model and estimate power consumption and (2) manage power consumption.

The four main components of the system are defined, as follows:

- A power modeling and estimation component responsible for creating power estimation models for a variety of devices.
- A knowledge management component aims to represent knowledge in extensible, reusable, and scalable ways.
- An intelligence management component aims to create reinforcement learning environments and models based on the devices present in the environment. It takes energy-oriented actions to manage these devices while respecting user comfort.
- A data repository component aims to save historical data and models for future use and analysis.

In the next chapter, we fulfill the first system capability that consists of modeling and estimation of power consumption for different devices by implementing one component of the modeled solution. It will be supported by scientific and experimental results for validation.

Chapter 4

Automated Power Estimation of Heterogeneous Devices

Contents

4.1 Introduction

With the explosion of smart and connected devices, there is a need to measure and monitor the power consumption of these devices. The impact of all these devices on ICT power consumption and carbon footprint is undeniably rising, therefore, need to be measured . There is a rise in popularity and usage of alternative architectures on connected devices that have a huge variety of hardware and software configurations (*i.e.*, ARM/RISC due to its lower power consumption than x86/CISC). External power meters can provide accurate power measurements for specific workloads and environments (such as in [110]). However,

these meters are costly financially, scale poorly for a large park of devices, have a time-consuming setup, and require physical access to each device. Therefore, it is important to provide software-based power models for these devices. However, without embedded power sensors or constructors' power models and API, it is challenging to provide accurate power models for this variety of device configurations. In addition, monitoring the power consumption at run-time (in addition to other performance metrics) helps software developers to detect misbehaving software, or specific power drains due to a particular hardware configuration.

Current power estimation techniques are either based on mathematical formulas (such as in [111]), or on a static data set used to generate an empirical model (such as in [112]). In addition, such models target a single device release with no way to estimate other revisions or variations of the device without manually conducting the experiments again.

Our main motivation is to provide an automated approach to model the power consumption of various devices, while allowing the models to be updated, extended, improved, and shared. We argue that such an approach leads to democratizing power comprehension of hardware and software in different devices and environments.

Providing an accurate and automated approach to solve these questions is challenging, and in particular regrading the heterogeneity of the environment, the empirical validity, and the automated power modeling. The detailed specification of the functional and non-functional requirements have already been detailed in the previous chapter.

In this chapter, we present an automated approach and architecture to empirically generate power models for, potentially, unlimited devices and configurations. Our approach provides always up-to-date and accurate power models with low error rates. The approach follows a crowd-sourcing architecture where benchmarking components can run on any device, generate empirical data, and lastly, our power model generator component will generate an accurate power model for the specific device, or improve the model if a previous one already exists. Monitoring software can connect to our architecture to query and retrieve the most accurate and up-to-date power model for their devices. The data collection and model generation phases are relatively fast as they only take a few minutes. Once the model is generated, power consumption can be estimated with negligible overhead in real-time. With time, the accuracy of the model can be improved as more data are collected and fed to generate more accurate regression models. This architecture follows the logical architecture presented in chapter 3 and represents its implementation of the power modeling and estimation component on the physical level.

The main contributions of this chapter can be summarized as follows: (i) energy estimation models are always up-to-date due to the continuous data benchmarking as new models are automatically generated when necessary, (ii) model generation is done in an

automated manner from the data collection to the testing (with human intervention needed only to start the benchmarking process, and (iii) our proposed approach is a collaborative one where benchmarks can also be crowd-sourced, and energy estimation models are shared across users and devices.

We provide a proof-of-concept implementation for automated power modeling of the entire current set of Raspberry Pi devices. A comprehensive experiment validates our approach, implementation, and power models. We generate various and accurate power models, using two regression models, with very low error rates as low as 0.3%. Our models are vastly more accurate than existing models with as much as 10 times lower error rates. We also discuss and analyze the impact of multiple hardware and operating system configurations, and discuss a use-case scenario in remote power monitoring.

4.2 Related Work of Power Measurement and Estimation

Several approaches and tools have been proposed to measure or estimate the power consumption of computing devices. Some focus on hardware meters while others use software-based approaches. In this section, we review the related approaches to monitoring and estimating power in computers and IoT devices, and discuss the relevant work around power and regression benchmarks.

In [113], the authors studied the energy impact of users' operations in Raspberry Pi compared to other computing devices. Measurements were carried out using hardware meters due to the lack of accurate software approaches. In [114], a method for energy estimation of Zolertia RE-Mote devices was proposed. It combines offline profiling with online energy estimation, using both a software-based mode and a hardware-based one. The former uses theoretical energy for each operating state, taken from the datasheet, and captures the time spent in each state, and provide energy estimations with an error margin of 53% The latter uses an integrated circuit to measure accurately the power consumption of each state in real-time. A hybrid (hardware and software) power measurement platform for wireless IoT devices called EMPIOT was proposed in [115]. It mainly targets the issue of the power consumption measurement for peripherals. It studied the impact of various design parameters on precision and overhead. It was evaluated by running sleep, encryption, and communication workloads on five different computing devices. SMARTWATTS [116] is a power monitoring platform that increases the accuracy of its CPU and DRAM power models by using an online calibration technique for containers. The authors argue that this approach can be implemented on various machines because it does not need any training phase or pre-configuration. Since 2014, many systems use the Running Average Power Limit (RAPL) feature for power consumption measurements. This feature became

available on most Intel CPUs. Yet many devices having ARM or AMD processors do not support RAPL.

In [117], a micro-benchmark-based modeling approach for heterogeneous processors was proposed, in which the authors state that statistical modeling has a significant initial cost during the model training. In [118], the authors propose a technique to generate CPU power models without having a profound knowledge of the CPU architecture. It automatically detects the hardware performance counters correlated to the power consumption and generates the power model from the selected features. Their approach supports changing the learning approach according to the need and software-defined power metering. In [119], a power prediction by applying linear regression to on-chip performance monitoring counters, in particular for the number of micro-operations that are fetched, completed, and retired in each monitoring cycle. In [120], a piece-wise linear function was proposed to estimate the power consumption of an AMD processor. The result of this approach showed a better fit to the collected data when compared with linear and exponential functions. The micro-benchmark data was collected by stressing four AMD performance counters. In [121], the authors reviewed direct measurement and estimation models. Most of the reviewed models apply linear regression techniques on hardware performance counters. They showed that previous studies were limited regarding the considered workloads and the impact of the complexity for each model. In [122], the authors proposed an approach based on statistical power modeling by applying regression analysis on high-level activity metrics. For example, they collected the time spent in each state and the occurrence of certain events and focused mainly on the interaction between the processor and the memory. In [123], the authors surveyed the literature approaches used for energy consumption modeling and prediction for data centers. They noticed a linear relationship between power consumption and CPU utilization.

PowerPi [124] and EMM [125] proposed power models for a particular Raspberry Pi version (RPi 2B and RPi 3B+, respectively). In both approaches, they empirically created a data set correlating power consumption from a power meter to CPU utilization. A linear regression analysis was then applied to generate a power model. However, two main limitations of their approach: first, the model targets only one particular device version and cannot be used on other models due to the distinct power consumption of each RPi model [126]. And second, improving the accuracy of the regression requires manually conducting the experiments again with additional data. In [127], the authors proposed five different power modeling techniques by correlating software metrics with physical power measurements using Mantis [128]. The latter generates a power model using a one-time model fitting technique by collecting software metrics and correlating them to the measured power. It collects metrics from the main system components, namely CPU, memory, and

disk. In the paper [127], the authors found that the CPU is the major energy consumer in a computer, but the relation between power and utilization is not always linear.

In [129], the authors proposed a deep neural network (DNN) that predicts the remaining battery of IoT devices. This approach is based on pre-processing the data (eliminate missing values, convert them to numerical format, and normalize data). Then, a Moth Flame Optimization is used to select the optimal features that are input for a DNN model. It provided advantages regarding the feature selection and the battery life estimation accuracy. However, overload of the approach was not calculated and tends to be high due to the significant processing needed. In [130], the authors have gone beyond only predicting battery life using machine learning models to propose a DNN alongside a blockchain. The use of blockchain, as a secure and trustworthy prediction storage, improved the authenticity of the prediction from a security point of view. The DNN predicted the remaining battery with an average accuracy of 90%.

Table 4.1: Summary of related work.

Paper	Suitability	Error (%)	Description
[113]	Any	N/A	Hardware solution
[114]	IoT	4-18	Hardware/Software
[115]	IoT	3.5	Hardware/Software
[116]	Containers	N/A	RAPL
[118]	CPU	1.5	Performance counters
[120]	CPU	3-7	Performance counters
[127]	CPU	10	Performance counters
[117]	AMD APU	3-7	Regression
[119]	CPU	2.6	Regression
[122]	ARM SoC	5	Regression
[129, 130]	IoT	5.17	DNN
Most comparable related work			
[124]	RPi 2B	14.56	Regression
[125]	RPi 3B+	40.76	Regression

The related work is summarized in Table 4.1. Although some results appear consistent with prior research, the existing research has multiple limitations: i) Proposed energy estimation models are quickly out-of-date due to software or kernel updates, or new revisions. Such updates are becoming more frequent in modern software, therefore reducing the efficiency of a model generated from a particular software version or hardware revision, ii) The process of model generation is not fully automated and requires human intervention for different tasks (running the benchmark, collecting data, generating, and validating

the energy model), and iii) Estimation models are not easily and automatically shared in an effective way between different devices and users. Users need to manually acquire the appropriate energy model for their devices.

To the best of our knowledge, none of the power estimation model generation approaches is based on a continuous improvement method or proposed a technique to automate model generation on a large scale.

4.3 Automated Power Modeling Architecture

In this section, we present our automated power modeling architecture where we describe the physical architecture of both our client and server components, along with the automatic generation of power models.

The architecture aims to generate an always up-to-date and accurate power model for various computing devices, such as servers, PCs, single-board computers, or embedded and IoT devices. We achieve this automatic generation with a multi-component architecture aimed to collect and process power data and metrics, apply machine learning algorithms, and generate an updated, more accurate, power model.

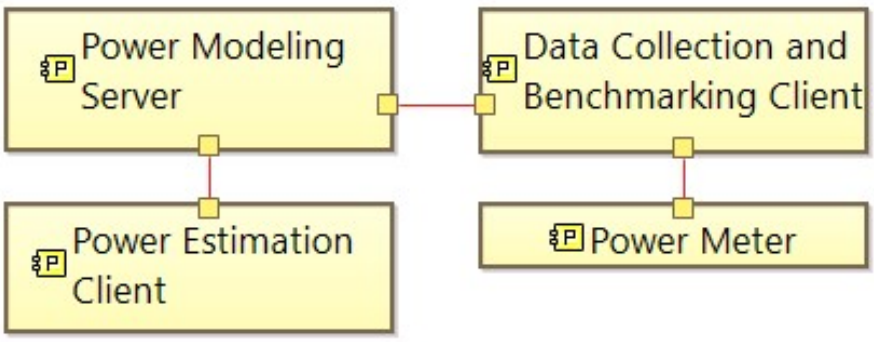


Figure 4.1: An overview of our automated power modeling physical architecture

Concretely, our architecture, in Figure 4.1, is composed of three distinct but complementary components: a data collection and benchmarking client, a machine learning and power modeling server, and a power estimator client. Simply put, our benchmarking client will collect run-time software, hardware, and power metrics, which are then sent to the power modeling server. The latter will then generate empirical power models based on the current and previously collected metrics, using machine learning techniques. Finally, the power estimator client will query the server for the most up-to-date and accurate model for the device it's running on, and use that model to estimate the power consumption of the device.

Each component of our architecture can be independently implemented as we aim to provide a decentralized and decoupled architecture. For instance, our generated power models can be used by third-party power estimator clients to provide run-time and live power estimations, or used by hardware management software to supervise the power consumption of a set of devices.

In addition, the architecture is designed to maintain a high level of flexibility to manage: new devices introduced to the environment, changes occurring to the existing ones (such as OS or kernel updates), new metrics to consider and collect, or changing the model generation algorithm.

4.3.1 Data Collection and Benchmarking Client Architecture

Figure 4.2 presents the physical architecture of our data collection and benchmarking client. Its main role is to collect software, hardware, and power metrics of run-time and real-world workloads. The collected data will then be used as a training and validation data set for the machine learning algorithms in our server.

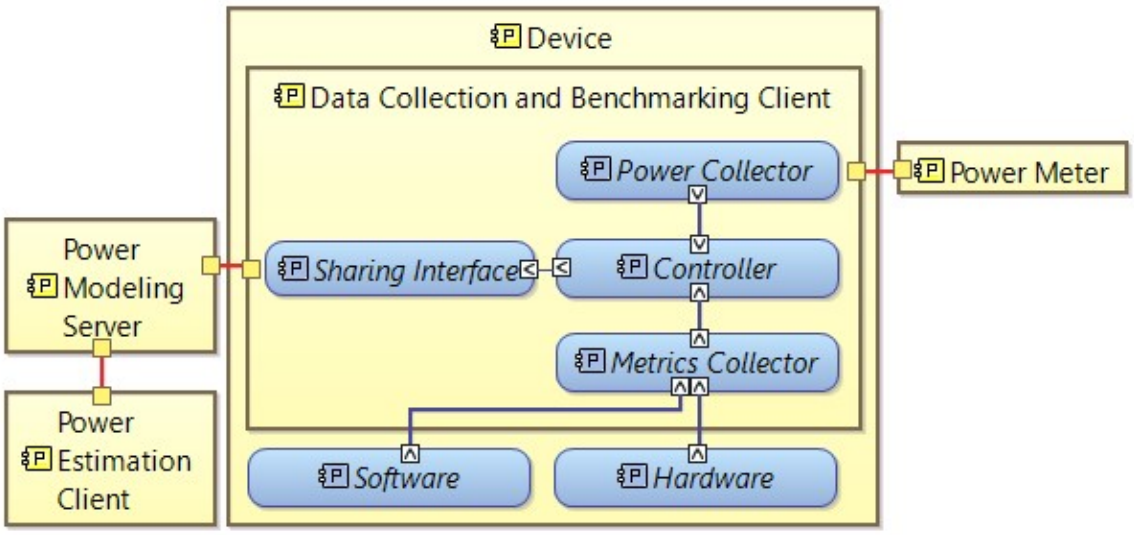


Figure 4.2: Benchmarking client physical architecture

Our benchmarking client first needs to collect the power consumption of the workload. This first step requires an accurate power measurement component, as this metric will be used as the *truth* for this power metric. Therefore, we recommend using a physical power device, such as an external power meter, or an integrated physical power sensor.

The main components of our architecture are the following:

Power Collector : this component connects to the power measurement sensor and collects run-time power metrics (such as the power consumption in watts, the current, the voltage, or any other power-related metrics), and associates each measure to a timestamp.

Metrics Collector : this component collects various metrics from the operating system, hardware (through the OS), and software. For example, it can collect the number of CPU cycles, the transmitted data packets in a network, the number of storage access requests, or more complex metrics and data (such as software running, network throughput, quality of service, or metadata about the operating system or the software workload).

Controller : this component orchestrates the data collection from the energy and metrics collectors. It controls the frequency of data collection and sharing with the server. It also makes sure that the metric collector is running simultaneously with the power collector.

Sharing Interface : this component's role is to share the collected data to the power modeling server. It can be implemented as a web service API, or through a file-sharing mechanism (on a local network, over FTP, etc.), or any other sharing method understood by the server.

4.3.2 Power Modeling Server Architecture

Figure 4.3 presents our power modeling server physical architecture. Its main role is to receive generate accurate and always up-to-date power models using the data collected from the benchmarking client. The server acts as a centralized entity towards multiple benchmarking clients, receiving data from multiple similar or different devices.

For example, benchmarking clients can collect data from workloads running on multiple instances of a same-type device. This data collection can also spread across time, and the server will regenerate a new updated power model each time a new data set of metrics is received from a benchmarking client.

The main components of our architecture are the following:

Collector Interface : this component receives the data collected by the benchmarking clients. At this point, the data is received as is, and further processing is handled in the next components.

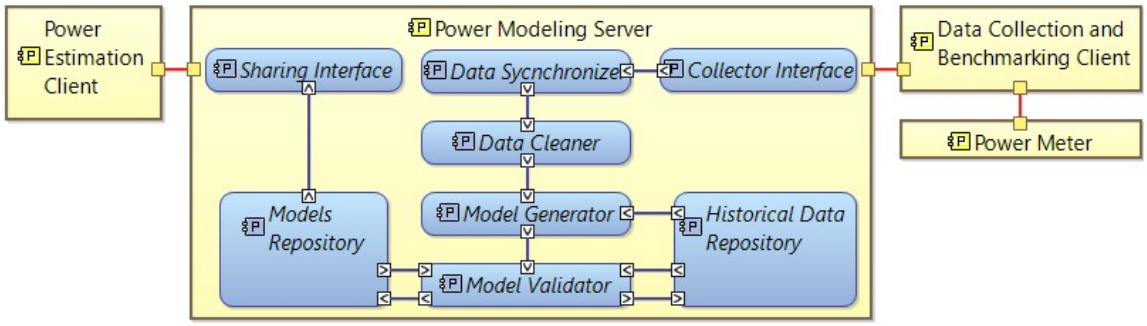


Figure 4.3: Power modeling server physical architecture

Data Synchronizer : this component processes the received data (which might be in multiple formats or files), synchronize timestamps between the metric and power data, verifies and synchronizes clock diversion between the timestamp of the computing device and the one from the physical power sensors or meters.

Data Cleaner : this component cleans the synchronized data by identifying and eliminating redundant, erroneous, or out-of-context data. For example, the cleaner tries to identify when the useful workload started and ended, and discards data points outside this range. The cleaner also verifies that the received data is valid, such as if the provided power values are within the power range of the device it was run on, or whether bogus or spamming data are present.

Model Generator : this component generates a power estimation model based on empirical machine learning techniques, using the newly received data along with the data already stored in the server about the particular device. For example, the estimation models can be based on any prediction algorithm (such as regressions, decision trees, neural networks, etc.).

Model Validator : this component validates that the new model is more accurate than the current model stored in the server for the device. For example, using all available data sets (including the newly received ones), it can compare the average error of the new model to the currently stored one, and then keep the more accurate one.

Historical Data Repository : this database contains all the collected data of all devices, benchmarks, metrics, and workloads. The newly received, cleaned and validated data are added to this repository, and therefore contributing to building big data set associating various metrics and data to power consumption, and gradually improving the accuracy of our model generator.

Power Models Repository : this component contains all the generated power models for all devices. Only the most accurate power model is saved to this repository, along with the model and estimation parameters, and the average error of the model. The repository distinguishes between devices, but also between revisions of devices (for example, Raspberry Pi 4B revision 1 and revision 2), between operating systems architectures (32 or 64 bits), etc.

Sharing Interface : this component provides a sharing mechanism for power estimator clients to retrieve the latest up-to-date power model of their device. As with our other sharing interfaces, it might be implemented as a web service API, a format-specific file, or any other sharing mechanism and format.

4.4 Implementation for Raspberry Pi Power Models

In this section, we present a specific implementation of our architecture aimed to generate accurate power models for single-board computers in general and Raspberry Pi devices in particular. To the best of our knowledge, our implementation is the first to provide a comprehensive set of benchmarks and power models for the entire current set of Raspberry Pi devices.

4.4.1 Data Collection and Benchmarking Client

Our implementation of the benchmarking client consists of two main components: a workload generating benchmark and a data collector. Our architecture supports multiple types of data collections and can generate power models for multiple hardware components. However, in our proof-of-concept implementation, we focus on generating an accurate power model for the ARM processor of Raspberry Pi devices by collecting CPU metrics.

The workload generated by the benchmark consists of applying variable loads on the device's processor. We decide to apply a stress load on the CPU covering the entirety of the load range, i.e., we stress the CPU from 0% all up to 100%, with an incremental step of 5%. The load is applied for 60 seconds for each percentage step. We also saved the workload timestamp and store everything in a CSV file.

The data collector component is a program collecting CPU metrics. In particular, we collect CPU cycles from the Linux *proc* interface (`/proc/stat`). Then we calculate the CPU utilization (ranging from 0 to 1) and save this data to another CSV file.

We calculate the CPU utilization by calculating the ratio of the busy cycles (CPU

cycles in user and kernel mode) with the total cycles which includes idle ones:

$$u[t] = \frac{c_{busy}[t] - c_{busy}[t-1]}{c_{total}[t] - c_{total}[t-1]} \tag{4.1}$$

where:

- $c_{busy}[t]$ is the total number of busy cycles up to time t (busy is here equal to: user + nice + system from */proc/stat*).
- and $c_{total}[t]$ is the sum of $c_{busy}[t]$ and the number of idle cycles $c_{idle}[t]$.

In addition, we collect the actual power usage using a power meter and store the power data in a third separate CSV file. The power data is collected from another device to reduce the impact on the workload and accuracy of the benchmark. The controller makes sure the collection of metrics and power is done with the same frequency and at the same time. It gathers the three CSV files and prepares them to be shared with the power modeling server.

4.4.2 Power Modeling Server

We build our power modeling server following a decision algorithm presented in Figure 4.4.

We first collect the three generated CSV files from the client (in our implementation, sharing the CSV file from a common storage location). We then normalize and clean the data:

- Remove irrelevant data points (the ones from before, after, and separating the workloads),
- Synchronize timestamps of the three CSV files,
- Synchronize the clock diversion between the Raspberry Pi timestamp and the power meter,
- Aggregate the three CSV files into one containing the label power, the CPU utilization, and the timestamp,
- Remove the inconsistency in the data that results in anomalies (peaks and troughs present at the beginning and end of workloads).

We then proceed with the generation and validation of our power model. First, we check if an estimation model already exists on the server for the specific Raspberry Pi device and version. If no model exists, then we proceed to generate a new power model ($M'(n)$) using the collected data, calculate its average error $E'(n)$. Furthermore, we save

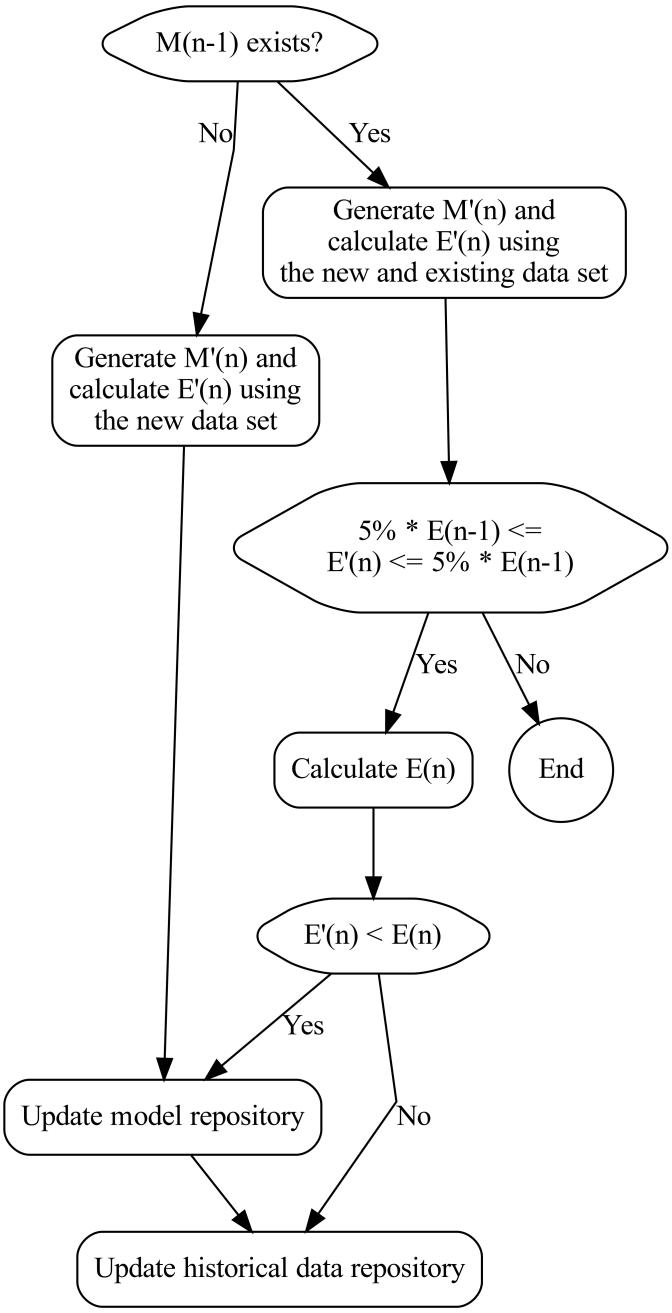


Figure 4.4: Our implementation approach for power modeling

the new power model to the model repository and the collected data to the historical data repository.

However, if a current power model exists ($M(n-1)$ along with its average error $E(n-1)$), then we proceed with the following process:

- We first read all the data saved in the historical data repository of the specific device,

and temporally add the newly collected data to form a new data set. This data set is then used to generate a new power model $M'(n)$, and the average error for this new model is also calculated $E'(n)$.

- If the error rate of this new model $E'(n)$ is outside of an accepted predefined range compared to the previous model $E(n-1)$, then we discard the collected data (as we consider it is not valid), and the server keeps its data and power model. In our implementation, we consider a 5% range around the error rate as a good indicator of whether the data is valid or has been trafficked. The latter can happen if the power data of a device has been mixed with the collected metrics of another, or fake data has been sent, or an error in converting numbers happens in the client.
- If the error rate is within the predefined range, then we calculate the error rate $E(n)$ of the existing model $M(n-1)$ using all the data (historical and new ones). We do this additional calculation because the current error rate $E(n-1)$ has been calculated using the historical data only.
- We then compare the new error rate $E(n)$ of the current model $M(n-1)$, with the error rate $E'(n)$ of the new model $M'(n)$. The model with the lowest error rate will then be stored in the models repository as the new up-to-date and accurate power model of the specific device. And lastly, we store the newly collected data in the historical data repository.

At the end of this process, the server will contain additional data points which will, over time, improve the accuracy of our empirical power model generation.

In our implementation, we use linear and polynomial regression algorithms to generate power models, as a correlation was observed between CPU utilization and power consumption in Raspberry Pi devices.

In the next section, we detail the empirical experimentation to validate our implementation and power models.

4.5 Empirical Validation and Discussions

In this section, we detail the empirical experimentations that validate our approach, implementation, and generated power models.

4.5.1 Experimental Setup

Our experimental setup consists of 8 Raspberry Pi devices from different generations and revisions as detailed in Table 4.2, dating back from 2012 until the latest current model.

We run our workload on both 32 bits (armv7l) and 64 bits (aarch64) operating systems for model 4B (for each of the 2 revisions we used).

We used the same SD card to boot Raspberry OS on all devices, switching operating systems and ARM architecture accordingly. We also automated the benchmark experimentation by adding a boot script to `/etc/rc.local`.

Table 4.2: The variety of Raspberry Pis used during experiments

Model	Rev.	OS	CPU Architecture	Cores	Released
Zero W	1.1	32	armv6l	1	2017
1B	2	32	armv6l	1	2012
1B+	1.2	32	armv6l	1	2014
2B	1.1	32	armv7l	4	2015
3B	1.2	32	armv7l	4	2016
3B+	1.3	32	armv7l	4	2018
4B	1.1	32/64	armv7l/aarch64	4	2019
4B	1.2	32/64	armv7l/aarch64	4	2019

To collect power consumption, we use the PowerSpy2 power meter [1]. PowerSpy2 is a Bluetooth power meter used for advanced and accurate analysis. To reduce interference in the experiments, we use a separate computer to connect to the meter and collect power metrics.

We run all our experiments on Raspberry Pi OS (version based on Debian 9 stretch), with Linux kernel 4.14. Our components and tools are written in Python and run with version 2.8, and in C compiled with GCC 6.3.0. For Raspberry Pi 4B, we the supported version of the OS based on Debian 10 buster with Linux Kernel 5.4, and GCC 8.3.0.

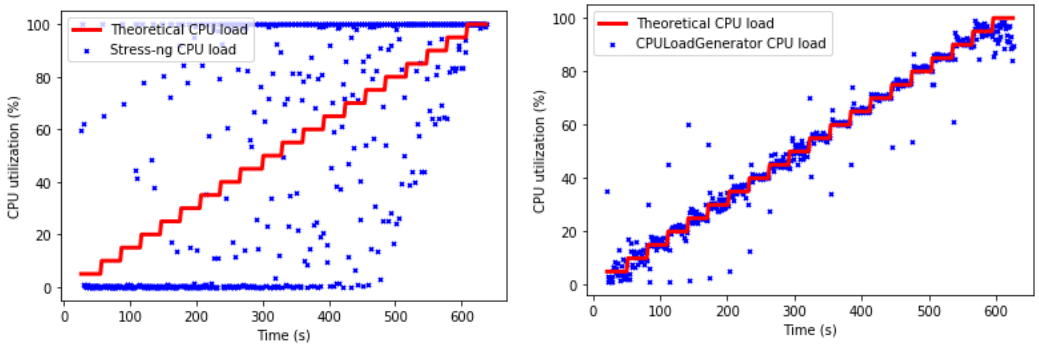
To further reduce interference on the accuracy of our experiments, and as we aim to generate a CPU power model, we disconnected all external peripherals during the workload (including the monitor through HDMI, keyboard, and mouse through USB ports and the network through the Ethernet interface). We also disabled from the operating system all network interface cards (i.e., WiFi and Bluetooth). We also limited the running applications to the minimum as to only monitor the power impact of the workload. Furthermore, we made sure that every device was cooled down before running the experiments, as overheating can have an impact on power consumption. Specifically, each Raspberry Pi was disconnected from its power supply until the device was cooled down.

[1] https://www.alciom.com/en/our-trades/products/powerspy2/

4.5.2 Benchmark Data Collection

To collect our CPU metrics, we wrote a minimal C program that read the `/proc/stat` file every second and calculated the CPU utilization. The latter was then saved into a CSV file.

As described in our implementation in Section 4.4.1, we stressed the CPU from 0% to 100% with a 5% increment.



(a) (b)

Figure 4.5: CPU theoretical load compared to the load generated by (a) the stress-ng command and (b) the CPULoadGenerator

We initially used the same stress command (or stress-ng) used in the literature [125] to specify a percentage CPU load. However, we noticed that the CPU load was inconsistent, with the actual CPU load altering between 0% or 100% in various time duration, rather than consistently stabilizing at the asked percentage load. Instead, we used a Python script, CPULoadGenerator [2], which consistently stressed the CPU at the asked percentage with a small degree of variation. Figure 4.5 outlines the differences in CPU load consistency between the two tools.

For each experiment, we stressed the CPU for 60 seconds for each CPU load step and is followed by a 10-second pause. A 60-second pause precedes each experiment in order to reduce the impact of our script on the results. In total, each experimental benchmark runs on average for about 25 minutes (24 min and 20 sec).

Each benchmark generates a total of 1460 data points. After the cleaning phase, we end up with more than a thousand data points. These are then used to generate our empirical power models. For the purpose of our experiments, we run our benchmark a few times for the Raspberry Pi 3B+ and Raspberry Pi 4B rev 1.2 (64 bits) and ended up with around 5400 data points for the former, and around 2000 data points for the latter. In total, RPi Zero has 666 data points, 1B has 1034, 1B+ has 954, 2B has 1137, 3B has 1105, 3B+

[2] https://github.com/GaetanoCarlucci/CPULoadGenerator

has 5383, 4B 1.1 (32 bits) has 1089 and its 64 bits version has 998, 4B 1.2 (32 bits) has 1017 and its 64 bits has 2014 data points. The difference in the number of data points for each device type is due to our additional experiments (in particular for the 3B+ and 4B, for instance, to compare 32 bits vs 64 bits, or the validation of the power model generator). Additionally, our cleaning script strips the waiting time between each experiment run of the stress benchmark. As the stress command is not precisely perfect in its timing, the duration of each test might vary by a few seconds, hence the additional data points.

4.5.3 Regression Power Models

For our experiment, we used two regression algorithms in our server to generate the power models: linear and polynomial regression. We choose these two regression algorithms because of the visible correlation between power and CPU utilization. However, other machine learning techniques and algorithms can be applied to generate different regression models.

Table 4.3: RPi generated power models using polynomial regression algorithms

Raspberry Pi	y-intercept	Degree								
		1	2	3	4	5	6	7	8	9
RPi Zero W Rev 1.1	0.85	7.21	-135.52	1254.81	-6329.45	18502.37	-32098.03	32554.68	-17824.35	4069.18
RPi 1B Rev 2	2.826	3.54	-43.59	282.49	-1074.12	2537.68	-3761.78	3391.05	-1692.84	357.80
RPi 1B+ Rev 1.2	1.251	1.86	-18.11	101.53	-346.39	749.56	-1028.80	863.88	-403.27	79.93
RPi 2B Rev 1.1	1.36	5.14	-103.3	1027.17	-5323.64	15592.04	-26675.60	26412.96	-14023.47	3089.79
RPi 3B Rev 1.2	1.52	10.05	-234.19	2516.32	-13733.56	41739.92	-73342.79	74062.65	-39909.43	8894.11
RPi 3B+ Rev 1.3	2.48	2.93	-150.40	2278.69	-15008.56	51537.32	-98756.89	106478.93	-60432.91	14053.68
RPi 4B Rev 1.1	2.57	2.79	-58.95	838.88	-5371.43	18168.84	-34369.58	36585.68	-20501.31	4708.33
RPi 4B Rev 1.1 (64 bits)	3.41	-11.83	137.31	-775.89	2563.40	-4783.02	4974.96	-2691.92	590.36	
RPi 4B Rev 1.2	2.59	12.34	-248.01	2379.83	-11962.42	34444.27	-58455.27	57698.69	-30618.56	6752.27
RPi 4B Rev 1.1 (64 bits)	3.41	-3.07	47.75	-271.97	879.97	-1437.47	1133.33	-345.13		

Our power modeling server is implemented in Python scripts, automating the data cleaning, synchronization, power modeling, and storing data into the repositories. We, then, generated power models for all different Raspberry Pi models.

Table 4.4 outlines the generated power models using linear regression algorithms, and Table 4.3 outlines the models generated with polynomial regression models.

4.5.4 Validation of Power Models

To validate the accuracy of our power models, we compare the power consumption calculated by our models to the power consumption measured from the power meter. This allows us to calculate the absolute difference between these two values for every data point in the collected metrics. We then calculate an average error using all measurements from all devices with different regression models.

Table 4.4: RPi generated power models using linear regression algorithms

Raspberry Pi	Estimation Model
RPi Zero W Rev 1.1	$P = 0.4733 \times U + 0.9201$
RPi 1B Rev 2	$P = 0.1424 \times U + 2.9117$
RPi 1B+ Rev 1.2	$P = 0.1220 \times U + 1.3143$
RPi 2B Rev 1.1	$P = 1.1488 \times U + 1.2903$
RPi 3B Rev 1.2	$P = 3.4774 \times U + 1.0782$
RPi 3B+ Rev 1.3	$P = 3.2983 \times U + 2.0022$
RPi 4B Rev 1.1	$P = 3.7121 \times U + 2.2058$
RPi 4B Rev 1.1 (64 bits)	$P = 4.4958 \times U + 2.3073$
RPi 4B Rev 1.2	$P = 3.4842 \times U + 2.2434$
RPi 4B Rev 1.2 (64 bits)	$P = 4.5344 \times U + 2.2857$

Figure 4.7 presents the measured correlation between CPU load (in percentage), and the power consumption (in watts) for our two regression models (linear and polynomial) and the actual measurements from the power meter. Except for Raspberry Pi 1B and 1B+, our empirical benchmarks show a better fit for the polynomial regression.

This translates into a lower average error for the polynomial model as compared to the linear model for all experiments and Raspberry Pi devices, as seen in Figure 4.6. The average error for the linear models varies from as low as 0.34% for RPi 1B+, to 7.81% for RPi 3B. In contrast, the highest average error for the polynomial model is 3.83% to the RPi 3B+.

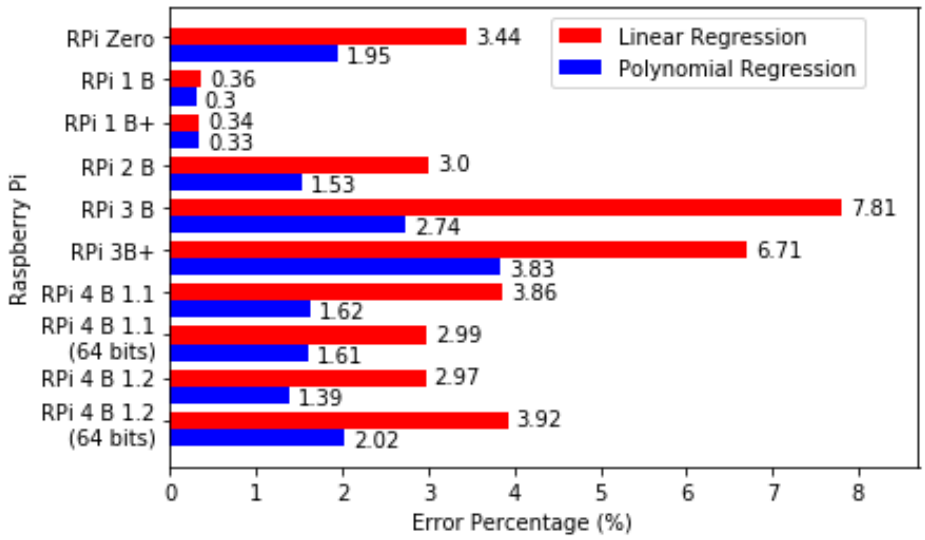


Figure 4.6: Average Error percentage for linear and polynomial regression algorithms per Raspberry Pi

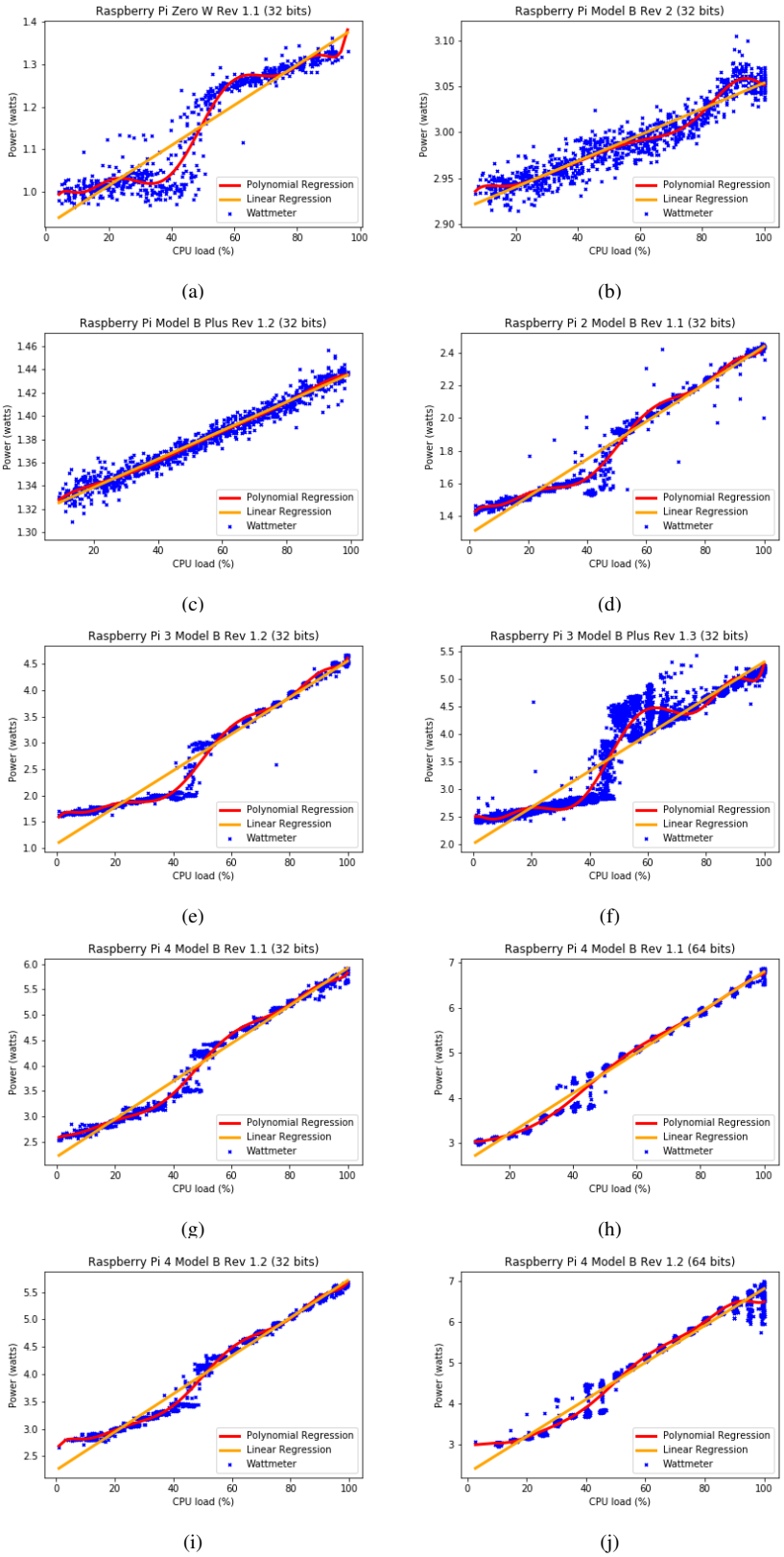


Figure 4.7: Linear vs polynomial regression power estimation models of the (a) RPi Zero W (b) RPi 1B (c) RPi 1B+ (d) RPi 2B (e) RPi 3B (f) RPi 3B+ (g) RPi 4B 1.1 (h) RPi 4B 1.1 (64 bits) (i) RPi 4B 1.2 (j) RPi 4B 1.2 (64 bits)

4.5.5 Impact of Raspberry Pi Revisions

The Raspberry Pi Foundation often revises its current offering of devices, with modifications to various hardware components. As our implementation focuses on generating power models for the CPU, we suspect that revisions on the USB port or other minor modifications will only have a small impact on the average error of our models.

We proceed to run our benchmarking client and generate power models for the Raspberry Pi 4B revisions 1.1 and 1.2, as seen in the previous Tables 4.4 and 4.3. For this particular device, the differences between revisions 1.1 and 1.2 are minimal and related to the USB-C connector as some electronic components were added and reallocated to fix a fault regarding the connector.

Table 4.5 shows minor differences in the average error between the rev 1.1 and 1.2 for the same OS architecture (32 or 64 bits). This difference is not negligible for accurate power measurements as an increase of up to 56% was observed for using the power model of another revision. However, the average error compared to the power meter is still low in both power models when switching revisions (*i.e.*, around 3% to 4% for rev 1.1). Therefore, we recommend generating and using power models for specific revisions, while still allowing estimator clients to use another revision power model if one isn't provided for the specific revision.

4.5.6 Impact of 32 and 64 bits Raspberry Pi Versions

Newer Raspberry Pi devices have a 64 bit supported ARM architecture, where users can run either a 32 or a 64 bits operating system. As a stable 64 bits version of Raspberry OS hadn't been released during our experiments, we use its latest beta version (arm64-2020-08-24). Recent experiments had shown that a 64 bits OS on a Raspberry Pi 4 provides a much higher performance compared to a 32 bits OS, up to doubling the performances in benchmarks [131]. Therefore, we suspect that our power models generated in a 32 bits OS would not provide a similar accuracy on a 64 bits OS.

Table 4.5: Comparison of the average error of the linear models for 32 bits and 64 bits OS, for both revisions of Raspberry Pi 4B

RPi / Power model	RPi 4 B 1.1 32 bits	RPi 4 B 1.1 64 bits	RPi 4 B 1.2 32 bits	RPi 4 B 1.2 64 bits
RPi 4 B 1.1 32 bits	**3.86**	12.47	4.64	12.47
RPi 4 B 1.1 64 bits	10.68	**2.99**	12.22	2.97
RPi 4 B 1.2 32 bits	3.70	14.65	**2.97**	14.63
RPi 4 B 1.2 64 bits	10.60	3.97	12.09	**3.92**

For both Raspberry Pi 4B revisions, we generate power models running our benchmarks on a 64 bits OS, as seen in the previous Tables 4.4 and 4.3. As we suspected, our benchmarks running on a 64 bits OS have, on average, higher power consumption than the same device running the same benchmarks on a 32 bit OS.

Table 4.5 presents the average error of each generated power model when applied to the benchmark data of every experiment. We observe, consistently, that a different architecture highly impacts the accuracy of the generated power models, up to more than 5 times. For instance, RPi 4B rev 1.2 32 bits power models are nearly 5 times less accurate when used on the same revision but with a 64 bits OS. These results confirm our hypothesis and the higher performance of a 64 bits OS on supported devices as seen in the literature [131].

Consequently, we recommend using power models generated specifically for the device's architecture.

4.5.7 Impact of Connected Peripherals

Raspberry Pi devices are designed to be easily connected to external peripherals via a variety of physical interfaces. To assess the impact of connected peripherals on the validity of our energy models, we conduct the same benchmark experience in two different scenarios on a Raspberry Pi 4B, revision 1.2. In the first scenario, we disconnect all peripherals and follow the procedure mentioned in Section 4.5.1. In the second scenario, we launch the benchmark after connecting the Raspberry Pi to a screen using the mini HDMI interface, a USB wired keyboard, and a wireless mouse, and we activate WiFi.

We generate two power models, one for each scenario, and calculate its average error. As seen in Table 4.6, both CPU linear and polynomial models have a lower error when disconnecting the peripherals. However, the models generated with the peripherals are still within an acceptable margin below 8%. This proves that our approach can generate CPU power models with an acceptable accuracy even with interference from connected peripherals.

Table 4.6 also shows the average error when using the power model of one scenario onto the data of the other scenario, *i.e.*, using the power model generated with the peripherals on benchmarking data generated without the peripherals, and vice versa. The average error shows a lower accuracy of the model, but still within a range below 8%. This means our CPU power models generated in an ideal benchmark setup (with peripherals disconnected), are still accurate enough to, not only estimate the power of the CPU, but to estimate the power of the Raspberry Pi device (as the CPU is shown to be the most power-consuming component).

Raspberry Pi devices are often used in a headless server setup (such as to control industrial machines, control heating or lightning in a smart home or city, a web or NAS

server, etc.). In these situations, our approach generates accurate CPU power models without interference from peripherals.

Table 4.6: Comparison of the average error of the linear and polynomial models with and without peripherals on a Raspberry Pi 4B, rev. 1.2

Power Model	Data	
	with	without
Linear with	**4.19%**	7.83%
Linear without	7.34%	**2.92%**
Polynomial with	**3.73%**	7.72%
Polynomial without	6.91%	**1.64%**

4.5.8 Comparison to the State of the Art Models

To assess the validity of our models and our automated approach in generating up-to-date models, we compare our generated power models to the ones provided by the state of the art. In particular, we tested and compared our models to two models: PowerPi [124] and EMM [125].

PowerPi provided linear power models for the Raspberry Pi 2B and used a custom tool to stress the CPU using an infinite loop doing two additions of two integer variables. `cpulimit` was used to limit the CPU utilization which was stressed with a 10% step.

EMM provided linear power models for the Raspberry Pi 3B+ and used the `stress-ng` command to stress the CPU for a particular load. As we explained in Section 4.5.2, we found that using this command generates an inconsistent CPU load. In comparison, we use a more random CPU load generator (compared to PowerPi), and a more consistent CPU load tool (compared to EMM). We also stressed the CPU with a 5% step, collecting more data points at more CPU load percentages. Therefore, we have a more comprehensive and complete data set to generate more accurate power models. Our architecture also follows a crowd-sourced approach allowing adding additional benchmarks and data to constantly improve the generate power models.

PowerPi announces an average error of 1.2%, while EMM announces an average error of 1.25% with a maximum of 3%. However, in our experiments, we found that both models have a much higher error rate: 14.56% for PowerPi and 40.76% for EMM (cf. Figure 4.8). In contrast, our linear models provide an error rate of 3% for RPi 2B, and 6.71% for RPi 3B+, and even much lower error rates for our polynomial models (1.53% and 3.83%, respectively).

In addition, our approach is based on a dynamic model generation that deals with

dynamic workloads. It is validated on a large variety of Raspberry Pi devices in comparison with state-of-the-art methods (which are tested on a single device). It also allows the collection of a large data set due to its decentralized data collection technique. Another advantage of our approach is its flexibility: new models can be added or modified as the system evolves. As such, it can integrate into complex and moving environments (such as when new devices are often introduced or updated).

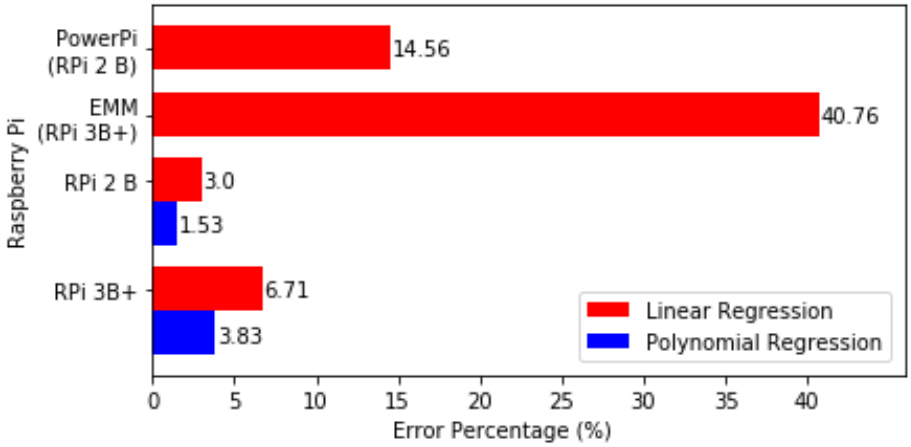
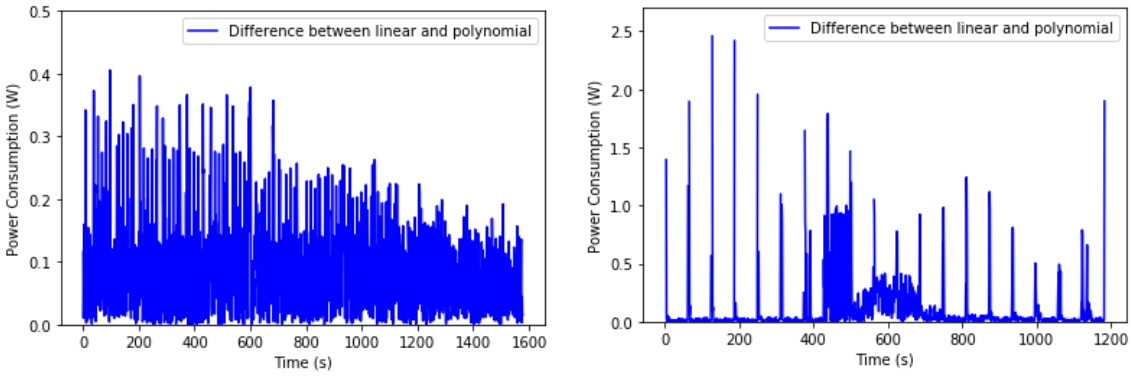


Figure 4.8: Average error percentage for linear and polynomial regression algorithms for our approach compared to the literature

4.5.9 Overhead of Regression Models

Most of our generated polynomial models have a degree of 9, requiring calculations up to the power of 9. In our analysis, we found that polynomial models with a higher degree than 9 have negligible accuracy improvements but with a higher calculation complexity. As these models have a much higher accuracy than the linear ones, we compared the overhead of running both models in our implementation of the power estimator client.

Our client is a minimal C program reading CPU cycles, calculating CPU utilization, and applying the power models. The client monitors power consumption at run-time and provides a power value every second. We compare the power overhead of running the client with both power models, and also in comparison to the base power consumption without our client. We conduct our experiment on two Raspberry Pi models: RPi Zero W for a low-power device, and the 3B+ for a more recent higher-power one.



(a) (b)

Figure 4.9: Power overhead of the (a) RPi Zero W and the (b) RPi 3B+

Figure 4.9 outlines the absolute differences in Watts between both our implementations: linear and polynomial models. For both devices, although we observe some rare data points of higher diversion, the overall difference is quite low with an absolute average difference of 0.084 watts (corresponds to a relative difference of 1.32%) for the RPi Zero W, and only 0.109 watts (corresponds to a relative difference of 0.72%) for the RPi 3B+. These numbers show a negligible overhead for using our polynomial models over the linear ones, even on low-power devices. We, therefore, recommend using the polynomial power models even for run-time power monitoring.

4.5.10 Validation of the Power Model Generator

The core idea of our power modeling generator, described in Section 4.4.2, is to allow third-party benchmarking clients to send new benchmark metrics to further improve the accuracy of the generated power models. In this section, we validate our approach with a breakdown of a step-by-step experiment of the linear power model in a Raspberry Pi 3B+ device.

We run our experiment 10 times, emulating 7 benchmarks sending data incrementally one after the other for one particular device (Raspberry Pi 3B+). Our server implementation then runs our model generator algorithms and keeps the best accurate power model in every step.

The result of this breakdown is outlined in Table 4.7 for the linear model, and Table 4.8 for the polynomial model. Each row of the table represents a new server iteration (receiving new data, data normalization and cleaning, validation, power model generation, and comparison, etc.). $M'(n)$ indicates the newly generated power model in the server, along with its error rate $E'(n)$. $E(n)$ is the error rate of the currently saved power model

using all the data. And $M(Server)$ or $M(S)$ is the power model that is saved after the current iteration.

Table 4.7: Breakdown of our Power Model Approach (linear model on RPi 3B+, bold for the selected model)

Step	M'(n)	E'(n)	E(n)	M(Server)
S1	$P = 3.357 \times U + 2.024$	**7.64%**	-	$S1$
S2	$P = 3.271 \times U + 2.032$	**6.61%**	7.07%	$S2$
S3	$P = 3.272 \times U + 2.011$	**6.13%**	6.23%	$S3$
S4	$P = 3.285 \times U + 2.014$	6.58%	**6.51%**	$S3$
S5	$P = 3.296 \times U + 2.004$	6.72%	**6.66%**	$S3$
S6	$P = 3.294 \times U + 1.997$	6.45%	**6.43%**	$S3$
S7	$P = 3.292 \times U + 1.992$	**6.25%**	6.27%	$S7$

Table 4.8: Breakdown of our Power Model Approach (polynomial model on RPi 3B+, bold for the selected model)

Step	P[0]	Degree of M'(n)									E'(n)	E(n)	M(S)
		1	2	3	4	5	6	7	8	9			
S1	2.34	11.95	-334.15	3997.0	-23579.71	76091.11	-140313.95	147466.42	-82193.81	18858.15	**3.68**	-	$S1$
S2	2.45	5.33	-186.67	2534.97	-15983.94	53687.34	-101595.06	108702.03	-61390.2	14229.05	**3.73**	4.15	$S2$
S3	2.49	2.66	-129.57	1976.33	-13076.81	45047.92	-86526.32	93476.51	-53148.84	12380.92	**3.63**	3.79	$S3$
S4	2.48	3.07	-146.99	2215.22	-14580.93	50066.17	-95944.1	103459.96	-58730.58	13660.99	3.8	**3.71**	$S3$
S5	2.49	2.96	-151.39	2289.23	-15064.96	51709.94	-99070.29	106810.86	-60621.69	14098.12	3.82	**3.73**	$S3$
S6	2.51	1.69	-123.33	2010.51	-13596.98	47304.09	-91325.7	98938.95	-56341.33	13134.87	3.78	**3.71**	$S3$
S7	2.52	0.74	-102.99	1810.06	-12545.46	44155.31	-85798.75	93326.76	-53291.99	12449.07	3.74	**3.69**	$S3$

As we can observe in this breakdown, the newly generated model is not always the most accurate. For instance, for the linear model, at step 4, the new model has a worst average error (6.58%) compared to the current model (6.51% calculated with all data including the new ones). This also happens in steps 5 and 6. However, across multiple iterations, we observe a decrease in the average error, which started at 7.64%, then gradually went down up to 6.25% after only 7 benchmarks and model iterations. We observe a similar breakdown for the polynomial model with an improvement of the error rate and our approach uses the most accurate power model on every step.

We argue that the more benchmark data we have, the more our architecture and approach will provide empirical power models with better accuracy.

4.6 Use Cases

4.6.1 Use Case of Remote Power Monitoring

A use case illustrating the advantages of our approach is remote power monitoring of a park of deployed Raspberry Pi devices. Examples of such use cases vary from monitoring environmental metrics [132, 133], smart management [134], or health [135]. In particular, smart devices, such as Raspberry Pis, send collected metrics and their status (including CPU statistics) to a central monitoring service.

An example of the latter is Zabbix [3], an open-source server used for real-time monitoring of a large number of clients. In each Raspberry Pi client, a Zabbix agent is installed to allow remote monitoring and management. It can send CPU utilization and many other metrics for the device in real-time. We developed a prototype plugin for Zabbix to integrate our power models and architecture into its web interface. Our plugin updates the power models of the monitored devices by connecting to our power modeling server. It also tweaks the Zabbix web interface to calculate the power consumption of monitored devices, in real-time, and displays them along with the CPU utilization, as seen in Figure 4.10.

With our approach and power models, remote management tools can efficiently, accurately and with no overhead on the monitored devices, monitor the power consumption in real-time. It also allows these tools to always have updated and accurate power models, and to support new device power models easily by just calling our sharing interface.

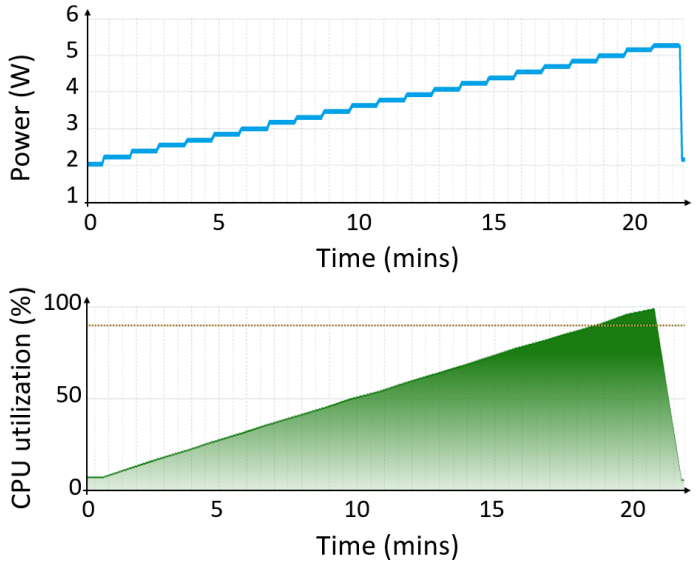


Figure 4.10: Power consumption of a RPi displayed on a Zabbix server web interface

[3] https://www.zabbix.com/

4.6.2 Use Case of LED Bulb Power Estimation

A use case of a smart LED bulb is experimented in order to evaluate the approach of power estimation models generating and its suitability with devices different than RPi. LED bulbs are considered one of the basic devices to have in a smart home, thus, the importance to test our approach on them.

To collect power consumption, we use the PowerSpy2 power meter described in 4.5.1. We run all our experiments on a smart RGB multi-color LED bulb [136] manufactured by LSC Smart Connect (Ref. 2578539 , 9 watts, 806 lumens), that communicates using 2,4 GHz Wi-Fi. For each of the experiments a total of approximately 800 data points is collected. We exclude the data when the brightness is equal to 0 because at that level of brightness none of the single led lights that consists the LED bulb is turned on so the only component consuming energy is the Wi-Fi circuit. The power consumption in this particular condition is 0.62919 W.

The first experiment is conducted on the white color setting, and the parameter that we change is the brightness from 0 to 100% with a 10% step. The collected data is analysed and a power estimation model is generated. Figure 4.11 presents the measured correlation between brightness (in percentage), and the power consumption (in watts) for our linear regression model and the actual measurements from the power meter.

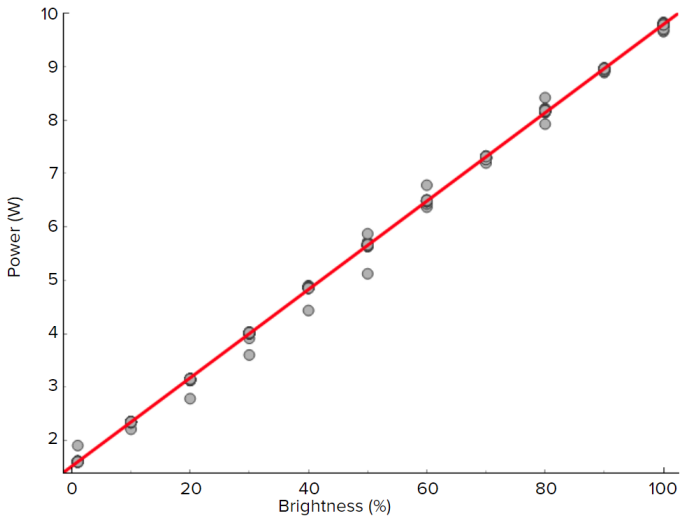


Figure 4.11: Linear regression power estimation model of the white LED bulb

The second experiment shows the different power consumption of different colors on a 100% brightness level, as seen in figure 4.12. We choose 6 basic colors (blue, cyan, green, magenta, red, and yellow) but the power consumption of any color can be measured.

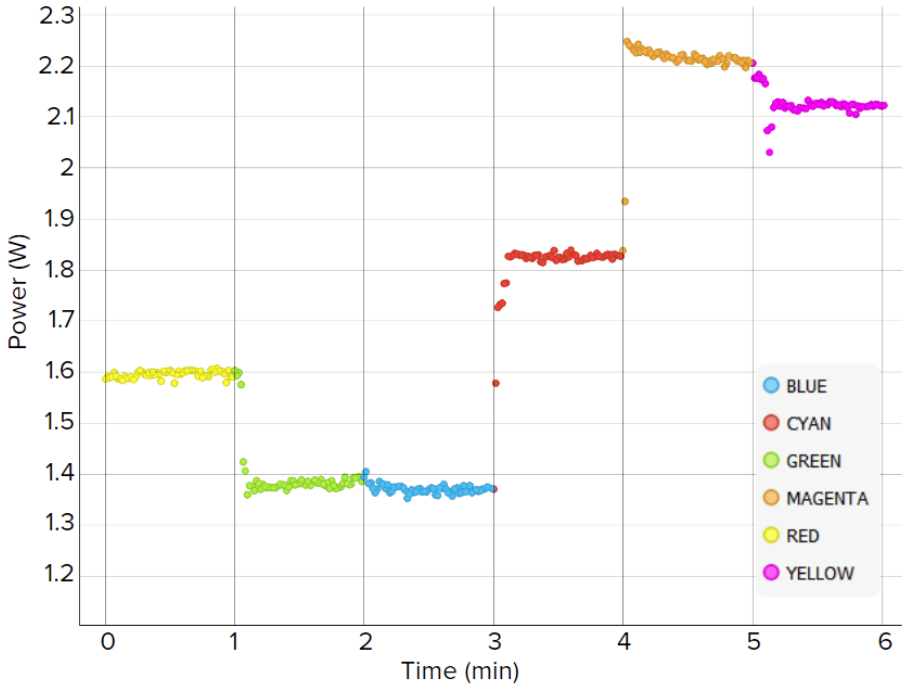


Figure 4.12: Power consumption of the LED bulb for different colors

The third experiment combines the two previous ones. We change the colors and for each of the colors we vary the brightness level of the light, as seen in Figure 4.13. For each color a new estimation model is generated to maintain the high accuracy of the models.

Table 4.9 presents the generated LED bulb power estimation models using linear regression algorithms for each of the chosen colors and their error percentages. Our approach proves its compatibility with different smart home environment devices when data of the device are available. In particular, it provides high accuracy with low error rates.

Table 4.9: LED bulb generated power models using linear regression algorithms

Color	Estimation Model	Error (%)
White	$P = 8.26923 \times Brightness + 1.51994$	0.5000
Red	$P = 0.84307 \times Brightness + 0.76452$	1.2983
Green	$P = 0.61603 \times Brightness + 0.67526$	2.3818
Blue	$P = 0.67791 \times Brightness + 0.66825$	2.8333
Cyan	$P = 1.30569 \times Brightness + 0.65592$	4.0611
Magenta	$P = 1.49404 \times Brightness + 0.75228$	2.7146
Yellow	$P = 1.47088 \times Brightness + 0.75325$	2.5741

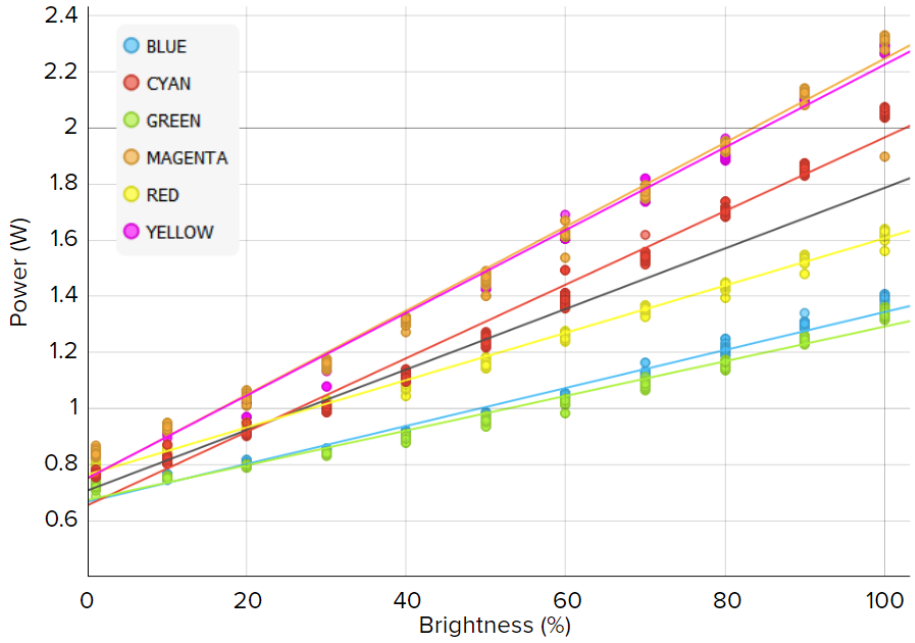


Figure 4.13: Linear regression power estimation model of the LED bulb for different colors

4.7 Threats to Validity

Our approach and experimentation suffer from the following threats to validity:

- Our implementation is limited to one type of single-board computers, i.e., Raspberry Pi devices and one type of home devices, i.e., led bulb. We run our experiments on a wide variety of RPi device models from each generation, but some devices and revision models were not modeled.
- Our implementation only uses few metrics to correlate to the power consumption, i.e., CPU utilization, LED brightness, LED color. Even though our results show a strong correlation, and allow us to generate very accurate power models (that can have an average error as low as 0.3%), we did not investigate additional metrics or other hardware components (such as the WiFi or Bluetooth).
- Although we made sure no interference happened to our experiments and data collection, we could not formally discard that no external factors impacted the results. In particular, the experiments spanned over 4 months in different weather conditions (winter and spring) and therefore different room temperatures. Additionally, the power overhead of the benchmarking client was not subtracted from the collected data because we wished to emulate real usage of the client (where it is difficult to

automatically deduce a variable overhead. This deduction process might require additional data collection, such as the CPU cycles of the client process, and therefore adds an additional overhead itself).

4.8 Summary

In this chapter, we presented an architecture to automate the generation of always up-to-date and accurate power models for a variety of devices. Our approach allows crowd-sourced benchmarking of devices, the collection of various metrics, and the generation of specific power models based on the collected data. A sharing interface allows power estimator client devices to query and retrieve the most accurate power model available for the client's device.

We implemented a proof-of-concept client and server to automate the generation of power models for Raspberry Pi devices and home devices i.e, LED bulbs. We also conducted a comprehensive experiment validating our approach, algorithms, and power models. The latter provides high accuracy with error rates as low as 0.33% and up to 7.81% for linear models, and 0.3% up to 3.83% for polynomial models. Furthermore, we analyzed and discussed the impact of device revisions, CPU and OS architectures, and the overhead of both generated power model types. Finally, we validated our approach in the power model generator and provided an example of a use-case scenario of our models in remote power monitoring.

We present the implementation of the power modeling and estimation component previously introduced in chapter 3. It can generate and share power models with other system components, namely the intelligence management component. It aims to have a vision of power consumption for management purposes.

In the next chapter, we present the remaining contributions, namely, the knowledge component and the reinforcement learning agent generator. In addition to the experiments conducted to validate our approach.

Chapter 5

Automated Energy Management Framework

Contents

5.1 Introduction

Power consumption is increasing in recent years, especially in the residential sector. Therefore, estimating the power consumption of devices is an essential step into managing energy in connected environments. In the previous chapter, we presented our approach for modeling and estimating power for heterogeneous devices. Once the power consumption of each device is known, we get a better visibility on energy drains that enables us to accomplish energy management actions, and taking appropriate actions to reduce energy consumption is necessary. However, users taking energy optimizing decisions are not always satisfying energy-efficient decisions due to a lack of knowledge (e.g., right timing, appropriate actions to take, and parameters to set). In addition, the consideration of all possible scenarios through a rule-based reasoning, used widely these days, becomes impossible with all potential devices and choices. Therefore, we believe it is necessary to develop a decision-making framework that operates as Energy Management Systems (EMS) in complex and heterogeneous environments.

Based on the state of the art of potential approaches and technologies to develop this EMS, two well-adapted solutions have been selected: (a) ontologies and (b) Reinforcement Learning (RL). On the one hand, ontologies can be used as a reliable knowledge representation technique. On the other hand, reinforcement learning has emerged as a powerful machine learning technique for dynamic environments, characterized by changing situations and needs for personalizations, where actions can have positive or negative consequences. It enables the development of a behavior by taking actions and getting feedback from an environment through a trial and error process. In addition, users interact directly with devices and any actions accomplished on a device will impact the user comfort. Thus, user preferences are highly considered in order to ensure the greatest user satisfaction.

In this chapter, we present our framework for automated energy management in section 5.2. We propose our approach of representing energy related knowledge in CPS through an energy-oriented ontology extension in section 5.3. Then, we describe our automated reinforcement learning agent and environment generation algorithm in section 5.4. Results and discussions will be presented in the next chapter.

5.2 Automated Energy Management Architecture

In order to solve the problem of energy management in smart and connected systems (i.e., smart homes and CPS), we propose using an automated approach to generate the characteristics of the environment and the decision agent that accomplishes energy-efficient actions based on RL algorithms. In the following, we use the previously presented contribution that generates power estimation models (Chapter 4). In addition, we detail our knowledge management component. To do so, we propose an energy-oriented ontology extension by adding the power consumption concern and other necessary concerns to the ontology. The extension allows proper representation of the relationship between power consumption, properties, and devices. We also detail our intelligence management component that uses this knowledge to generate RL agents and environments based on the present devices and their interactions with each other. Then, we train our agent with the generated virtual environment and show that energy optimization can be done while respecting user preferences.

Concretely, Figure 5.1 presents our automated energy management physical architecture. It is composed of three distinct but complementary components: knowledge sources (power modeler, ontology, and user preferences), environment generator, and RL environment (agent, virtual environment, and interpreter). We detail the main components of our architecture in the following:

Power Estimation Model Repository. It aims to save and share generated power estima-

tion models for different devices. At this stage, we extract power estimation models for devices present in the environment.

Environment Generator. This component generates RL environments by collecting knowledge from the ontology and the power modeling component.

ECPS Ontology. This component contains information representing the environment and aims to provide a knowledge base for the environment generator.

User Preferences. This component contains the user preferences that can be an exact value (e.g., binary state of a device) or min/max value (e.g., light brightness).

RL environment. This component is composed of three sub-components: agent, virtual environment, and interpreter. The agent applies action to the virtual environment based on the states and rewards.

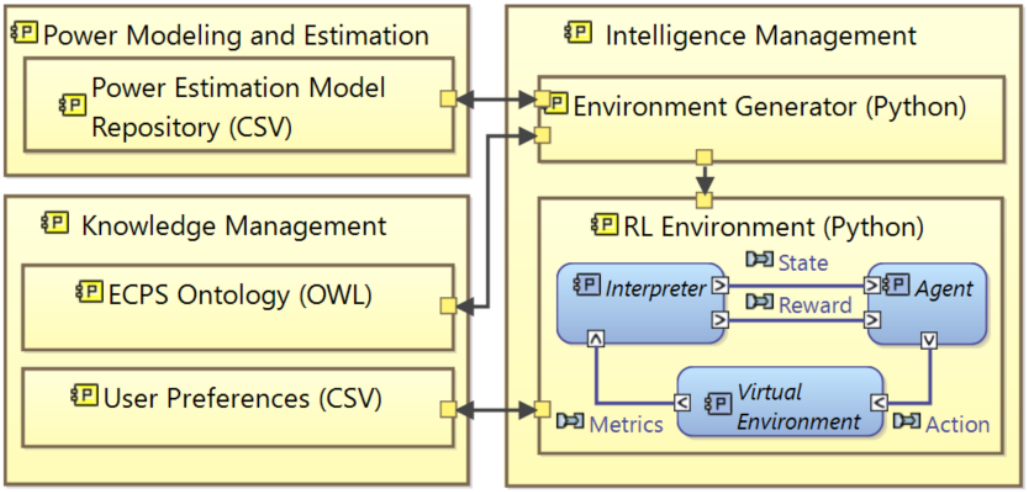


Figure 5.1: Knowledge and intelligence management physical architecture

5.3 Knowledge Management

An ontology is a defined set of concepts and relationships used to represent the meanings and a shared understanding in a certain domain [100]. Ontologies are considered highly effective for knowledge modeling and information retrieval. They provide a common representation and definition of the concepts and the relationship between them. In addition, ontologies support inference to discover new relationships and knowledge. They

are scalable and their reuse is possible. Therefore, an ontology is used to formally represent knowledge that is found in an environment. In the following, we compare state-of-the-art ontologies for connected environments then we present our energy-oriented ontology extension applied on the SAREF ontology.

5.3.1 Related Work of Ontologies for Connected Environments

There has been numerous studies that proposed ontologies defining the entities and properties of connected devices and appliances. In the following, we focus on the most broadly used ontologies in the domain of smart homes, especially the ones that aim to optimize energy:

DesMaHo [137] is an energy ontology focusing on demand side management in smart home. The goal of this ontology is to use produced power (local or external) for running appliances with respect to user preferences. DesMaHo divided devices based on their loads and their ability to produce or store energy. Loads are defined as shiftable, not shiftable, and program dependently shiftable. It does not consider data generated by devices in a fine-grained manner.

DIMMER [138] is an ontology that exploit information about energy performance indicators on the city, district, and building level. Its implementation allowed access and visualization of energy related information, cost analysis, and user behavior. However, the devices in the building were not modeled leading to a lack of accurate knowledge of contextual changes and data collected by sensors.

SARGON [139] is a smart energy domain ontology that extends SAREF with domain-specific information representing energy in buildings and electrical grids. However, it considers energy on a high level of the environment such as building and grid instead of the energy used per device leading to a lack of correlations between the measurements and power consumption. In addition, devices already present in the environment that have sensing and actuation capabilities are not considered as control devices.

Web-of-Objects (WoO) [140] defines a systematic description of smart distributed applications enabling the integration of any connected device. It considers some contextual data (e.g. weather and location), however, it does not consider all potential layers of the system and the real time power consumption per device.

RealEstateCore (REC) [141] is an ontology aiming to allow data integration for smart buildings. It is developed to support two specific use cases: Energy usage analysis and presence analysis. It defines devices as part of a building with sensing and actuating capabilities without a visibility on power consumption per device.

ThinkHome [142] introduces semantic context and artificial intelligence to achieve a smart home. It is based on an ontology for knowledge representation. ThinkHome con-

siders mainly user comfort, user behavior, contextual information (e.g. weather), building information (e.g. spaces and walls), and energy information (e.g. environmental impact and energy sources). Lack of heterogeneity is the main drawback of this approach because it focused on HVAC without studying other types of devices.

Semantic Sensor Network Ontology (SSN) [143] includes a core light weight ontology called SOSA (Sensor, Observation, Sample, and Actuator). It was strongly influenced by SSNX ontology. SSN/SOSA describes sensors (e.g. observations, features of interest, samples, and observed properties) and actuators. Both Semantic Sensor Network (SSN) and SOSA can support different use-cases and applications including IoT, industrial and household infrastructure, and large scale scientific monitoring. However, they do not consider state of devices and energy information per device.

Smart Applications REFerence Ontology (SAREF) [144] focuses on the concept of device as a central concern and defines its properties that be measurable, controllable, or both. In addition, it defines the state of the device that can be appliances, sensors, actuators, meters, and HVAC.

Table 5.1 compares different ontologies for smart homes and connected environments. We compare these ontologies, based on their degree of representation of concerns by using four comparison levels: an ontology fully represents a concern (*High*), it represents a concern in a limited manner (*Medium*), it represents a concern minimally (*Low*), or it does not represent a concern (*None*). We compare ontologies based on the following criteria:

- *Power consumption per device.* It shows the ability of an ontology to represent the manufacturer typical power consumption and the real-time power consumption per device.

- *Sensor/Actuator data.* It shows the possibility to collect data from sensors and actuators on different devices.

- *Contextual data.* Different types of data other than the measured ones can be represented (e.g., outside temperature, user presence, and user activities).

- *Fine-grained data.* Determines the ability of an ontology to represent specific concerns involving great attention to details.

- *Ease of implementation with RL.* An ontology can be implemented easier with a RL technique if it represents the three main concepts of RL: actions, states (controllable and uncontrollable), and metrics.

Table 5.1: Comparative table of potential ontologies for energy management in connected environments

Reference	*Power Consumption per device*	*Sensor/Actuator data*	*Contextual data*	*Fine-grained data*	*Ease of implementation with RL*
DeSMaHo [137]	*Low*	*Low*	*None*	*None*	*Medium*
DIMMER [138]	*Low*	*None*	*None*	*Low*	*Low*
SARGON [139]	*None*	*Low*	*Low*	*Low*	*Low*
WoO [140]	*Low*	*Medium*	*Medium*	*Low*	*Medium*
REC [141]	*Low*	*Medium*	*Medium*	*Low*	*Medium*
ThinkHome [142]	*Low*	*Medium*	*High*	*Low*	*High*
SSN [144]	*Low*	*High*	*High*	*Medium*	*Medium*
SAREF [143]	*Low*	*High*	*High*	*Medium*	*High*

Table 5.1 shows that the most two suitable ontologies based on our defined criteria are SSN and SAREF. The main problem is that ontologies do not usually consider power estimation and measurement. They lack of representation of different layers of a system and location of devices. Therefore, it is necessary to extend one of these ontologies to represent these concepts. The way these concepts are defined in SAREF provides a compatible and easy way to represent the knowledge base for reinforcement learning model generation.

5.3.2 SAREF Ontology

SAREF [144] is a well-known and adopted ontology in research. It focuses on the concept of a device as a central concern and defines its properties. These properties can be measurable, controllable, or both. The main entities of the SAREF architecture are the following: A *device* is the main focus and center of all entities. It defines the type of the device (i.e., appliance, sensor, actuator, meter, or HVAC). A *task* is the goal for which a device is designed from a functional viewpoint. Each device is specified by a *profile*. *Properties* are quality of a feature of interest that can be measured or controlled. *Measurements* are measured values made over properties. *Functions* are series of commands that work together and complete each other to accomplish a task. *Services* are the representation of functions to a network. It allows the function to be accessible over the network. *Commands* are low level orders used to change the state of a device. *States* are operating modes in which a

device is running. *Commodity* is an economic resource from which a device consumes, e.g., electricity, gas, coal, and oil. *Energy and power consumption* of a device are the ones typically provided by the manufacturer, therefore, it can be useful in cases where it is impossible to measure or estimate the real-time power consumption of a device. In the next section, we detail the energy-oriented extension proposed for SAREF.

5.3.3 Energy-Oriented Ontology Extension

Different widely used ontologies were proposed in the literature. However, they did not consider real-time power estimation and management, the spacial location of devices, or various metrics from different layers of a system. In order to include the real-time power and different layers of a device into the knowledge base, we propose the extension of the SAREF ontology with the following classes, attributes, and relations:

- ***ecps:Location*** is the geographic space in which a device is deployed. It helps build a contextual understanding of the environment and identify devices that share the same location. In our case, we used an identifier to describe the location of a device, however, locations can be defined using the alignment with other ontologies such as the Building Topology Ontology (BOT) [145].

- ***ecps:Layer*** is the structural level of the system to which a property belongs. It can be one of the following layers of a CPS identified in chapter 2: processing, communication, services, control, sensing, and actuating. In addition, we consider the contextual layer that contains external properties that a device can measure.

- ***ecps:LayerState*** is the state in which a specific layer is running (e.g., communication layer is running on a low power consumption mode when there is no need for exchanges). All layer states combined compose the state of the device.

- ***ecps:Power Estimation Model*** is the mathematical model that takes as input the properties of a device and returns the real time power consumption. Power estimation models for different types of RPi is presented in chapter 4. Power estimation models are not limited to the ones previously generated, any model can be adopted.

- ***ecps:Real Power*** is the real-time power consumption of a device. It can be measured directly by a hardware wattmeter or estimated using an estimation model. The *ecps:Real Power* concept represents the real time power consumption, in contrast with *saref:Power* that is the typical power consumption defined by the factory during the manufacturing process.

- ***ecps:hasRange*** represents the accepted values for each property. It takes one of the following forms: (i) a list of accepted values (e.g., light color [red, green, blue, white]); (ii) a range with a minimum and maximum accepted values (e.g., humidity [0,100]); and (iii) ranges specified with a minimum and maximum accepted values that includes a step that defines a fixed value increased or decreased from a property measurement (e.g., light brightness [0,100,1%]). We define *ecps:hasRange* to specify the boundaries and our maneuver intervals for each variable.

- ***ecps:deviceId, ecps:deviceName, ecps:propertyId, ecps:propertyName*** represent respectively the id and name of a device used to identify a device from another. In addition to the id and name of a property used to identify to which device belongs each property. These definitions are used to generate the source code of the RL agent and name the variable corresponding to each property.

- Relations between the classes and the data properties are defined by the following predicates: *ecps:belongsTo, ecps:consumes, ecps:contains, ecps:affects, ecps:estimates, ecps:hasState, ecps:isAppliedTo, ecps:isComposedOf, and ecps:isInside*.

Figure 5.2 shows the energy-oriented extension applied on the SAREF ontology with the previously proposed classes and relations.

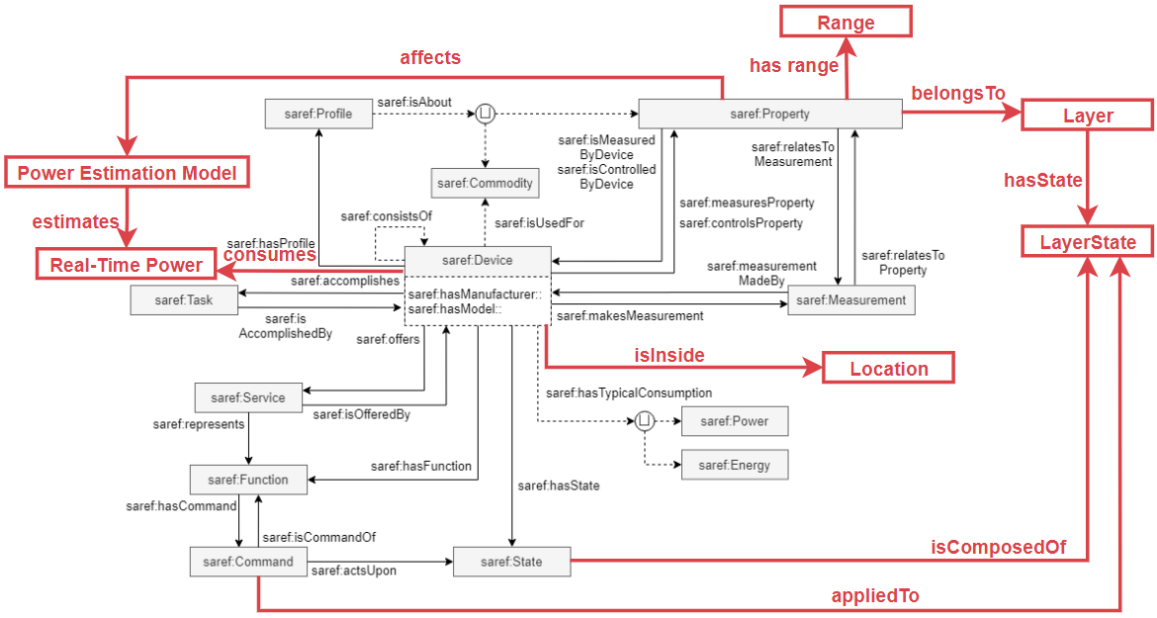
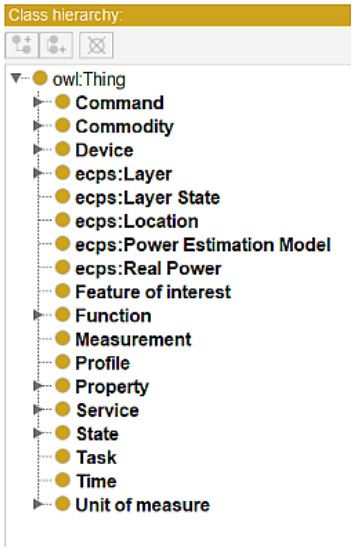
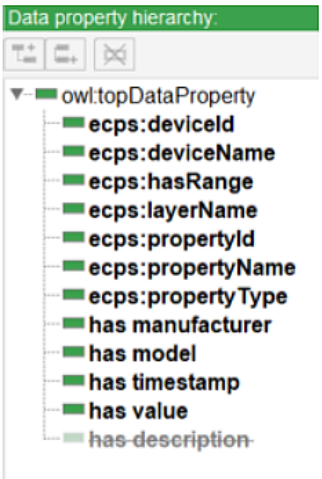


Figure 5.2: SAREF extension overview

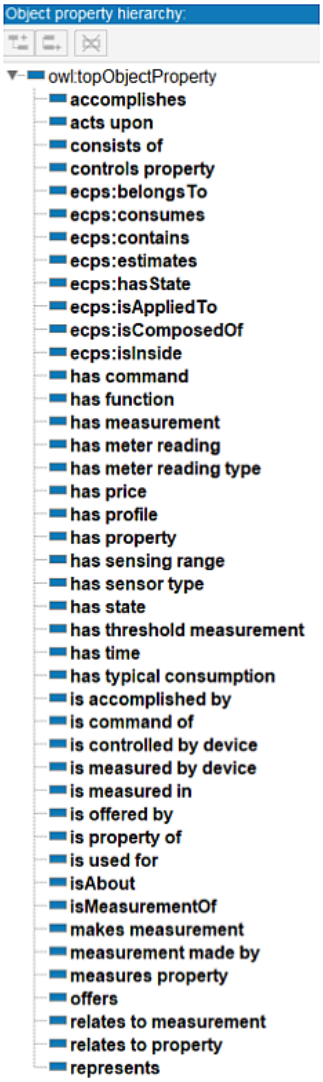
The ontology is defined using the Web Ontology Language (OWL) [146], a language of knowledge representation for ontologies in the Protégé ontology editor and knowledge management system [147]. As seen in Fig. 5.3, a series of classes, data properties (attributes), and object properties (relations) are defined in order to make the ontology suitable with the environment generator. Newly introduced concepts have names that start with the prefix *ecps*.



(a) Class Hierarchy



(b) Data Property Hierarchy



(c) Object Property Hierarchy

Figure 5.3: SAREF ontology extension implementation in Protégé

The proposed energy-oriented extension for the SAREF ontology defines the architecture of a device and the relationship between each device, properties, actions, power consumption, and the environment in which a device is present. We present, in the next section, RL environment generation algorithms.

5.4 Intelligence Management

The intelligence management component aims to collect knowledge from the knowledge management component and create a virtual training environment and a RL agent based on the devices present in the environment and the relations between them. The virtual environment is used to train the agent to accomplish the best power optimizing action.

The developed energy-oriented ontology extension provides the necessary knowledge to the intelligence management component through SPARQL queries. Deep reinforcement learning has emerged as an effective approach for solving different sequential decision making problems (e.g., video games, robot manipulation, and board games). Appropriate simulation environments are the main drivers of the success and satisfying results of reinforcement learning. However, generating RL environments is time consuming, complex, and oriented to specific scenarios. In the following, we present our intelligence management component. First, we compare state-of-the-art reinforcement learning environment generation approaches. Then, we present our algorithm for environment and agent generation.

5.4.1 RL Environment Generation Related Work

In the recent years, a large number of existing studies in the broader literature have examined the use of machine learning, in particular reinforcement learning, to accomplish a variety of goals. However, few research studies focused on the generation of reinforcement learning environments. Most studies have created and parameterized environments and agents manually. In the following, we focused on the reinforcement learning environment generation:

Environment generation. Few are the literature studies that focused on RL environment generation. In [148], authors proposed a solution to automatically build a series of complex tasks based on a RL agent's current skill level. However, the latter does not build an environment from scratch but rather it generated complex tasks from a small set of basic actions. In [149], authors proposed an approach to modify environments based on the difficulty of environmental challenges. It pairs the generation of challenges and the optimization of agents to solve those challenges. It aims to generate the most efficient agent by continuously copying agents that have best results to be trained with other altered environments. A solution for exploration problems in RL algorithm is proposed in [150]. It is based on changing the difficulty of the environment using a more complex set of goals each time the agent reach an equilibrium. In [151], authors proposed generating entire environments instead of only the actions of a policy. It aims to increase the usability of RL beyond predefined environment. [149, 150, 151] are efficient to train agents, however, they all start by mutating an existing environment that has limited changeable parameters.

Automatic scenario generation. In the literature, several techniques have

been adopted for automatically generating diverse scenarios in a RL environment. A probabilistic object-oriented programming language for designing, modeling, and generating scenarios called SCENIC was proposed in [152]. This domain-specific programming language generates data sets useful for deep learning tasks by assigning different configurations for objects and agents, as well as imposing constraints with variable difficulties. SCENIC was used, in [153], to generate a dataset for Google Research Football environment [154]. In [155], authors formalize automatically generating scenarios for new difficulty levels using RL. However, generated scenarios were simple and implemented in the same environment. Many studies used scenarios generation for critical and extreme situations testing, specially for autonomous vehicles [156, 157, 158, 159].

Reinforcement learning for energy management. Recently, there is a growing body of work focused on using RL for energy management in smart homes and buildings [160, 161]. In [162], authors modeled the energy management system using a Markov decision process. They described states, actions, transitions, and rewards of the environment. Their environment is based on a smart energy building connected, an external grid, a distributed renewable energy source, an energy storage system, and a vehicle-to-grid station. In their study, actions were limited to {Buying, Charging, Discharging, Selling} energy, thus, a high-level solution with no impact on individual devices. In [163], authors proposed a RL algorithm that ensures user satisfaction, followed by energy savings and load shifting. It focused on houses equipped with renewable energy resources and setting a dynamic indoor temperature setpoint. A multi-agent RL for home-based demand response was proposed in [164]. Each of the agents was responsible for a type of device (non-shiftable, power-shiftable, time-shiftable, or electric vehicle). In [165], authors proposed a home energy optimization strategy based on deep Q-learning (DQN) and double deep Q-learning (DDQN) to perform the scheduling of home appliances tasks. In [166], authors proposed an intelligent multi-agent system, called Thinkhome. It was responsible for the execution of control strategies that manage the building state. In [167], authors proposed a demand response algorithm of the next hour for home energy management systems. It focused on reducing electricity cost by shifting loads based on historical data. [168], proposed a deep RL energy management algorithm based on Deep Deterministic Policy Gradients (DDPG) to control a HVAC system. However, an important issue in the previously mentioned existing research is that their environments are static and it is difficult to train agents in new environments.

5.4.2 Environment and Agent Generator

Prior RL environment generation research approaches are inconsistent regarding the heterogeneity of potential devices, flexibility in environment generation, and ease of implementation. We believe that this approach will help us prepare agents for the complexity and

diversity of real-world tasks without needing to manually specify training environments to cover this diversity.

The environment and agent generator is responsible for the generation of each section of the reinforcement learning agent based on TensorFlow and Keras-RL, in addition to an environment built using OpenAI Gym. Algorithm 1 provides the pseudocode of how the reinforcement learning environment is generated. First, a series of SPARQL queries are executed to collect knowledge from the ontology using the *request_sparql* function. Graphs containing information concerning actions, metrics, states, and power estimation models of each device corresponding to a unique location are collected. Four main SPARQL requests provide the agent constructor with the necessary knowledge to build the reinforcement learning agent. Queries 1, 2, 3, and 4 show respectively the SPARQL queries used to select actions that can be accomplished in the environment, power consumption models, metrics that can be collected, and the states each device can have. These four queries are inspired by the MAPE loop. Query 1 illustrates the execute and plan phases, query 2 illustrates the analyze and plan phases, and queries 3 and 4 illustrate the monitor and plan phases.

Query 1: Selects the actions that can be accomplished in an environment

```
SELECT ?Device ?Property ?DataType ?Range
WHERE {
        ?Device saref:controlsProperty ?Property;
        saref:ecps:isInside saref:Room1.
        ?Property rdf:type ?Class.
        ?Class rdfs:subClassOf ?SuperClass.
        ?SuperClass owl:onDataRange ?DataType.
        ?Property saref:ecps:hasRange ?Range.
    }
```

Query 2: Selects the power consumption models

```
SELECT DISTINCT ?Power ?DataType ?PEM_value ?State ?Device
WHERE {
        ?Power rdf:type saref:ecps:RealPower.
        ?Device saref:ecps:consumes ?Power.
        ?PEM saref:ecps:estimates ?Power.
        ?PEM saref:hasValue ?PEM_value.
        ?device saref:ecps:isInside ?location, saref:Room1.
        ?Power rdf:type ?Class.
        ?Class rdfs:subClassOf ?SuperClass.
        ?SuperClass owl:onDataRange ?DataType.
        ?Device saref:hasState ?State.
    }
```

Query 3: Selects the metrics that we can collect from device

```
SELECT DISTINCT ?Property ?DataType ?Range ?Value
WHERE {
        {
            ?Device saref:controlsProperty ?Property;
            saref:ecps:isInside saref:Room1.
            ?Property rdf:type ?Class.
            ?Class rdfs:subClassOf ?SuperClass.
            ?SuperClass owl:onDataRange ?DataType.
            ?Property saref:hasValue ?Value.
            ?Property saref:ecps:hasRange ?Range.
        }
        UNION
        {
            ?Sensor saref:measuresProperty ?Property.
            ?Device saref:ecps:isInside saref:Room1.
            ?Property rdf:type ?Class.
            ?Class rdfs:subClassOf ?SuperClass.
            ?SuperClass owl:onDataRange ?DataType.
            ?Property saref:hasValue ?Value.
            ?Property saref:ecps:hasRange ?Range.
        }
    }
```

Query 4: Selects the states that each device can have

```
SELECT DISTINCT ?Device ?StateType ?Range ?Value ?State
WHERE {
        ?Device saref:hasState ?State.
        ?State rdf:type ?StateType.
        ?StateType rdfs:subClassOf ?SuperClass.
        ?State saref:hasValue ?Value.
        ?State saref:ecps:hasRange ?Range.
    }
```

The collected knowledge is used for the creation of the source code of all the functions that compose the reinforcement learning environment. As seen in algorithm 1, a function called *generate_code* generates the source code of each of the reinforcement learning environment sections based on the collected ontology knowledge. It generates the source

code of six main components: (a) actions, (b) states, (c) rewards, (d) power estimation, (e) initialization, and (f) reset. A template of the source code of the reinforcement learning environment is created. It includes the necessary libraries, fixed chunks of code, and variable parts in the source code. However, the automated source code generator writes the majority of the code by using the *write_code* function. It identifies the blank variable sections of the source code that change from one environment to another and writes the corresponding code in each section. It can generate the source code of any number of properties and devices due to its extensible way and ability to request knowledge and use it.

Algorithm 1: Reinforcement learning environment generation algorithm

procedure GENERATERLE($ontology$)
  Initialize sparql queries
  $actions, metrics, states, power \leftarrow request\_sparql(ontology, sparql\_query)$
  $actions\_code = generate\_code(actions, states)$
  $states\_code = generate\_code(metrics, states)$
  $rewards\_code = generate\_code(power, metrics)$
  $power\_estimation\_code = generate\_code(power, metrics)$
  $init\_code = generate\_code(actions, metrics, states, power)$
  $reset\_code = generate\_code(actions, metrics, states)$
  $write\_code(init\_code, reset\_code, actions\_code, states\_code,$
    $rewards\_code, power\_estimation\_code)$
end procedure

Once the source code is generated, the reinforcement learning environment is ready to be launched. Its structure is presented in algorithm 2. The following explains the role of each of its functions:

- *__init__* function defines the observation and the action spaces, i.e., the minimum and maximum acceptable values for each observed metric or action. In addition to a few other attributes, i.e., accepted values for each property and the initial state of each device. This function also initializes states, properties, and power consumption per device. It also encodes and maps properties and states from their initial format to a numerical or Boolean format to be accepted by the reinforcement learning algorithm.

- *step* function executes a step in the environment by applying an action and returns the new reward, observation, power consumption per device, and other info. It includes the list of possible actions and the reward calculation based on user preferences and total power consumption. In this function, randomness and external changes impacting the environment can be applied (e.g., outside temperature changes).

Algorithm 2: Reinforcement learning environment algorithm

Procedure `__init__(self)`**:**
- Set controlled properties upper bounds
- *Set controlled properties lower bounds*
- *Set measured properties observation space*
- *Initialize a set of states*
- *Initialize a set of properties*
- *Initialize a set of power consumption*
- *Map states to integer values*
- *Initialize total energy*

Procedure `reset(self)`**:**
- *Reinitialize states*
- *Reinitialize properties*
- *Reinitialize power consumption*
- *Reinitialize total energy*

Function `calculate_power(device_type, device_state, properties[])`**:**
- **if** *device_type* $\in$ *devices* **then**
  - $power = estimate\_power(device\_type, device\_state, properties[])$
- **else**
  - $power = default\_manufacturing\_power(device\_type, device\_state)$
- **return** power

Function `total_power(self)`**:**
- $total\_power = 0$
- **foreach** *device* $d_i \in devices$ **do**
  - $d_i.power = calculate\_power(d_i.type, d_i.state, properties[])$
  - $total\_power = total\_power + d_i.power$
- **return** total_power

Function `step(action)`**:**
- $Execute(action)$
- **if** *properties* $\neq$ *userPreferenes* **then**
  - **if** *userPreferenes.isCountable* **then**
    - $reward = -1 \times \frac{count(userPreferences.unsatisfied)}{count(userPreferences)}$
  - **else**
    - $reward = -1 \times \frac{|V^i_{user} - V^i_t|}{1 + |V^i_{user} - V^i_t|}$
- **else**
  - $reward = 1 - 2 \times \frac{self.total\_power}{MAX\_POWER}$
- $cycle\_length = cycle\_length - 1$
- **if** *cycle_length == 0* **then**
  - $day\_ended = True$
  - $daily\_energy = daily\_energy + total\_power(self)$
- **else**
  - $day\_ended = False$
  - $daily\_energy = 0$
- **return** reward,properties[],power[],states[],total_power

- *calculate_power* function contains the power estimation models for each device in the environment. It returns the value of the power consumption of a specific device at a particular time.
- *total_power* function returns the total power consumption of all devices in the environment at a particular time. When collecting knowledge in 2, all devices with power consumption that can be managed are identified and this particular information is transferred to the environment.
- *reset* function re-initiates the environment to its initial state without changing the observation and the action spaces. It is used at the end of each set of steps that makes up an episode.
- *log* function records events occurring in the environment into a CSV file, in particular, the value of each property, states of devices, power consumption for each device, total power consumption, executed action, and reward value. The log function is showed in algorithm 3.

Algorithm 3: Logging function

```
1
2  Function log(self):
3      foreach device d_i ∈ devices do
4          foreach property p_i ∈ d_i.properties do
5              data_to_save+ = property
6          data_to_save+ = d_i.power
7          data_to_save+ = d_i.state
8      data_to_save+ = env.total_power_consumption
9      data_to_save+ = env.action
10     write(data_to_save)
11     return total_power
12
```

Deep RL agent is based on a feedback loop that takes as an input states and rewards. Therefore, rewards play a critical role in RL by providing a motivation for agents to learn and explore. We define a set of reward equations based on the user satisfaction and the power consumption. The reward function is equal to a rational number bounded between -1 and +1. These three reward equations are presented in (5.1), (5.2), and (5.3).

First, it considers user preferences and makes sure all user preferences are respected. Doing so guarantees the respect of the user preferences before starting to manage power consumption. If any preference is not respected, the reward will be negative between 0 and -1. It is calculated by counting the number of unrespected preferences and dividing it by

the total number of user preferences, as seen in Eq. (5.1), when the user preferences are not measurable.

$$R_t = -1 \times \frac{count(unsatisfied\ user\ preferences)}{count(user\ preferences)} \quad \in [-1, 0] \tag{5.1}$$

However, if the user preferences are measurable (i.e., temperature), the reward is calculated based on the difference between the property value V_t^i at a specific time t and the desired value V_{user}^i , as seen in Eq. (5.2).

$$R_t = -1 \times \frac{|V_{user}^i - V_t^i|}{1 + |V_{user}^i - V_t^i|} \quad \in [-1, 0] \tag{5.2}$$

Once all user basic preferences are respected, Eq. (5.3) is adopted to calculate the reward and ensure a higher reward for each power-aware action. This equation is bounded between -1 and 1. Its value is -1 when the current power consumption is equal to the maximum power consumption and tends to 1 when the power consumption goes down to 0. Reaching 0 as power consumption for the entire environment is difficult due to user preferences but remains possible in some cases.

$$R_t = 1 - 2 \times \frac{P_t^{total}}{P^{max}} \quad \in [-1, 1] \tag{5.3}$$

where: R_t is the reward calculated at time t, P_t^{total} is the total power consumption estimated or measured at time t, and P^{max} is the maximum total power consumption of the environment.

Once the virtual training environment with all its necessary components is generated, the model and the agent are generated and trained using this virtual environment. Algorithm 4 shows the model and agent generation pseudocode. First, states and actions are defined based on the previously generated reinforcement learning environment. Then, the model is built using the *build_model* function. The model is based on a Deep-Q network that takes as an input layer the set of states and returns possible actions in output nodes. It has two hidden dense Keras neural network layers along with a ReLU activation, as seen in figure 5.4. The *build_agent* function specifies the policy and builds the agent based on the previously built model. The agent and model are compiled using the *compile* function, they are later used for training and testing. Finally, the *fit* function trains the agent given the environment and the number of training steps.

Algorithm 4: Deep reinforcement learning model and agent

1: **procedure** CREATERLAGENT($generated\_environmentenv$)
2: $states = env.observation\_space.shape[0]$
3: $actions = env.action\_space.n$
4: $model = build\_model(states, actions)$
5: $agent = build\_agent(model, actions)$
6: $agent.compile(model, actions)$
7: $agent.fit(env, nb\_steps)$
8: **end procedure**

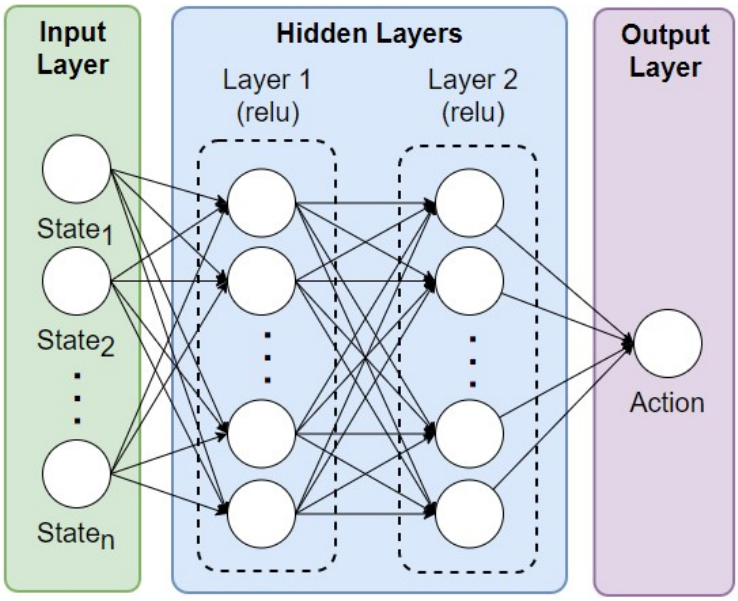


Figure 5.4: Deep neural network layers

At this stage, knowledge of the connected environment is extracted from the ontology. A virtual environment is generated and a RL agent is trained using it. In addition, reward functions are defined in a generic way. Validation of our reasoning will be presented in the next chapter in a smart home environment.

5.5 Summary

In this chapter, we presented our automated energy management framework for CPS based on knowledge and intelligence management components. Our approach allows the representation of knowledge and the generation of virtual learning environments with a spotlight on energy. It aims to help train RL agents in different environments.

First, we reviewed related work of ontologies in connected environments and reinforcement learning environment generation. Then, we presented an energy-oriented extension for ontologies applied to the SAREF ontology. We presented four queries used to collect knowledge from ontology. Furthermore, we developed the algorithms of our RL approach used to generate the RL agent and environment. We also presented different reward functions for different scenarios. Different environments can be generated based on the corresponding knowledge representation.

In the next chapter, we evaluate our approach with two proof-of-concept case studies in a smart home environment. Comprehensive experiments are conducted to validate the approach, algorithms, environment, and model generation. Furthermore, we analyze and discuss the obtained results.

Chapter 6

Validation and Discussions

Contents

6.1 Introduction

With the exponential increasing number of connected and smart devices, the total energy consumption of an environment containing these devices is increasing. These environments are not only composed of devices that interact with each other to accomplish specific tasks, but also involves humans which adds an extra level of complexity to the system. Therefore, evaluations of any energy management approach should be done in a context where devices produce various data types ranging from physically measures values (e.g. humidity) to cyber data (e.g. CPU utilization) and human needs and preferences must be considered.

Our previously presented approach consists of proposing an automated energy management framework based on power estimation, knowledge representation, and reinforcement learning techniques. It aims to reduce power consumption at the holistic level. Having accurate power estimation models for different devices is the first step in overall energy management. Our power modeling and estimation component was validated separately in chapter 4.

The main objective of this chapter is to ensure the efficiency of applying the previously proposed energy management framework. More particularly, we aimed at validating the components related to knowledge and intelligence management (cf. sections 5.3 and 5.4). This chapter aims at guaranteeing that the four components of our approach work together to facilitate energy management while taking into consideration user satisfaction. Therefore, we focus on studying energy consumption and reward values during the validation process. Our approach was evaluated in two separate case studies, both in the context of a smart home. The first case study was conducted in a living room equipped with a variety of devices, while the second case study involved a smart home with a heating system. These two case studies provide a variety of device types, data types, and customizable environments.

In this chapter, we focus on evaluating the entire approach, mainly the knowledge and intelligent management components. In the following, we present the case study descriptions, experimental setup, implementations, and discuss results.

6.2 Case Studies Descriptions

Smart home environments are scalable and contain devices that are quite heterogeneous having different types, interfaces, vendors, etc. In addition, these environments are dynamic as changes can occur to existing devices such as software updates or hardware updates that lead to a change in power consumption or location changes that impact the knowledge of the environment. In the following, we detail the description and experimental setup for each of the two study cases in the context of a smart home.

6.2.1 Living Room (Case Study I)

As seen in Figure 6.1, the first case study is defined as a living room of a smart home with several devices. A led bulb controls its brightness (LedBrightness) and color (LedColor). A TV controls its brightness (TVBrightness). A Raspberry Pi measures its CPU utilization with a mounted light sensor that measures the room brightness. The LED bulb, TV, and raspberry pi are considered power consumers and each has a power estimation model that gives their power consumption in real-time. A door sensor checks if the door is opened or closed without being able to change its state. The door sensor is not considered as a power consumer because its power consumption is minimal and unchangeable. LedBrightness has a minimum value of 0 and a maximum value of 100 and can be incremented or decremented by 1% per minute to ensure a smooth light transition that is not annoying for the users. LightColor can have one of the following values: red, blue, green, and white. Devices states

are usually on, off, or sleep. However, they are not limited to these states and can take many other forms. For the living room scenario, user preferences are generated based on a smart residential load simulator for energy management presented in [169]. It simulates different loads of configurable home appliances (i.e., lighting, water heater, stove, refrigerator, and dishwasher) using mathematical models. They are manually adjusted to add randomness that may occur from one day to another, between weeks, and during weekends.

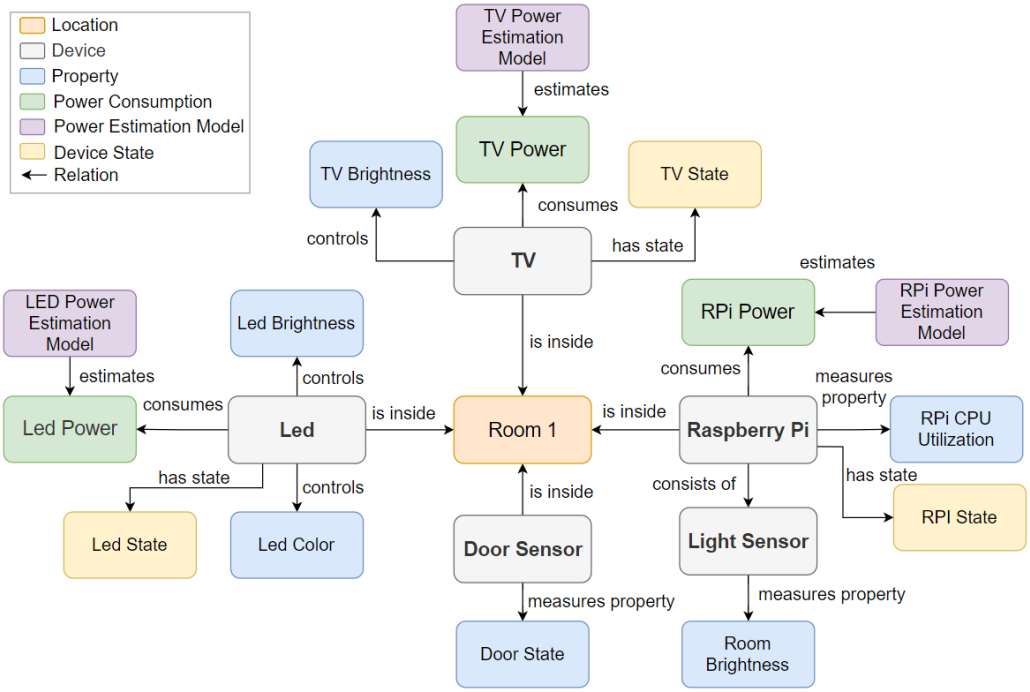


Figure 6.1: Devices, properties, and relations representation (case study I)

6.2.2 HVAC (Case Study II)

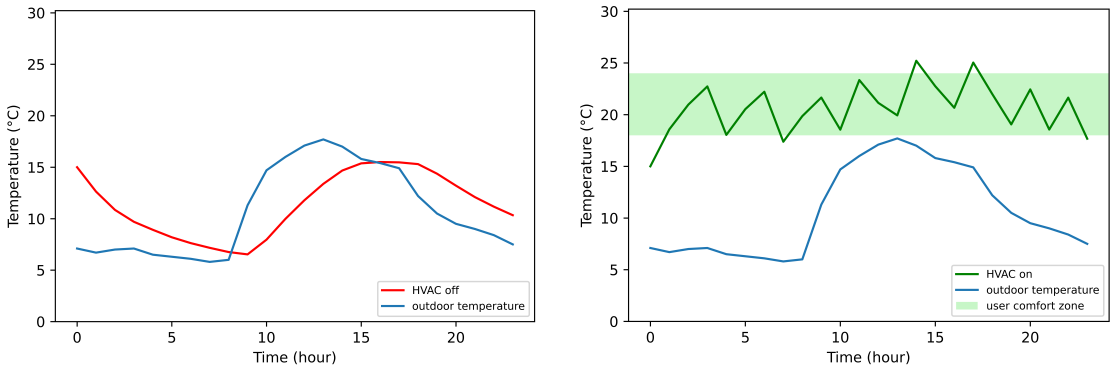
Another case study is conducted with different devices in a different context. It is based on a HVAC system in a smart home. The room where the HVAC is mounted is equipped with an indoor temperature sensor and a presence sensor. In addition, a temperature sensor is positioned outside the room to capture the outdoor temperature. We model the environment using the previously proposed extension of the SAREF ontology as done in the living room case study. Then, we launch the agent generator to build the environment and the reinforcement learning agent. To simulate indoor temperature dynamics in the environment, the exponential decay model in Eq. (6.1) is adopted [170].

$$T_{t+1}^{in} = \varepsilon \times T_t^{in} + (1-\varepsilon) \times (T_t^{out} \pm \frac{\eta_{hvac}}{A} \times e_t) \quad (+ : heating, - : cooling) \tag{6.1}$$

where T_t^{in} denotes room internal temperature during time t, T_t^{out} is the real outside temperature during time t, $\varepsilon \in [0, 1]$ is a thermal time constant, η_{hvac} is the thermal conversion efficiency (heating) or the coefficient of performance (cooling), A is the thermal conductivity that varies based on the insulation of a space, and $e_t \in [0, e_{max}]$ is the electric HVAC system power input at time t (It ranges between 0 when the HVAC system is turned off and e_{max} the maximum power consumption of the HVAC system). We consider $\varepsilon = 0.7$, $\eta_{hvac} = 2.5$, $A = 252\, w/°C$ based on the study conducted in [168], $e_t = \frac{U_t}{100} \times e_{max}$ where U_t is the percentage of utilization of the HVAC system, and $e_{max} = 2000\, w$.

Eq. (6.1) was widely adopted in previous research such as in [168, 171, 172, 173, 174]. This experiment used the outdoor temperature dataset proposed by USCRN/USRCRN [175] due to its accuracy, low error rates, and high quality of collected data. The dataset is cleaned and filtered mainly to remove missing data indicated by the lowest possible integer, i.e., $-9999, 0$. The used dataset was collected in Santa Barbara, California, the USA in 2021. Occupant temperature preferences are extracted from the study conducted in [176]. Temperature preferences range between $20°C$ and $24°C$ during the day. They range between $18°C$ and $22°C$ during the night when the home occupant is sleeping.

Figure 6.2 shows the outside temperature of a sample day in January 2021. In addition, Figure 6.2(a) shows the calculated indoor temperature with the HVAC turned OFF. In Figure 6.2(b), two conditions are added to guarantee that the inside temperature respects user preferences. It illustrates a real-life scenario where the HVAC is turned ON and OFF according to the room temperature.



(a) HVAC turned OFF

(b) HVAC turned ON

Figure 6.2: Outdoor and indoor temperature samples

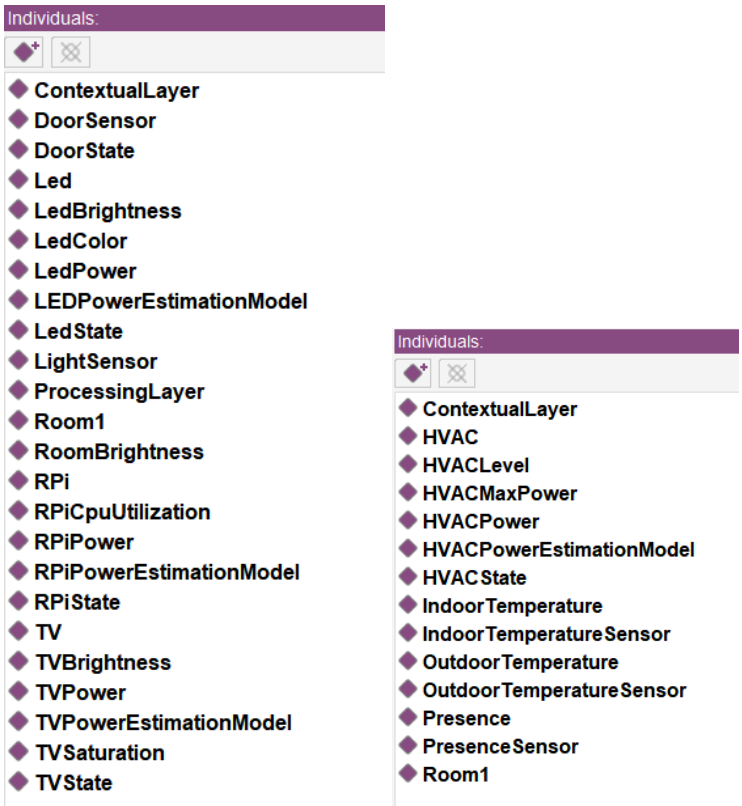
6.3 Implementation

The implementation of these two case studies is divided into two parts. Firstly, the knowledge belonging to each case study is represented as individuals belonging to the extended

energy-oriented ontology. Secondly, the knowledge is queried and a RL environment is generated. Then, an agent is trained using the generated environment.

The experiments are conducted on a Dell Precision 7530 workstation equipped with an Intel i7 8th generation 2.20 GHZ processor (64 bits), 16 GB of RAM, and running Windows 10, Python version 3.7, TensorFlow version 2.7.0, Keras-RL version 2.8.0, OpenAI Gym version 0.25.2 and version 5.5.0 of protégé.

6.3.1 Knowledge Management Implementation



(a) Living Room Scenario Individuals (b) HVAC Scenario Individuals

Figure 6.3: Knowledge representation as individuals

In order to represent the knowledge found in the environment, we use the extended energy-oriented ontology presented in section 5.3. We create a set of individuals for each of the defined case studies. Figure 6.3(a) and figure 6.3(b) define each device, controllable property, measurable property, and power estimation model present respectively in the living room scenario and the HVAC scenario. In addition, the relations between these concepts are identified and set.

6.3.2 Intelligence Management Implementation

The data of individuals is collected and a source code is generated for each of the use cases. First, each of the SPARQL queries (cf. section 5.4.2) is executed resulting in a set of JSON responses that are transformed into Python Pandas DataFrames for easier handling and processing. It allows the generator to observe the devices present in the environment and their corresponding sensors and actuators for future analysis and source code generation. Tables 6.1,6.2,6.3, and 6.4 present the DataFrames that result from the execution of the SPARQL queries of case study I. In the following, we present the implementation of case study I. Appendix A presents an example of a raw SPARQL queriy result regarding case studies I.

Table 6.1: Actions DataFrame

	Device	Property	DataType	Range
0	Led	LedBrightness	int	MinMaxStep:0,100,1
1	Led	LedColor	string	List:Red,Blue,Green,White
2	TV	TVBrightness	int	MinMaxStep:0,100,10
3	TV	TVSaturation	int	MinMaxStep:0,100,5

Table 6.2: Power Estimation Models DataFrame

	Power	DataType	PEM_value	State	Device
0	LedPower	float	Philips_LED_9	LedState	Led
1	RPiPower	float	Raspberry Pi 3 Model B Rev 1.2	RPiState	RPi
2	TVPower	float	SonyTV_32	TVState	TV

Table 6.3: Properties DataFrame

	Property	DataType	Range	Value
0	LedBrightness	int	MinMaxStep:0,100,1	50
1	LedColor	string	List:Red,Blue,Green,White	White
2	TVBrightness	int	MinMaxStep:0,100,10	90
3	TVSaturation	int	MinMaxStep:0,100,5	50
4	DoorState	boolean	Bool:True,False	true
5	RoomBrightness	int	MinMaxOnly:0,100	60
6	RPiCpuUtilization	float	MinMaxOnly:0,100	80.0

Table 6.4: States DataFrame

	Device	StateType	Range	Value	State
0	Led	OnOffState	OFF,ON	ON	LedState
1	TV	OnOffState	OFF,ON	ON	TVState
2	RPi	MultiLevelState	OFF,SLEEP,ON	ON	RPiState

Then, we copy an empty template of the environment that contains the necessary libraries, functions, and code structure. Special comments are used as placeholders in the source code; they are used to indicate the starting and the ending of each dynamic code, as seen in source code 2 (i.e., *#START_INIT* and *#END_INIT*).

```
def __init__(self):
    le = preprocessing.LabelEncoder()
    self.firstStep=True
    #START_INIT
    self.action_space = Discrete(18)
    HIGH = np.array([100,3,100,100,True,100,100,1,1,2],
    ↪  dtype=np.float32)
    LOW = np.array([0,0,0,0,False,0,0,0,0,0], dtype=np.float32)
    self.observation_space = Box(LOW, HIGH)
    self.state = None
    le.fit(['Red', 'Blue', 'Green', 'White'])
    LedColorMapping = dict(zip(le.classes_, le.transform(['Red',
    ↪  'Blue', 'Green', 'White'])))
    self.LedBrightness=50
    self.LedColor=3
    self.TVBrightness=90
    self.TVSaturation=50
    self.DoorState=True
    self.RoomBrightness=60
    self.RPiCpuUtilization=80.0
    le.fit(['OFF', 'ON'])
    LedStateMapping = dict(zip(le.classes_, le.transform(['OFF',
    ↪  'ON'])))
    self.LedState=1
    le.fit(['OFF', 'ON'])
    TVStateMapping = dict(zip(le.classes_, le.transform(['OFF',
    ↪  'ON'])))
    self.TVState=1
    le.fit(['OFF', 'SLEEP', 'ON'])
    RPiStateMapping = dict(zip(le.classes_, le.transform(['OFF',
    ↪  'SLEEP', 'ON'])))
    self.RPiState=2
    self.PEM=["LedPower","RPiPower","TVPower"]
    self.LedPower=0
    self.RPiPower=0
    self.TVPower=0
    self.propertyList=["LedBrightness", "LedColor",
    ↪  "TVBrightness", "TVSaturation", "DoorState",
    ↪  "RoomBrightness", "RPiCpuUtilization", "LedState",
    ↪  "TVState", "RPiState"]
    #END_INIT
    # Set episode length
    self.cycle_length = CYCLE_LENGTH
    self.episodeEnergy=0
```

Source Code 1: Initialization generated source code

```
def __init__(self):
    le = preprocessing.LabelEncoder()
    self.firstStep=True
    #START_INIT
    #END_INIT
    # Set episode length
    self.cycle_length = CYCLE_LENGTH
    self.episodeEnergy=0
```

Source Code 2: Placeholders of the RL environment initialization function

The dynamic generated source code of the initialization function of the RL environment is presented in source code 1. Likewise, the source code sections of the actions, rewards, states, power consumption, and the reset are generated. The generated source code for case studies II is available in Appendix B. After the environment is generated, an agent is also cloned from an agent template. The agent is then trained using the generated environment. In the next section, we present the results of training the agent using our generated environment.

6.4 Results

A prototype of the proposed power management approach has been developed to reduce power consumption in smart connected environments. It is based on a continuous process of monitoring, analyzing, making and executing adaptive decisions. In particular, the following two application case studies have been developed, simulated, and experimented in a smart home for validation purposes.

6.4.1 Living Room Results (Case Study I)

A first experiment has been conducted for 70 episodes (100k steps). A step represents one iteration accomplished by the reinforcement learning algorithm until it reaches its final goal. An episode is composed of a defined series of steps. At the beginning of each episode, the environment resets to its initial state and the agent's reward is set to zero. Each episode lasts for 24 hours and one step is executed per minute (1440 steps per episode). Once per minute, metrics and states of devices are monitored. Then, the agent analyses them and chooses adaptive actions to be executed per device. This process is influenced by the MAPE-K feedback loop but no long-term plans are prepared ahead of time and actions are spontaneous. However, the agent has the possibility to execute a neutral action that keeps all devices in their current state. User preferences for the 28 days are defined (i.e., led state, led brightness, led color, TV brightness, and TV state), in addition, we define the user state that can be sleeping, awake, and away.

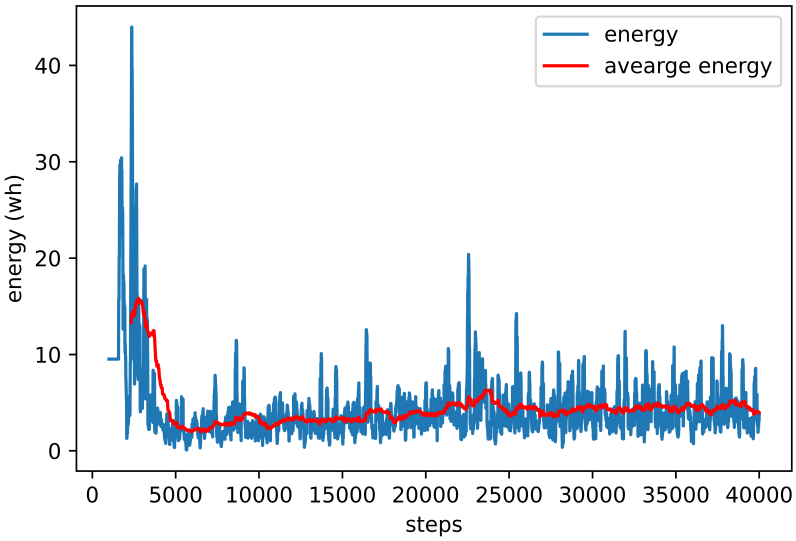
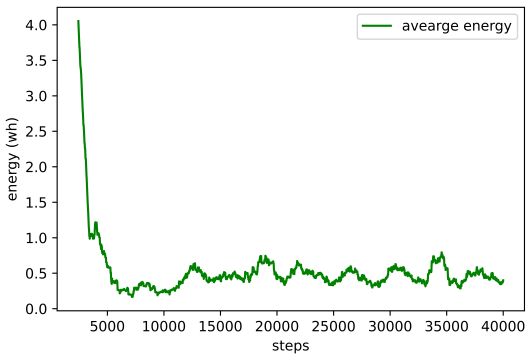
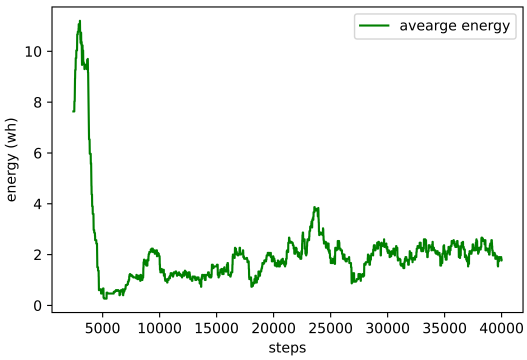


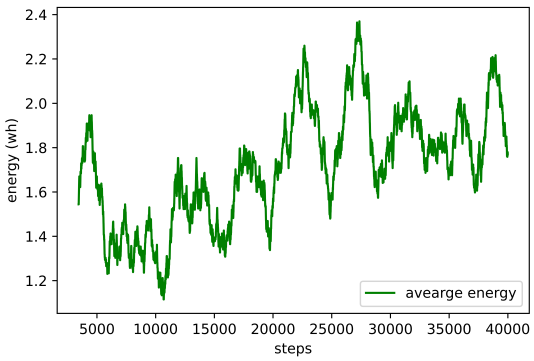
Figure 6.4: Energy consumption per step (case study I)



(a) Led bulb energy consumption



(b) TV energy consumption



(c) Raspberry Pi energy consumption

Figure 6.5: Energy consumption per device per step (case study I)

Figure 6.4 shows the total energy consumption of the environment per step and the average per day. Figure 6.5 shows the average energy consumption of the three devices we are managing (i.e., led, TV, and raspberry pi). It highlights that the energy consumption of the led bulb and the TV have decreased after a few hundred steps. However, the energy consumption of the RPi tends to increase because the user preferences require it to stay turned on. In such scenario, energy has converged after a brief period of time (4 days), and the reward has passed from a negative value to a positive one meaning that the user preferences are respected and that the power consumption is at its lower accepted values, as seen in figure 6.6. This study case shows that the proposed framework can deal with different types of devices, properties, and actions related to these devices. It validates that this approach can be used to represent knowledge, generate reinforcement learning environments, and train agents. In addition, it confirms the ability to improve energy efficiency and savings in the long term.

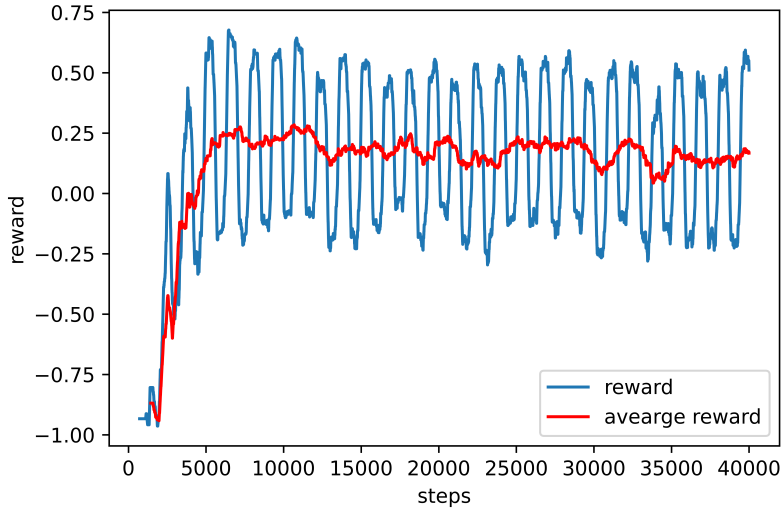


Figure 6.6: Reward point per step (case study I)

6.4.2 HVAC Results (Case Study II)

A second experiment was conducted in the HVAC environment for 596 episodes (100k steps). Each episode has lasted one week with a frequency of one step executed each hour (168 steps per episode). User preferences of the minimum and maximum accepted temperatures are defined for an entire week. Results show an increase in the reward per episode, therefore, the user satisfaction and the power consumption, as seen in Figure 6.8. User preferences vary between the day and the night. However, for the purpose of simplicity, Figure 6.7 graphs the user comfort zone between the lower and higher accepted values. Figure 6.7 shows the room temperature per step and the average room temperature per

episode. Results show that in the first part of the training (before the 20 000^{th} step), the indoor temperature is under the user preferences and converges with time to respect user preferences. Therefore, validating the ability of the proposed reward function to lead to a better user experience. Initial room temperature is randomized for each new episode ranging between $12°C$ and $18°C$, and the initial HVAC level is a random value between 30% and 70%.

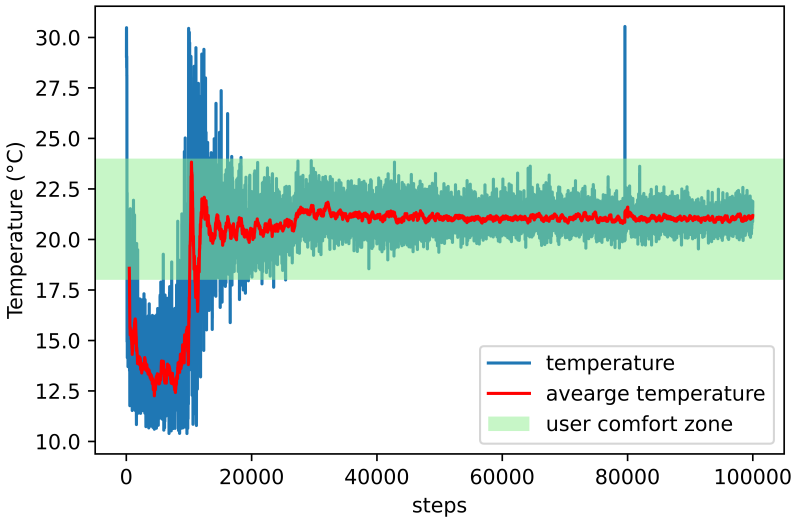


Figure 6.7: Indoor temperature evolution (case study II)

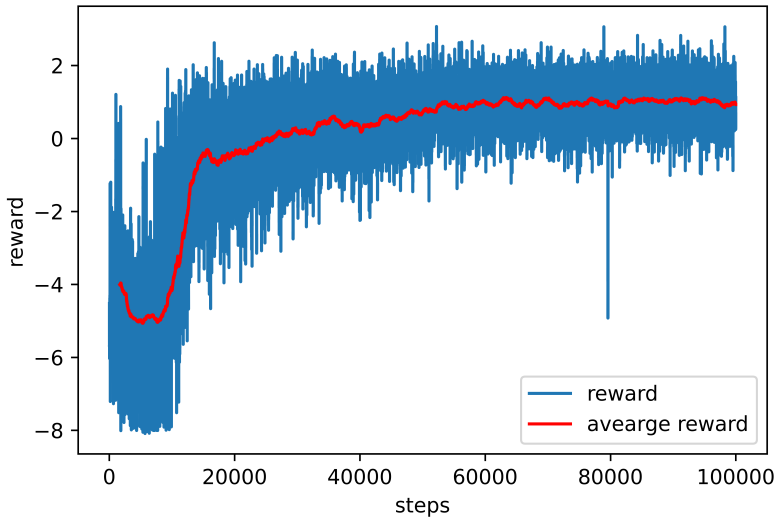


Figure 6.8: Reward convergence process (case study II)

Figure 6.9 highlights that the energy consumption of the HVAC system increases with time and then stabilizes. This is due to the priority of user preferences over energy consumption. In the following, we compare the total energy consumption of our management approach with other traditional ones.

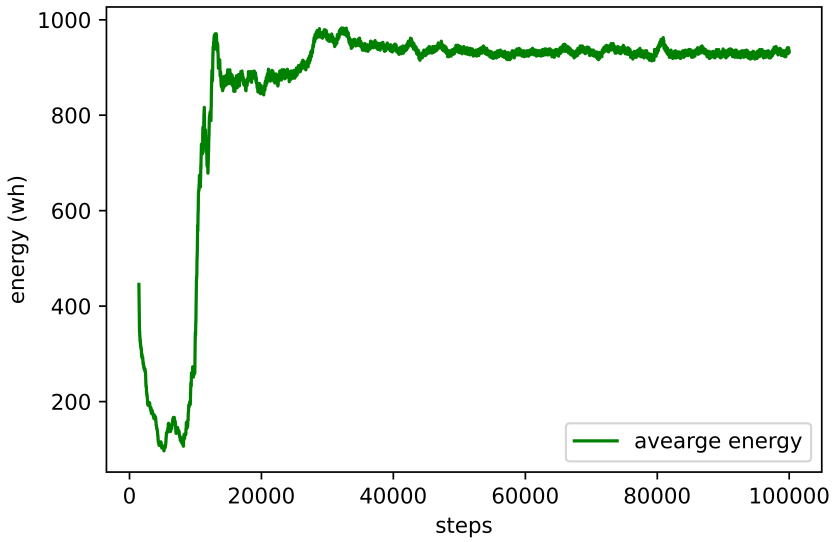


Figure 6.9: Energy consumption of the HVAC system (case study II)

To compare the energy consumption of the proposed approach to other control values, we identify three different baselines:

- *Baseline 1:* Represents the results of the proposed automated energy management approach, proposed in this thesis, including the knowledge, environment and model generation, and training of the agent. It also considers user preferences and presence.
- *Baseline 2:* Represents an environment where user preference indoor temperature is set to $21°C$. This value is obtained by calculating the average temperature between the minimum and maximum accepted values. It was represented previously in Figure 6.2(b). The HVAC is turned ON whenever the room temperature goes below $21°C$ and turned OFF when it outpaces this value. The obtained values are based on the captured indoor and outdoor temperature readings.
- *Baseline 3:* Represents an environment with a traditional electric heater that stays ON all the time without capturing and considering any of the indoor or outdoor temperatures.

Figure 6.10 shows the mean value of energy for each of the previously defined baselines. The results show that the proposed method has a lower energy consumption compared

to the other two traditional methods. Baseline 1 energy consumption equals 932,82 wh, while baselines 2 and 3, respectively, are 1083,3 wh and 2000 wh. We consider that the model has already converged before step number 40 000 and energy consumption is almost stable. Therefore, in the 1^{st} baseline, the average energy consumption is calculated with data corresponding to steps higher than 40 000.

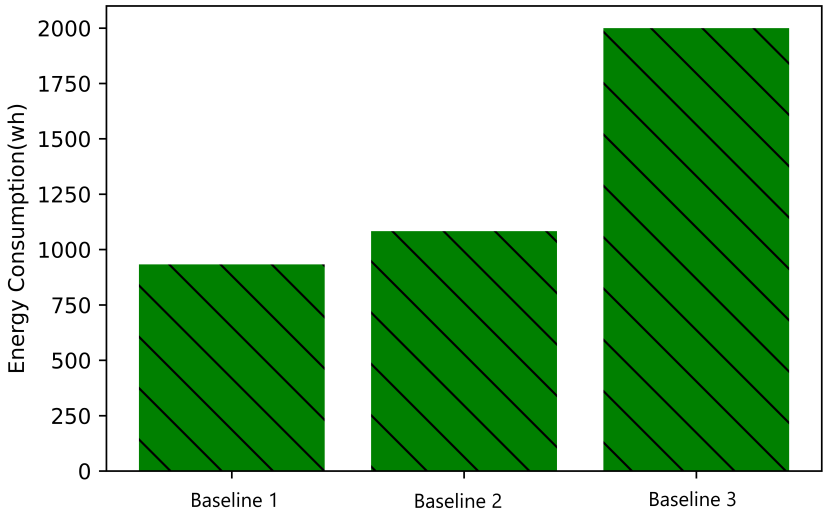


Figure 6.10: Mean value of HVAC energy consumption (case study II)

6.5 Discussion

The results indicate that automated power management is done using the proposed approach that leads to a reduction in power consumption. In the following, we discuss the result obtained in each phase of the automated energy management framework implementation.

6.5.1 Interpretations

The proposed approach in this thesis confirms its capability to answer different non-functional requirements, in addition to offering an efficient energy management system as the primary functional requirement.

In the first place, the results presented in section 6.4 highlight a significant decrease in power consumption while respecting user preferences in both case studies. In addition, our proposal performs well given the low convergence period (4 days for the case study I), the high user satisfaction (the temperature is maintained in the comfort zone), and the

energy consumption decrease. From these results, it is clear that the reward functions are well adapted as feedback to the agent that has the two main goals of facilitating energy optimization and user comfort.

The use of ontologies as a knowledge representation guarantees high flexibility and reusability because we could easily map a large diversity of real scenarios using the provided semantics. The knowledge management component also ensures that the framework is extensible for representing new knowledge concepts. The implementation in two use cases, with completely different devices, confirms the general purpose of our approach because the code generation is done in an automated way allowing the adaptation of the scenario to the framework. It proves the possibility of generating customizable environments with various devices. A high level of autonomy is also guaranteed because of the set of actions that are accomplished in real time, based on the changes and states of devices. The approach also showed the capability of dealing with different metrics that correspond to different scenarios and layers of CPS, therefore, we conclude that the approach is extensible on the level of a smart home. In addition, the reliability of the approach can be illustrated by its adaptation to human behavior to reach the highest levels of user satisfaction. Moreover, the framework is easy to implement because of well-described knowledge representations and straightforward intelligence management techniques and algorithms.

Compared to related work, our proposal meets the functional and non-functional requirements defined in chapter 3.

6.5.2 Limitations

Although Our automated energy management framework is established as an efficient way to manage energy consumption, it is important to acknowledge its limitations.

The proposed framework is based on a Markov decision process where a direct future action fully depends on the current states of devices and does not require any knowledge of the past. Moreover, the actions are spontaneous and no long-term plan is prepared ahead of time.

Another limitation in our energy management framework is that a single agent is responsible for two distinct purposes (reducing power consumption and increasing user satisfaction). However, multi-agent RL algorithms are emerging as solutions for complex environments with more than one optimization problem. Multiple learning agents can coexist in a shared environment where each agent is motivated by its own reward function, and accomplishes actions that boost its objectives (e.g., energy efficiency, user comfort, cost reduction).

In addition, one-room scenarios are identified and evaluated in the previous sections. However, different rooms constitute a smart home. Therefore, we consider the necessity

of imagining and testing the solution with more case studies to evaluate scalability dimensions. Multi-layer metrics are important to optimize some low-power consumption devices. However, some layer-level optimizations are not significant compared to the energy optimization that can occur on the device level (e.g., reducing the power consumption of a CPU in a HVAC system is not significant compared to the energy used by the system itself).

Although widely accepted, it suffers from some limitations due to the difficulty of calculating user comfort to compare it with other solutions.

6.6 Summary

In this chapter, we evaluated the automated energy management framework with two proofs-of-concept case studies in the context of a smart home: (i) a living room with a variety of devices and (ii) a smart home with a heating system. We mainly validated the knowledge management and intelligence management components. Comprehensive experiments were conducted to validate the approach, algorithms, environment, and model generation. Finally, we analyzed and discussed the obtained results.

We implemented the two case studies by representing individuals belonging to the extended energy-oriented ontology. Then, knowledge collection queries were performed and the source code of the RL environments and agents were generated. The agent was trained using the generated environments. The results of these experiments highlight a significant decrease in power consumption while respecting user preferences in both case studies. In addition, this proposal performs well given the low convergence period, the high user satisfaction, and the energy consumption decrease. Discussions show that the previously identified functional and non-functional requirements (cf. chapter 3) are reached.

In the next chapter, we conclude and summarize the results of our research project, outline our contributions, and highlight perspectives for future research directions.

Chapter 7

Conclusion and Perspectives

Contents

In this chapter, we summarize and outline our proposal in response to the challenges and research objectives. We also lay out the perspectives and highlight new future research directions. Finally, the list of our publications produced during this research project are presented.

7.1 Summary of the Dissertation

In this thesis, we presented a framework for energy management at a holistic level for Cyber-Physical Systems (CPS). We motivated our work using two scenarios illustrated in the smart home domain: a living room with a variety of devices (first scenario) and a smart home with a heating management system (second scenario). Our framework is automated, flexible, and generic. In addition, it can deal with different devices and metrics. The proposed solution is developed based on four main components: power modeling and estimation, knowledge management, intelligence management, and data repository. The integration of each of these components together met our research objectives and led to high levels of energy efficiency and user satisfaction.

In **Chapter 1**, we highlighted the importance of energy management as a topic of interest in the context of CPS. Then, we focus on the objectives of this thesis of (i) identifying

and representing knowledge and metrics affecting energy consumption; (ii) estimating real-time power consumption of any device in the environment; and (iii) identifying actions that can be applied to reduce the total energy consumption. We presented a generic smart home scenario that illustrates the motivation behind this work and the challenges we addressed. Finally, we glanced at the state of the art before proposing the basis of our approach.

In **Chapter 2**, we presented background of the context of our work. We first introduced CPS and gathered different architectural modeling. Then, we presented a systematic review of the existing energy management approaches and compared them based on several criteria. Our findings showed that state-of-the-art research lacks the variety of studied metrics not covering all potential energy drain sources in a system. Literature solutions deal with limited layers of the CPS in a restricted number of devices mainly due to the challenges of exchanging data and heterogeneous environments. In addition, most solutions are not autonomous. However, some could be considered to have a limited degree of autonomy. Literature solutions are also mostly domain specific. Their flexibility is highly limited to the devices and the environment in which they were developed. This state of the art justified our research goals for providing a solution.

In **Chapter 3**, we presented the requirements analysis and the solution architectural design, using a formal model-based software engineering methodology, in response to the lack of such approaches in the literature. First, we defined functional and non-functional needs, as well as, different architectural levels to model the system (i.e., operational, system, and logical). Then, we defined the four components of the system design:

- A power modeling and estimation component as a first step towards having better visibility on energy drains that enables energy management actions to be accomplished (in response to the **RO1** and **RO2**);
- A knowledge management component providing a common representation of different concepts and the relations between them using an energy-oriented extension for ontologies (in response to the **RO1** and **RO3**);
- An intelligence management component targeting the energy management issue using RL techniques and the previous two components (in response to the **RO3**);
- A data repository allowing the storage of collected metrics, applied actions, power consumption, and power estimation models.

Finally, we justified some of our development choices such as the use of ontologies and reinforcement learning algorithms.

In **Chapter 4**, we presented the implementation of the power modeling and estimation component. We implemented a proof-of-concept client and server to automate the generation of power models for various smart home devices (i.e., Raspberry Pi devices and LED bulbs). It allowed these models to be updated, extended, improved, and shared. Our component provided up-to-date and accurate power estimation models, compared to related work, with error rates as low as 0.33% and up to 7.81% for linear models, and 0.3% up to 3.83% for polynomial models. Furthermore, we found an important impact of device revisions and OS architectures on power consumption. Therefore, we recommended using power models generated specifically per device revision and architecture. Finally, we analyzed the influence of this component on the total consumed energy and found that the power consumption can be estimated with negligible overhead in real time.

In **Chapter 5**, we presented the implementation of the knowledge management and intelligence management components. First, we presented an energy-oriented extension for ontologies applied to the SAREF ontology. It defined various concepts needed for global knowledge representation of complex connected environments. Then, we defined four essential queries used by the RL environment generator to collect knowledge from ontology. In addition, we presented the algorithms used to generate RL agents and environments in an automated manner and find the best energy-efficient actions. Our solution showed the possibility to generate various RL environments and agents based on the corresponding knowledge representation. Finally, we defined generic reward functions that consider both energy consumption and user preferences.

In **Chapter 6**, we evaluated the automated energy management framework with two proofs-of-concept case studies in the context of a smart home. We mainly aimed to validate the knowledge management and intelligence management components. The implementation of these case studies showed a convenient RL environment generation based on the defined knowledge. The results of the accomplished experiments highlighted a significant decrease in power consumption while respecting user preferences in both case studies. In addition, this proposal performs well given the low convergence period, the high user satisfaction, and the energy consumption decrease.

7.2 Perspectives

This thesis opens the doors to a variety of new research directions that we can apply in the future to further enhance and validate our approach. In the following, we discuss the most important points to address based on the identified open issues that deserve further research.

Cover more types of devices in real-world environments

Future research is needed to evaluate the behavior of the proposed method in a real-world environment with real physical sensors and actuators when dealing with communication protocol heterogeneity, missing data, and latency. First, we plan to study and model the power consumption of additional hardware components of connected devices, such as motors, resistors, networks (e.g., WiFi and Ethernet), and interfaces (e.g., USB). We would like to study the impact of additional software and hardware metrics on power consumption, as this might help improve the accuracy of the generated power models. In addition, we plan to expand our implementation to cover more single-board devices (e.g., new embedded devices) and additional smart home devices. The idea is to automatically support any future device with a universal power manager.

Implement and evaluate our solution in larger scales and more heterogeneous scenarios

In the future, further investigation is needed to evaluate our solution in various spatio-temporal arrangements. In particular, we plan to study the impact of such an approach over a longer period and the effect of changes in user behavior with time. Furthermore, it will be important for future research to investigate extreme cases such as during the COVID-19 pandemic, and all the changes it could bring to user behavior. In addition, we plan to test our approach with more data collected from different locations that vary from one region to another (e.g., temperature). We also plan to expand the implementation to cover additional devices on a larger scale (e.g., include many rooms in the same home, and many homes in a building). Therefore, the spacial aspect presented in our ontology should be expanded and aligned with other ontologies such as the Building Topology Ontology [145].

Enrich and personalize reinforcement learning models

In this thesis, we used the same hidden layers for the RL deep neural network and different input and output layers. However, we argue that unique neural network layers could be generated for each environment. In addition, our solution is based on a single agent responsible for energy optimization and user comfort. Multi-agent RL algorithms can be

used as solutions for complex environments with more than one optimization problem. It allows the easy consideration of additional growing concerns such as the cost of electricity. Therefore, further investigation is needed to integrate multi-agent RL algorithms in such solutions.

Investigate deeply the user behavior

In this thesis, we considered the preferences fixed by the user ahead of time. More studies need to address different user behavior, in particular, the potential rebound effect. Environments can also have many users with different preferences. Therefore, we find it interesting to study conflict resolutions in response to these scenarios. In addition, we plan to integrate a visualization and notification component that shows the users their energy consumption in real time. Besides its informational purposes, it shows users the impact of each action on the total energy consumption just so they become aware of how to act in environments with no intelligent energy management systems. Finally, develop methods that quantify user comfort to compare our approach with other solutions.

Embed the common representation of each device to its firmware

Nowadays, IoT and home automation devices provide limited information about their capabilities in terms of collecting data and accomplishing actions. Home automation companies need to study low-level exchange frames to integrate each device into their solution (e.g., decode LoRaWAN payload). Therefore, adopting the use of the proposed ontology extension to describe each of the devices allows their integration easily for automation purposes. Its use is not limited to energy optimization management systems but guarantees higher levels of flexibility, scalability, and integrability. Therefore, we must develop and embed lightweight versions of this knowledge model to the firmware devices by default and load it with appropriate descriptions in terms of touch-points (sensors and actuators). In addition, different vendors need to agree on standardizing this kind of knowledge representation to be more beneficial.

7.3 Publications

Journals

- **Houssam Kanso**, Adel Noureddine, Ernesto Exposito. *An Automated Energy Management Framework for Smart Homes*. Submitted to publication, Journal of Ambient Intelligence and Smart Environments. Under Review

- **Houssam Kanso**, Adel Noureddine, Ernesto Exposito. *Automated Power Modeling of Computing Devices: Implementation and Use Case for Raspberry Pis*. Sustainable Computing: Informatics and Systems 37 (2023) 100837. 10.1016/j.suscom.2022.100837. hal-03912723

- **Houssam Kanso**, Adel Noureddine, Ernesto Exposito. *A Review of Energy Aware Cyber-Physical Systems*. ACM Transactions on Cyber-Physical Systems (2023) 1–42. 10.1080/23335777.2022.2163298. hal-03912724

Conferences

- Adel Noureddine, Matias Martinez, **Houssam Kanso**, Noëlle Bru. *Is well-tested code more energy efficient?*. 11th Workshop on the Reliability of Intelligent Environments, Jun 2022, Biarritz, France. hal-03635797

- Adel Noureddine, Matias Martinez, **Houssam Kanso**. *A Preliminary Study of the Impact of Code Coverage on Software Energy Consumption*. 2nd International Workshop on Sustainable Software Engineering (SUSTAINSE), Nov 2021, Melbourne, Australia.10.1109/ASEW52652.2021.00057 . hal-03380602

- Kamar Kesrouani, **Houssam Kanso**, Adel Noureddine. *A Preliminary Study of the Energy Impact of Software in Raspberry Pi devices*. 29th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2020, Bayonne, France. hal-02936861

Bibliography

[1] David Mercer. Global Connected and IoT Device Forecast Update. Technical Report May, Strategy Analytics - Research, Experts, and Analytics, 2019.

[2] Dave Evans. The Internet of Things: How the Next Evolution of the Internet Is Changing Everything. Technical Report April, Cisco Systems, 2011.

[3] IEA.org [Online]. Electricity Information – Analysis. `https://www.iea.org/reports/electricity-information-2019`, [Accessed on 2022 Oct 17].

[4] Lorenz M; Hilty, Wolfgang; Lohmann, and Elaine M Huang. Sustainability and ICT – An overview of the field. *Notizie di Politeia*, 27(104):13–28, 2011.

[5] Abram Hindle. Green mining: a methodology of relating software change and configuration to power consumption. *Empirical Software Engineering*, 20(2):374–409, apr 2015.

[6] José Antonio Esparza Isasa, Peter Gorm Larsen, and Finn Overgaard Hansen. Energy-Aware Model-Driven Development of a Wearable Healthcare Device. In Michaela Huhn and Laurie Williams, editors, *Huhn M., Williams L. (eds) Software Engineering in Health Care. SEHC 2014, FHIES 2014. Lecture Notes in Computer Science*, volume 9062, pages 44–63. Springer, 2017.

[7] Mohammad Abdullah Al Faruque and Fereidoun Ahourai. A model-based design of Cyber-Physical Energy Systems. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 97–104, Singapore, jan 2014. IEEE.

[8] Jiafu Wan, Hehua Yan, Di Li, Keliang Zhou, and Lu Zeng. Cyber-physical systems for optimal energy management scheme of autonomous electric vehicle. *Computer Journal*, 56(8):947–956, 2013.

[9] Shaolin Wang, Guiqing Zhang, Bin Shen, and Xiuying Xie. An Integrated Scheme for Cyber-physical Building Energy Management System. *Procedia Engineering*, 15(0):3616–3620, 2011.

[10] Paul Bogdan and Radu Marculescu. Towards a Science of Cyber-Physical Systems Design. In *2011 IEEE/ACM Second International Conference on Cyber-Physical Systems*, pages 99–108, Chicago, apr 2011. IEEE.

[11] Shuaiyin Ma, Yingfeng Zhang, Jingxiang Lv, Haidong Yang, and Jianzhong Wu. Energy-

cyber-physical system enabled management for energy-intensive manufacturing industries. *Journal of Cleaner Production*, 226:892–903, jul 2019.

[12] Hanno Georg, Sven Christian Muller, Christian Rehtanz, and Christian Wietfeld. Analyzing Cyber-Physical Energy Systems:The INSPIRE Cosimulation of Power and ICT Systems Using HLA. *IEEE Transactions on Industrial Informatics*, 10(4):2364–2373, nov 2014.

[13] Davide Basile, Felicita Di Giandomenico, and Stefania Gnesi. Statistical model checking of an energy-saving cyber-physical system in the railway domain. *Proceedings of the ACM Symposium on Applied Computing*, Part F1280:1356–1363, 2017.

[14] Wei Wu, Wenjia Li, Deify Law, and Woonki Na. Improving Data Center Energy Efficiency Using a Cyber-physical Systems Approach: Integration of Building Information Modeling and Wireless Sensor Networks. *Procedia Engineering*, 118:1266–1273, 2015.

[15] Bin Zhou, Wentao Li, Ka Wing Chan, Yijia Cao, Yonghong Kuang, Xi Liu, and Xiong Wang. Smart home energy management systems: Concept, configurations, and scheduling strategies. *Renewable and Sustainable Energy Reviews*, 61:30–40, 2016.

[16] Luca Parolini, Niraj Tolia, Bruno Sinopoli, and Bruce H. Krogh. A cyber-physical systems approach to energy management in data centers. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems - ICCPS '10*, page 168, New York, New York, USA, 2010. ACM Press.

[17] Wei Wang, Tianzhen Hong, Nan Li, Ryan Qi Wang, and Jiayu Chen. Linking energy-cyber-physical systems with occupancy prediction and interpretation through WiFi probe-based ensemble classification. *Applied Energy*, 236(February):55–69, 2019.

[18] Luis Sánchez, Ignacio Elicegui, Javier Cuesta, and Luis Muñoz. On the energy savings achieved through an internet of things enabled smart city trial. In *Proceedings of the 2014 IEEE International Conference on Communications (ICC)*, pages 3836–3841, Sydney, NSW, 2014. IEEE.

[19] Jinsoo Han, Chang-Sic Choi, and Ilwoo Lee. More efficient home energy management system based on ZigBee communication and infrared remote controls. *IEEE Transactions on Consumer Electronics*, 57(1):85–89, feb 2011.

[20] Kwang-soon Choi, Yang-keun Ahn, Young-Choong Park, Woo-chool Park, Hae-Moon Seo, Kwang-mo Jung, and Kyeung-hak Seo. Architectural Design of Home Energy Saving System Based on Realtime Energy-Awareness. In *Proceedings of the 4th International Conference on Ubiquitous Information Technologies & Applications*, pages 1–5, Fukuoka, Japan, 2009. IEEE.

[21] Man Lin, Yongwen Pan, Laurence T. Yang, Minyi Guo, and Nenggan Zheng. Scheduling Co-Design for Reliability and Energy in Cyber-Physical Systems. *IEEE Transactions on Emerging Topics in Computing*, 1(2):353–365, dec 2013.

[22] Yuvraj Agarwal, Stefan Savage, and Rajesh Gupta. SleepServer: A software-only approach for reducing the energy consumption of PCs within enterprise environments. In *Proceedings of the 2010 USENIX Annual Technical Conference*, pages 285–299, Boston, MA, USA, 2010.

[23] Luca Parolini, Bruno Sinopoli, Bruce H. Krogh, and Zhikui Wang. A cyber-physical systems approach to data center modeling and control for energy efficiency. *Proceedings of the IEEE*, 100(1):254–268, 2012.

[24] Lianyong Qi, Yi Chen, Yuan Yuan, Shucun Fu, Xuyun Zhang, and Xiaolong Xu. A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems. *World Wide Web*, 23(2):1275–1297, mar 2020.

[25] Elhadi M. Shakshuki, Haroon Malik, and Tarek Sheltami. WSN in cyber physical systems: Enhanced energy management routing approach using software agents. *Future Generation Computer Systems*, 31(1):93–104, feb 2014.

[26] Changyi Deng, Ruifeng Guo, Chao Liu, Ray Y. Zhong, and Xun Xu. Data cleansing for energy-saving: a case of Cyber-Physical Machine Tools health monitoring system. *International Journal of Production Research*, 56(1-2):1000–1015, jan 2018.

[27] Baris Aksanli and Tajana Simunic Rosing. Human Behavior Aware Energy Management in Residential Cyber-Physical Systems. *IEEE Transactions on Emerging Topics in Computing*, 8(1):45–57, 2016.

[28] Joaquim Leitão, Paulo Gil, Bernardete Ribeiro, and Alberto Cardoso. Improving household's efficiency via scheduling of water and energy appliances. In *Proceedings of the 13th APCA International Conference on Control and Soft Computing, CONTROLO*, pages 253–258, Ponta Delgada, Portugal, June 2018. IEEE.

[29] Shyam Sundar Prasad and Chanakya Kumar. An energy efficient and reliable internet of things. In *Proceedings of the 2012 International Conference on Communication, Information & Computing Technology (ICCICT)*, pages 1–4, Mumbai, oct 2012. IEEE.

[30] Chenlei Zhang, Abram Hindle, and Daniel M German. The Impact of User Choice on Energy Consumption. *IEEE Software*, 31(3):69–75, may 2014.

[31] Maxime Daniel, Guillaume Rivière, and Nadine Couture. CairnFORM. In *Proceedings of the Thirteenth International Conference on Tangible, Embedded, and Embodied Interaction TEI 2019*, pages 275–286, New York, NY, USA, mar 2019. ACM.

[32] Abigail Francisco and John E. Taylor. Understanding citizen perspectives on open urban energy data through the development and testing of a community energy feedback system. *Applied Energy*, 256(September), dec 2019.

[33] Ioannis C. Konstantakopoulos, Andrew R. Barkan, Shiying He, Tanya Veeravalli, Huihan Liu, and Costas Spanos. A deep learning and gamification approach to improving human-building interaction and energy efficiency in smart infrastructure. *Applied Energy*, 237(September 2018):810–821, mar 2019.

[34] Ching Hu Lu. IoT-enabled adaptive context-aware and playful cyber-physical system for everyday energy savings. *IEEE Transactions on Human-Machine Systems*, 48(4):380–391, 2018.

[35] Brian Orland, Nilam Ram, Dean Lang, Kevin Houser, Nate Kling, and Michael Coccia. Saving energy in an office environment: A serious game intervention. *Energy and Buildings*, 74(May):43–52, 2014.

[36] Jose-Miguel Horcas, Mónica Pinto, and Lidia Fuentes. Context-Aware Energy-Efficient Applications for Cyber-Physical Systems. *Ad Hoc Networks*, 82(8):15–30, jan 2019.

[37] Hakduran Koc and Pranitha P Madupu. Optimizing energy consumption in cyber physical systems using multiple operating modes. In *Proceedings of the 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 520–525, Las Vegas, NV, USA, jan 2018. IEEE.

[38] Simone Baldi, Christos D. Korkas, Maolong Lv, and Elias B. Kosmatopoulos. Automating occupant-building interaction via smart zoning of thermostatic loads: A switched self-tuning approach. *Applied Energy, Elsevier*, 231(August):1246–1258, 2018.

[39] Xue Lin, Paul Bogdan, Naehyuck Chang, and Massoud Pedram. Machine learning-based energy management in a hybrid electric vehicle to minimize total operating cost. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 627–634, Austin TX USA, nov 2015. IEEE.

[40] Hussain Shareef, Maytham S. Ahmed, Azah Mohamed, and Eslam Al Hassan. Review on Home Energy Management System Considering Demand Responses, Smart Technologies, and Intelligent Controllers. *IEEE Access*, 6:24498–24509, 2018.

[41] Chen-Khong Tham and Tie Luo. Sensing-Driven Energy Purchasing in Smart Grid Cyber-Physical System. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(4):773–784, jul 2013.

[42] Peng Zhao, M. Godoy Simões, and Siddharth Suryanarayanan. A conceptual scheme for cyber-physical systems based energy management in building structures. *2010 9th IEEE/IAS International Conference on Industry Applications, INDUSCON 2010*, 2010.

[43] Wei Wu, Muhammad Khalid Aziz, Hantao Huang, Hao Yu, and Hoay Beng Gooi. A real-time cyber-physical energy management system for smart houses. In *2011 IEEE PES Innovative Smart Grid Technologies*, pages 1–8, Perth, WA, Australia, nov 2011. IEEE.

[44] Ahmed Yousuf Saber and Ganesh Kumar Venayagamoorthy. Efficient utilization of renewable energy sources by gridable vehicles in cyber-physical energy systems. *IEEE Systems Journal*, 4(3):285–294, 2010.

[45] Zhenyu Zhou, Bingchen Wang, Mianxiong Dong, and Kaoru Ota. Secure and Efficient Vehicle-to-Grid Energy Trading in Cyber Physical Systems: Integration of Blockchain and Edge Computing. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(1):43–57, jan 2020.

[46] Raja Wasim Ahmad, Abdullah Gani, Siti Hafizah Ab Hamid, Muhammad Shiraz, Abdullah Yousafzai, and Feng Xia. A survey on virtual machine migration and server consolidation frameworks for cloud data centers. *Journal of Network and Computer Applications*, 52:11–25, jun 2015.

[47] Jan Kleissl and Yuvraj Agarwal. Cyber-Physical Energy Systems: Focus on Smart Buildings. In *Proceedings of the 47th Design Automation Conference on - DAC '10*, pages 749–754, New York, New York, USA, 2010. Association for Computing Machinery.

[48] Matthieu Simonin, Eugen Feller, Anne-Cécile Orgerie, Yvon Jégou, and Christine Morin. Snooze: An Autonomic and Energy-Efficient Management System for Private Clouds. In *Proceedings of the European Conference on Energy Efficiency in Large Scale Distributed Systems*, volume 8046, pages 114–117. Springer, New York, NY, USA, 2013.

[49] Xiaolong Xu, Xuyun Zhang, Maqbool Khan, Wanchun Dou, Shengjun Xue, and Shui Yu. A balanced virtual machine scheduling method for energy-performance trade-offs in cyber-physical cloud systems. *Future Generation Computer Systems*, 105:789–799, apr 2020.

[50] Deze Zeng, Lin Gu, and Hong Yao. Towards energy efficient service composition in green energy powered Cyber-Physical Fog Systems. *Future Generation Computer Systems*, 2018.

[51] IBM. An architectural blueprint for autonomic computing. Technical Report June, International Business Machines Corporation (IBM), 2006.

[52] J.O. Kephart and D.M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, jan 2003.

[53] U.S. Environmental Protection Agency and U.S. Department [Online] of Energy. ENERGY STAR ®. The simple choice for energy efficiency. `https://www.energystar.gov/`, 2017 [Accessed 2020 Mar 09].

[54] Edward R Griffor, Christopher Greer, David A Wollman, and Martin J Burns. Framework for cyber-physical systems: volume 1, overview. Technical Report June, National Institute of Standards and Technology, Gaithersburg, MD, jun 2017.

[55] Jianhua Shi, Jiafu Wan, Hehua Yan, and Hui Suo. A Survey of Cyber-Physical Systems. In *Proceedings of the 2011 International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6, Nanjing, November 2011. IEEE.

[56] Yao Xiao, Shahin Nazarian, and Paul Bogdan. Self-Optimizing and Self-Programming Computing Systems: A Combined Compiler, Complex Networks, and Machine Learning Approach. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(6):1416–1427, jun 2019.

[57] Liang Hu, Nannan Xie, Zhejun Kuang, and Kuo Zhao. Review of Cyber-Physical System Architecture. In *2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, pages 25–30, Shenzhen, Guangdong, China, apr 2012. IEEE.

[58] Yuankun Xue, Saul Rodriguez, and Paul Bogdan. A Spatio-Temporal Fractal Model for a CPS Approach to Brain-Machine-Body Interfaces. In *Proceedings of the 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 642–647, Singapore, 2016. Research Publishing Services.

[59] Mahboobeh Ghorbani and Paul Bogdan. A cyber-physical system approach to artificial pancreas design. In *2013 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 1–10, Montreal, sep 2013. IEEE.

[60] Gaurav Gupta, Sergio Pequito, and Paul Bogdan. Re-Thinking EEG-Based Non-Invasive Brain Interfaces: Modeling and Analysis. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 275–286, Porto, apr 2018. IEEE.

[61] Rouse Margaret. What is ICT (Information and Communications Technology)? [Online]. https://www.techtarget.com/searchcio/definition/ICT-information-and-communications-technology-or-technologies, 2005 [Accessed 2022 Jul 22].

[62] OSI [Online]. ISO/IEC 7498-1:1994 - Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model. https://www.iso.org/standard/20269.html, 1994 [Accessed on 2020 Apr 10].

[63] Manuel Sanchez, Ernesto Exposito, and Jose Aguilar. Industry 4.0: survey from a system integration perspective. *International Journal of Computer Integrated Manufacturing*, 00(00):1–25, 2020.

[64] A. A. Letichevsky, O. O. Letychevskyi, V. G. Skobelev, and V. A. Volkov. Cyber-Physical Systems. *Cybernetics and Systems Analysis*, 53(6):821–834, 2017.

[65] Matthew N. O. Sadiku, Yonghui Wang, Suxia Cui, and Sarhan M. Musa. Cyber-Physical Systems: A Literature Review. *European Scientific Journal, ESJ*, 13(36):52, 2017.

[66] Kaiyu Wan, Danny Hughes, Ka Lok Man, Tomas Krilavicious, and Shuju Zou. Investigation on Composition Mechanisms for Cyber Physical Sytems. *International Journal of Design, Analysis and Tools for Integrated Circuits and Systems*, 2(1):33–89, 2011.

[67] Yuankun Xue, Ji Li, Shahin Nazarian, and Paul Bogdan. Fundamental challenges toward making the iot a reachable reality: A model-centric investigation. *ACM Transactions on Design Automation of Electronic Systems*, 22(3):1–25, 2017.

[68] Hyun Jung La and Soo Dong Kim. A Service-Based Approach to Designing Cyber Physical Systems. In *Proceedings of the 2010 IEEE/ACIS 9th International Conference on Computer and Information Science*, pages 895–900, Yamagata, Japan, aug 2010. IEEE.

[69] Teodora Sanislav, George Mois, Silviu Folea, Liviu Miclea, Giulio Gambardella, and Paolo Prinetto. A cloud-based cyber-physical system for environmental monitoring. In *Proceedings of the 2014 3rd Mediterranean Conference on Embedded Computing (MECO)*, pages 6–9, Budva, June 2014. IEEE.

[70] Adam Hahn, Roshan K. Thomas, Ivan Lozano, and Alvaro Cardenas. A multi-layered and kill-chain based security analysis framework for cyber-physical systems. *International Journal of Critical Infrastructure Protection*, 11(December):39–50, dec 2015.

[71] Ovunc Kocabas, Tolga Soyata, and Mehmet K. Aktas. Emerging Security Mechanisms for Medical Cyber Physical Systems. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 13(3):401–416, may 2016.

[72] Ciprian-Radu Rad, Olimpiu Hancu, Ioana-Alexandra Takacs, and Gheorghe Olteanu. Smart Monitoring of Potato Crop: A Cyber-Physical System Architecture Model in the Field of Precision Agriculture. *Agriculture and Agricultural Science Procedia*, 6(December):73–79, 2015.

[73] Jacob Wurm, Yier Jin, Yang Liu, Shiyan Hu, Kenneth Heffner, Fahim Rahman, and Mark Tehranipoor. Introduction to Cyber-Physical System Security: A Cross-Layer Perspective. *IEEE Transactions on Multi-Scale Computing Systems*, 3(3):215–227, 2017.

[74] Danish Energy Agency. Denmark's Energy and Climate Outlook 2019. Technical report, Danish Energy Agency, Copenhagen, 2019.

[75] Barbara Kitchenham, Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. Systematic literature reviews in software engineering-a systematic literature review. *Information and Software Technology*, 51:7–15, 01 2009.

[76] Wanchun Dou, Xiaolong Xu, Shunmei Meng, Xuyun Zhang, Chunhua Hu, Shui Yu, and Jian Yang. An energy-aware virtual machine scheduling method for service QoS enhancement in clouds over big data. *Concurrency and Computation: Practice and Experience*, 29(14):e3909, jul 2017.

[77] Y.C. Liang, X. Lu, W.D. Li, and S. Wang. Cyber Physical System and Big Data enabled energy efficient machining optimisation. *Journal of Cleaner Production*, 187:46–62, jun 2018.

[78] Lara Qasim, Marija Jankovic, Sorin Olaru, and Jean-Luc Garnier. Model-based system reconfiguration: A descriptive study of current industrial challenges. In Eric Bonjour, Daniel Krob, Luca Palladino, and François Stephan, editors, *Complex Systems Design & Management*, pages 97–108, Cham, 2019. Springer International Publishing.

[79] U.S. Department of Energy [Online]. EnergyPlus — EnergyPlus. `https://energyplus.net/`, [Accessed on 2020 Jun 02].

[80] Yu Xin Liu, Anfeng Liu, Shuang Guo, Zhetao Li, Young June Choi, and Hiroo Sekiya. Context-aware collect data with energy efficient in Cyber-physical cloud systems. *Future Generation Computer Systems*, 2016.

[81] X. X. Li, F. Z. He, and W. D. Li. A cloud-terminal-based cyber-physical system architecture for energy efficient machining process optimization. *Journal of Ambient Intelligence and Humanized Computing*, 10(3):1049–1064, mar 2019.

[82] Friedenthal Sanford, Dov Dori, and Yaniv Mordecai. Why model? In *SEBoK Editorial Board.2022.The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 2.6, R.J. Cloutier (Editor in Chief)*. Hoboken, NJ: The Trustees of the Stevens Institute of Technology., 2022. Accessed on 29.09.2022, BKCASE is managed and maintained by the Stevens Institute of Technology Systems Engineering Research Center, the International Council on Systems Engineering, and the Institute of Electrical and Electronics Engineers Systems Council.

[83] Jason M. Aughenbaugh and Christiaan J. J. Paredis. The Role and Limitations of Modeling and Simulation in Systems Design. In *Computers and Information in Engineering*, pages 13–22. ASMEDC, November 2004.

[84] Irene Manotas, Christian Bird, Rui Zhang, David Shepherd, Ciera Jaspan, Caitlin Sadowski, Lori Pollock, and James Clause. An empirical study of practitioners' perspectives on green software engineering. In *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*, volume May, pages 237–248, New York, New York, USA, 2016. ACM Press.

[85] Thomas Kurpick, Claas Pinkernell, Markus Look, and Bernhard Rumpe. Modeling cyber-physical systems: Model-Driven Specification of Energy Efficient Buildings. In *Proceedings*

of the Modelling of the Physical World Workshop on - MOTPW '12, pages 1–6, New York, New York, USA, 2012. ACM Press.

[86] Paul Bogdan and Radu Marculescu. Cyberphysical systems: Workload modeling and design optimization. *IEEE Design and Test of Computers*, 28(4):78–87, 2011.

[87] Kyu Tae Park, Yong Tae Kang, Suk Gon Yang, Wen Bin Zhao, Yong-Shin Kang, Sung Ju Im, Dong Hyun Kim, Su Young Choi, and Sang Do Noh. Cyber Physical Energy System for Saving Energy of the Dyeing Process with Industrial Internet of Things and Manufacturing Big Data. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 7(1):219–238, jan 2020.

[88] Chenlei Zhang, Abram Hindle, and Daniel M. German. The Impact of User Choice on Energy Consumption. *IEEE Software*, 31(3):69–75, may 2014.

[89] Sulayman K. Sowe, Eric Simmon, Koji Zettsu, Frederic De Vaulx, and Irena Bojanova. Cyber-Physical-Human Systems: Putting People in the Loop. *IT Professional*, 18(1):10–13, 2016.

[90] Jasmeet Singh, Kshirasagar Naik, and Veluppillai Mahinthan. Impact of Developer Choices on Energy Consumption of Software on Servers. *Procedia Computer Science*, 62(Scse):385–394, 2015.

[91] Hayri Acar, Gülfem Isiklar Alptekin, Jean-Patrick Gelas, and Parisa Ghodous. The Impact of Source Code in Software on Power Consumption. *International Journal of Electronic Business Management*, 14:42–52, 2016.

[92] Wellington Oliveira, Renato Oliveira, and Fernando Castor. A study on the energy consumption of android app development approaches. In *Proceedings of the 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 42–52, Buenos Aires, 2017. IEEE.

[93] Adel Noureddine, Aurelien Bourdon, Romain Rouvoy, and Lionel Seinturier. A preliminary study of the impact of software engineering on GreenIT. In *Proceedings of the First International Workshop on Green and Sustainable Software (GREENS)*, pages 21–27, Zurich, Switzerland, jun 2012. IEEE.

[94] Hui Chen, Shinan Wang, and Weisong Shi. Where does the power go in a computer system: Experimental analysis and implications. In *Proceedings of the 2011 International Green Computing Conference and Workshops*, pages 1–6, Orlando, FL, USA, jul 2011. IEEE.

[95] R. Trobec, M. Depolli, K. Skala, and T. Lipic. Energy efficiency in large-scale distributed computing systems. In *Proceedings of the 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2013)*, pages 253–257, Opatija, Croatia, January 2013. IEEE.

[96] Muhammad Nassar, Julian Jarrett, Iman Saleh, and M. Brian Blake. Generating real-time profiles of runtime energy consumption for Java applications. In *Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering, SEKE*, volume 2014-Janua, pages 592–597, Vancouver, Canada, January 2014.

[97] Rubén Saborido, Venera Arnaoudova, Giovanni Beltrame, Foutse Khomh, Giuliano Antoniol, Soccer Mist, Swat Labs, and Polytechnique Montréal. On the Impact of Sampling Frequency on Software Energy Measurements. *PeerJ PrePrints*, 3:e1219v2, 2015.

[98] Raja Jurdak. *Wireless Ad Hoc and Sensor Networks*. Springer Series on Signals and Communication Technology. Springer US, Boston, MA, Boston, MA, 2007.

[99] Pascal Roques. 1 - reminders for the arcadia method. In Pascal Roques, editor, *Systems Architecture Modeling with the Arcadia Method*, pages 1–23. Elsevier, 2018.

[100] N. Noy and Deborah Mcguinness. Ontology development 101: A guide to creating your first ontology. *Knowledge Systems Laboratory*, 32, 01 2001.

[101] ORACLE [Online]. What Is a Database — Oracle. `https://www.oracle.com/database/what-is-database/`, 2015 [Accessed 2022 Jun 02].

[102] Michal Sir, Zdenek Bradac, and Petr Fiedler. Ontology versus database. *IFAC-PapersOnLine*, 48(4):220–225, 2015. 13th IFAC and IEEE Conference on Programmable Devices and Embedded Systems.

[103] Michael Uschold. Ontology and database schema: What's the difference? *Applied Ontology*, 10(3-4):243–258, 2015.

[104] Borja Ramis Ferrer, Wael M Mohammed, Mussawar Ahmad, Sergii Iarovyi, Jiayi Zhang, Robert Harrison, and Jose Luis Martinez Lastra. Comparing ontologies and databases: a critical review of lifecycle engineering models in manufacturing. *Knowledge and Information Systems*, 63(6):1271–1304, 2021.

[105] Carmen Martinez-Cruz, Ignacio J. Blanco, and M. Amparo Vila. Ontologies versus relational databases: Are they so different? A comparison. *Artificial Intelligence Review*, 38(4):271–290, 2012.

[106] Kamran Munir and M. Sheraz Anjum. The use of ontologies for effective knowledge modelling and information retrieval. *Applied Computing and Informatics*, 14(2):116–126, 2018.

[107] insightsoftware.com [Online]. Comparing Descriptive, Predictive, Prescriptive, and Diagnostic Analytics. `https://insightsoftware.com/blog/comparing-descriptive-predictive-prescriptive-and-diagnostic%2Danalytics/`, 2021 [Accessed 2022 Oct 05].

[108] MohammadNoor Injadat, Abdallah Moubayed, Ali Bou Nassif, and Abdallah Shami. Machine learning towards intelligent systems: applications, challenges, and opportunities. *Artificial Intelligence Review*, 54(5):3299–3348, jun 2021.

[109] Julianna Delua. Supervised vs. Unsupervised Learning: What's the Difference? — IBM [Online]. `https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning`, 2021 [Accessed 2022 Oct 07].

[110] Fabian Astudillo-Salinas, Daniela Barrera-Salamea, Andres Vazquez-Rodas, and Lizandro Solano-Quinde. Minimizing the power consumption in Raspberry Pi to use as a remote WSN gateway. *2016 8th IEEE Latin-American Conference on Communications, LATINCOM 2016*, 2016.

[111] Adel Noureddine, Romain Rouvoy, and Lionel Seinturier. Monitoring energy hotspots in software. *Automated Software Engineering*, 22(3):291–332, 2015.

[112] Basireddy Karunakar Reddy, Matthew J Walker, Domenico Balsamo, Stephan Diestelhorst, Bashir M Al-Hashimi, and Geoff V Merrett. Empirical cpu power modelling and estimation in the gem5 simulator. In *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 1–8. IEEE, 2017.

[113] Girish Bekaroo and Aditya Santokhee. Power consumption of the Raspberry Pi: A comparative analysis. *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies, EmergiTech 2016*, pages 361–366, 2016.

[114] Adnan Sabovic, Carmen Delgado, Jan Bauwens, Eli De Poorter, and Jeroen Famaey. Accurate Online Energy Consumption Estimation of IoT Devices Using Energest. *Lecture Notes in Networks and Systems*, 97(November):363–373, 2020.

[115] Behnam Dezfouli, Immanuel Amirtharaj, and Chia Chi (Chelsey) Li. EMPIOT: An energy measurement platform for wireless IoT devices. *Journal of Network and Computer Applications*, 121(NOVEMBER):135–148, 2018.

[116] Guillaume Fieni, Romain Rouvoy, and Lionel Seinturier. SmartWatts: Self-Calibrating Software-Defined Power Meter for Containers. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pages 479–488, guillaume2020, may 2020. IEEE.

[117] Tahir Diop, Natalie Enright Jerger, and Jason Anderson. Power modeling for heterogeneous processors. *ACM International Conference Proceeding Series*, pages 90–98, 2014.

[118] Maxime Colmant, Romain Rouvoy, Mascha Kurpicz, Anita Sobe, Pascal Felber, and Lionel Seinturier. The next 700 CPU power models. *Journal of Systems and Software*, 144:382–396, oct 2018.

[119] W. L. Bircher, M. Valluri, J. Law, and L. K. John. Runtime identification of microprocessor energy saving opportunities. In *Proceedings of the 2005 international symposium on Low power electronics and design - ISLPED '05*, page 275, New York, New York, USA, May 2005. ACM Press.

[120] Karan Singh, Major Bhadauria, and Sally A. McKee. Real time power estimation and thread scheduling via performance counters. *ACM SIGARCH Computer Architecture News*, 37(2):46–55, 2009.

[121] Christoph Möbius, Waltenegus Dargie, and Alexander Schill. Power consumption estimation models for processors, virtual machines, and servers. *IEEE Transactions on Parallel and Distributed Systems*, 25(6):1600–1614, 2014.

[122] Jose Nunez-Yanez and Geza Lore. Enabling accurate modeling of power and energy consumption in an ARM-based System-on-Chip. *Microprocessors and Microsystems*, 37(3):319–332, 2013.

[123] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. Data center energy consumption modeling: A survey. *IEEE Communications Surveys and Tutorials*, 18(1):732–794, 2016.

[124] Fabian Kaup, Philip Gottschling, and David Hausheer. PowerPi: Measuring and modeling the power consumption of the Raspberry Pi. *Proceedings - Conference on Local Computer Networks, LCN*, pages 236–243, 2014.

[125] Kamar Kesrouani, Houssam Kanso, and Adel Noureddine. A Preliminary Study of the Energy Impact of Software in Raspberry Pi devices. In *29th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 231–234, Bayonne, France, 2020.

[126] Raspberry Pi Foundation [Online]. Power Supply - Raspberry Pi Documentation. `https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md`, 2020 [Accessed on 2021 Apr 01].

[127] Suzanne Rivoire, Parthasarathy Ranganathan, and Christos Kozyrakis. A comparison of high-level full-system power models. *Workshop on Power Aware Computing and Systems, HotPower 2008*, 2008.

[128] Dimitris Economou, Suzanne Rivoire, Christos Kozyrakis, and Partha Ranganathan. Full-System Power Analysis and Modeling for Server Environments. *Workshop on Modeling, Benchmarking and Simulation (MoBS)*, pages 807–812, 2006.

[129] Sweta Bhattacharya, Praveen Kumar Reddy Maddikunta, Iyapparaja Meenakshisundaram, Thippa Reddy Gadekallu, Sparsh Sharma, Mohammed Alkahtani, and Mustufa Haider Abidi. Deep Neural Networks Based Approach for Battery Life Prediction. *Computers, Materials & Continua*, 69(2):2599–2615, 2021.

[130] Siva Rama Krishnan Somayaji, Mamoun Alazab, Manoj MK, Antonio Bucchiarone, Chiranji Lal Chowdhary, and Thippa Reddy Gadekallu. A Framework for Prediction and Storage of Battery Life in IoT Devices using DNN and Blockchain. In *2020 IEEE Globecom Workshops (GC Wkshps*, pages 1–6. IEEE, dec 2020.

[131] Matteo Croce. Why you should run a 64 bit os on your raspberry pi4 [online]. `https://matteocroce.medium.com/why-you-should-run-a-64-bit-os-on%2Dyour-raspberry-pi4-bd5290d48947`, january 2020 [Accessed 2022 Apr 01].

[132] A.J. Lewis, M. Campbell, and P. Stavroulakis. Performance evaluation of a cheap, open source, digital environmental monitor based on the raspberry pi. *Measurement*, 87:228–235, 2016.

[133] S. Kumar and A. Jasuja. Air quality monitoring system based on iot using raspberry pi. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 1341–1346, 2017.

[134] S. N. Ishak, N. N. N. A. Malik, N. M. A. Latiff, N. E. Ghazali, and M. A. Baharudin. Smart home garden irrigation system using raspberry pi. In *2017 IEEE 13th Malaysia International Conference on Communications (MICC)*, pages 101–106, Nov 2017.

[135] R. Kumar and M. P. Rajasekaran. An iot based patient monitoring system using raspberry pi. In *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*, pages 1–4, 2016.

[136] Eprel - european product registry for energy labelling [online]. https://eprel.ec.europa.eu/screen/product/lightsources/1286006?navigatingfrom=qr, [Accessed 2022 Oct 22].

[137] Beate Naser, Franziska Schäfer, and Jörg Franke. An Energy Ontology Focusing on Demand Side Management in Smart Homes. *Advanced Engineering Forum*, 19:124–131, 2016.

[138] Paolo Brizzi, Dario Bonino, Alberto Musetti, Alexandr Krylovskiy, Edoardo Patti, and Mathias Axling. Towards an ontology driven approach for systems interoperability and energy management in the smart city. In *Proceeding of International Multidisciplinary Conference on Computer and Energy Science, SpliTech 2016*, Split, Croatia, Jul 2016. IEEE.

[139] Maliheh Haghgoo, Ilya Sychev, Antonello Monti, and Frank H.P. Fitzek. SARGON – Smart energy domain ontology. *IET Smart Cities*, 2020.

[140] Safina Showkat Ara, Zia Ush Shamszaman, and Ilyoung Chong. Web-of-objects based user-centric semantic service composition methodology in the internet of things. *International Journal of Distributed Sensor Networks*, 2014, 2014.

[141] Karl Hammar, Erik Oskar Wallin, Per Karlberg, and David Halleberg. The realestatecore ontology. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtěch Svátek, Isabel Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web – ISWC 2019. Lecture Notes in Computer Science*, volume 11779.Springer, pages 130–145, Cham, 2019. Springer International Publishing.

[142] Christian Reinisch, MarioJ Kofler, Félix Iglesias, and Wolfgang Kastner. ThinkHome Energy Efficiency in Future Smart Homes. *EURASIP Journal on Embedded Systems*, 2011(1):104617, 2011.

[143] Laura Daniele, Frank den Hartog, and Jasper Roes. Created in Close Interaction with the Industry: The Smart Appliances REFerence (SAREF) Ontology. In *Lecture Notes in Business Information Processing*, volume 225, pages 100–112. Springer, 2015.

[144] Michael Compton, Payam Barnaghi, Luis Bermudez, Raúl García-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, Vincent Huang, Krzysztof Janowicz, W. David Kelsey, Danh Le Phuoc, Laurent Lefort, Myriam Leggieri, Holger Neuhaus, Andriy Nikolov, Kevin Page, Alexandre Passant, Amit Sheth, and Kerry Taylor. The SSN ontology of the W3C semantic sensor network incubator group. *Journal of Web Semantics*, 17:25–32, dec 2012.

[145] Mads Holten Rasmussen, Maxime Lefrançois, Georg Ferdinand Schneider, and Pieter Pauwels. BOT: The building topology ontology of the W3C linked building data group. *Semantic Web*, 12(1):143–161, nov 2020.

[146] Sean Bechhofer. *OWL: Web Ontology Language*, pages 2008–2009. Springer US, Boston, MA, 2009.

[147] Mark A. Musen. The protégé project: A look back and a look forward. *AI Matters*, 1(4):4–12, jun 2015.

[148] Izzeddin Gur, Natasha Jaques, Yingjie Miao, Jongwook Choi, Manoj Tiwari, Honglak Lee, and Aleksandra Faust. Environment Generation for Zero-Shot Compositional Reinforcement

Learning. *Advances in Neural Information Processing Systems*, 6(NeurIPS):4157–4169, 2021.

[149] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions. Technical report, UBER, jan 2019.

[150] Andres Campero, Roberta Raileanu, Heinrich Küttler, Joshua B. Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with AMIGo: Adversarially Motivated Intrinsic Goals. In *International Conference on Learning Representations (ICLR)*, pages 1–18, Virtual Only, jun 2021.

[151] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent Complexity and Zero-shot Transfer via Unsupervised Environment Design. *Advances in Neural Information Processing Systems*, 2020-Decem(NeurIPS), dec 2020.

[152] Daniel J. Fremont, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Scenic: A language for scenario specification and scene generation. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, page 63–78, New York, NY, USA, 2019. Association for Computing Machinery.

[153] Abdus Salam Azad, Edward Kim, Qiancheng Wu, Kimin Lee, Ion Stoica, Pieter Abbeel, and Sanjit A. Seshia. Scenic4RL: Programmatic Modeling and Generation of Reinforcement Learning Environments. *Currently Under Review*, jun 2021.

[154] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. Google Research Football: A Novel Reinforcement Learning Environment. *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, pages 4501–4510, jul 2019.

[155] J. Rowe, A. Smith, R. Pokorny, B. Mott, and J. Lester. Toward Automated Scenario Generation with Deep Reinforcement Learning in GIFT. In *Proceedings of the 6th Annual GIFT Users Symposium (GIFTSym6)*, page 65, Orlando, Florida, 2018.

[156] Ao Li, Shitao Chen, Liting Sun, Nanning Zheng, Masayoshi Tomizuka, and Wei Zhan. Scegene: Bio-inspired traffic scenario generation for autonomous driving testing. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):14859–14874, 2022.

[157] Zahra Ghodsi, Siva Kumar Sastry Hari, Iuri Frosio, Timothy Tsai, Alejandro Troccoli, Stephen W. Keckler, Siddharth Garg, and Anima Anandkumar. Generating and characterizing scenarios for safety testing of autonomous vehicles. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 157–164, 2021.

[158] Quentin Goss, Yara AlRashidi, and Mustafa Ilhan Akbas. Generation of modular and measurable validation scenarios for autonomous vehicles using accident data. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 251–257, 2021.

[159] Alessio Gambi, Vuong Nguyen, Jasim Ahmed, and Gordon Fraser. Generating critical driv-

ing scenarios from accident sketches. In *2022 IEEE International Conference On Artificial Intelligence Testing (AITest)*, pages 95–102, 2022.

[160] Karl Mason and Santiago Grijalva. A review of reinforcement learning for autonomous building energy management. *Computers and Electrical Engineering*, 78:300–312, 2019.

[161] Liang Yu, Shuqi Qin, Meng Zhang, Chao Shen, Tao Jiang, and Xiaohong Guan. A Review of Deep Reinforcement Learning for Smart Building Energy Management. *IEEE Internet of Things Journal*, 8(15):12046–12063, 2021.

[162] Sunyong Kim and Hyuk Lim. Reinforcement learning based energy management algorithm for smart energy buildings. *Energies*, 11(8), 2018.

[163] Paulo Lissa, Conor Deane, Michael Schukat, Federico Seri, Marcus Keane, and Enda Barrett. Deep reinforcement learning for home energy management system control. *Energy and AI*, 3:100043, mar 2021.

[164] Xu Xu, Youwei Jia, Yan Xu, Zhao Xu, Songjian Chai, and Chun Sing Lai. A Multi-Agent Reinforcement Learning-Based Data-Driven Method for Home Energy Management. *IEEE Transactions on Smart Grid*, 11(4):3201–3211, jul 2020.

[165] Yuankun Liu, Dongxia Zhang, and Hoay Beng Gooi. Optimization strategy based on deep reinforcement learning for home energy management. *CSEE Journal of Power and Energy Systems*, 6(3):572–582, 2020.

[166] Christian Reinisch, Mario J. Kofler, and Wolfgang Kastner. ThinkHome: A smart home as digital ecosystem. *4th IEEE International Conference on Digital Ecosystems and Technologies - Conference Proceedings of IEEE-DEST 2010, DEST 2010*, pages 256–261, 2010.

[167] Renzhi Lu, Seung Ho Hong, and Mengmeng Yu. Demand Response for Home Energy Management Using Reinforcement Learning and Artificial Neural Network. *IEEE Transactions on Smart Grid*, 10(6):6629–6639, 2019.

[168] Liang Yu, Weiwei Xie, Di Xie, Yulong Zou, Dengyin Zhang, Zhixin Sun, Linghua Zhang, Yue Zhang, and Tao Jiang. Deep Reinforcement Learning for Smart Home Energy Management. *IEEE INTERNET OF THINGS JOURNAL*, 7(4):2751 – 2762, sep 2019.

[169] Juan Miguel Gonzalez López, Edris Pouresmaeil, Claudio A. Cañizares, Kankar Bhattacharya, Abolfazl Mosaddegh, and Bharatkumar V. Solanki. Smart Residential Load Simulator for Energy Management in Smart Grids. *IEEE Transactions on Industrial Electronics*, 66(2):1443–1452, 2019.

[170] P Constantopoulos, F.C. Schweppe, and R.C. Larson. Estia: A real-time consumer control scheme for space conditioning usage under spot electricity pricing. *Computers & Operations Research*, 18(8):751–765, jan 1991.

[171] Dong Zhang, Shuhui Li, Min Sun, and Zheng O'Neill. An Optimal and Learning-Based Demand Response and Home Energy Management System. *IEEE Transactions on Smart Grid*, 7(4):1790–1801, jul 2016.

[172] Anupam A. Thatte and Le Xie. Towards a Unified Operational Value Index of Energy Storage in Smart Grid Environment. *IEEE Transactions on Smart Grid*, 3(3):1418–1426, sep 2012.

[173] Najmeh Forouzandehmehr, Samir M. Perlaza, Zhu Han, and H. Vincent Poor. A satisfaction game for heating, ventilation and air conditioning control of smart buildings. *GLOBECOM - IEEE Global Telecommunications Conference*, pages 3164–3169, 2013.

[174] Ruilong Deng, Zhaohui Zhang, Ju Ren, and Hao Liang. Indoor temperature control of cost-effective smart buildings via real-time smart grid communications. *2016 IEEE Global Communications Conference, GLOBECOM 2016 - Proceedings*, pages 0–5, 2016.

[175] Howard J. Diamond, Thomas R. Karl, Michael A. Palecki, C. Bruce Baker, Jesse E. Bell, Ronald D. Leeper, David R. Easterling, Jay H. Lawrimore, Tilden P. Meyers, Michael R. Helfert, Grant Goodge, and Peter W. Thorne. U.S. Climate Reference Network after One Decade of Operations: Status and Assessment. *Bulletin of the American Meteorological Society*, 94(4):485–498, apr 2013.

[176] Michael G. Just, Lauren M. Nichols, and Robert R. Dunn. Human indoor climate preferences approximate specific geographies. *Royal Society Open Science*, 6(3):180695, mar 2019.

Appendix A

Result of SPARQL Query

SPARQL result of query 1 (list of actions) in case study I:


```
{
   "results":{
      "bindings":[
         {
            "Device":{
               "type":"uri",
               "value":"Led"
            },
            "Property":{
               "type":"uri",
               "value":"LedBrightness"
            },
            "Range":{
               "type":"literal",
               "value":"MinMaxStep:0,100,1",
               "datatype":"string"
            },
            "DataType":{
               "type":"uri",
               "value":"int"
            }
         },
         {
            "Device":{
               "type":"uri",
               "value":"Led"
            },
            "Property":{
               "type":"uri",
               "value":"LedColor"
            },
            "Range":{
               "type":"literal",
               "value":"List:Red,Blue,Green,White",
               "datatype":"string"
            },
            "DataType":{
```

```
                "type":"uri",
                "value":"string"
            }
        },
        {
            "Device":{
                "type":"uri",
                "value":"TV"
            },
            "Property":{
                "type":"uri",
                "value":"TVBrightness"
            },
            "Range":{
                "type":"literal",
                "value":"MinMaxStep:0,100,10",
                "datatype":"string"
            },
            "DataType":{
                "type":"uri",
                "value":"int"
            }
        },
        {
            "Device":{
                "type":"uri",
                "value":"TV"
            },
            "Property":{
                "type":"uri",
                "value":"TVSaturation"
            },
            "Range":{
                "type":"literal",
                "value":"MinMaxStep:0,100,5",
                "datatype":"string"
            },
            "DataType":{
                "type":"uri",
                "value":"int"
            }
        }
    ]
},
"head":{
    "vars":["Device", "Property", "DataType", "Range" ]
}
}
```

Appendix B

Generated Source Code of Case Study II

Generated initialization function source code:

```
def __init__(self):
    le = preprocessing.LabelEncoder()
    self.firstStep=True

    #START_INIT
    self.action_space = Discrete(5)
    HIGH = np.array([100,50,50,1],dtype=np.float32)
    LOW = np.array([0,0,0,0],dtype=np.float32)
    self.observation_space = Box(LOW, HIGH)
    self.HVACLevel= 50 + random.choice([-20,20])
    self.HVACMaxPower=2000.0
    self.IndoorTemperature=15+ random.randint(-3,3)
    self.OutdoorTemperature=
    ↪  USER_PREFERENCE['AverageTemperature'].get(0)
    le.fit(['OFF', 'ON'])
    HVACStateMapping = dict(zip(le.classes_, le.transform(['OFF',
    ↪  'ON'])))
    self.HVACState=1
    self.PEM=["HVACPower"]
    self.HVACPower=0.0
    self.propertyList=["HVACLevel", "IndoorTemperature",
    ↪  "OutdoorTemperature", "HVACState"]
    self.state = (self.HVACLevel, self.IndoorTemperature,
    ↪  self.OutdoorTemperature, self.HVACState)
    #END_INIT

    # Set cycle length
    self.cycle_length = CYCLE_LENGTH
    self.episodeEnergy=0
```

Generated power estimation source code:

```
#START_ENERGY
def calculate_power(self,device, HVACLevel, HVACMaxPower,
 ↪  IndoorTemperature, OutdoorTemperature):
    if device == "HVACPower":
        if self.HVACState == 0:
            power=0
        else:
            power= HVACLevel/100*HVACMaxPower
    return(power)

def total_power(self):
    self.HVACPower=self.calculate_power("HVACPower",
     ↪  self.HVACLevel, self.HVACMaxPower,
     ↪  self.IndoorTemperature, self.OutdoorTemperature)
    total=self.HVACPower
    return total
#END_ENERGY
```

Generated actions source code (part of the step function):

```
#START_ACTIONS
    if action==0:
        if(self.HVACLevel>=0 and self.HVACLevel<=80):
         ↪  self.HVACLevel=self.HVACLevel+20
    if action==1:
        if(self.HVACLevel>=20 and self.HVACLevel<=100):
         ↪  self.HVACLevel=self.HVACLevel-20
    if action==2:
        self.HVACState=0
    if action==3:
        self.HVACState=1
    if action==4:
        pass
#END_ACTIONS
```

Generated reward function source code (part of the step function):

```
#START_REWARD
if (USER_PREFERENCE['USER_PRESENCE'].get(CSV_ID)==1):
  if(self.IndoorTemperature < USER_PREFERENCE['MIN_TEMPERATURE'].get(CSV_ID)):
    reward = -1*(USER_PREFERENCE['MIN_TEMPERATURE'].get(CSV_ID) - self.IndoorTemperature) / (1+(USER_PREFERENCE['MIN_TEMPERATURE'].get(CSV_ID) - self.IndoorTemperature))
  elif(self.IndoorTemperature > USER_PREFERENCE['MAX_TEMPERATURE'].get(CSV_ID)):
    reward = -1*(self.IndoorTemperature - USER_PREFERENCE['MAX_TEMPERATURE'].get(CSV_ID)) / (1+(self.IndoorTemperature - USER_PREFERENCE['MAX_TEMPERATURE'].get(CSV_ID)))
  else:
    reward=1-2*self.totalEnergy/MAX_POWER_T
else:
  if(self.HVACState==0):
    reward=1
  else:
    reward=-1
#END_REWARD
```

Generated states source code (part of the step function):

```
#START_STATES
self.state = (self.HVACLevel, self.IndoorTemperature, self.OutdoorTemperature, self.HVACState)
#END_STATES
```