electronics

MDPI

Article

Intelligent Optimization Mechanism Based on an Objective Function for Efficient Home Appliances Control in an Embedded Edge Platform

Rongxu Xu [1,†], Wenquan Jin [2,†], Yonggeun Hong [3] and Do-Hyeun Kim [1,*]

[1] Department of Computer Engineering, Jeju National University, Jeju 63243, Korea; rongxu@jejunu.ac.kr
[2] Big Data Research Center, Jeju National University, Jeju 63243, Korea; wenquan.jin@jejunu.ac.kr
[3] Department of Artificial Intelligence Convergence, Daejeon University, Daejeon 34520, Korea; yghong@dju.kr
* Correspondence: kimdh@jejunu.ac.kr; Tel.: +82-64-7543658
† These authors contributed equally to this work.

Abstract: In recent years the ever-expanding internet of things (IoT) is becoming more empowered to revolutionize our world with the advent of cutting-edge features and intelligence in an IoT ecosystem. Thanks to the development of the IoT, researchers have devoted themselves to technologies that convert a conventional home into an intelligent occupants-aware place to manage electric resources with autonomous devices to deal with excess energy consumption and providing a comfortable living environment. There are studies to supplement the innate shortcomings of the IoT and improve intelligence by using cloud computing and machine learning. However, the machine learning-based autonomous control devices lack flexibility, and cloud computing is challenging with latency and security. In this paper, we propose a rule-based optimization mechanism on an embedded edge platform to provide dynamic home appliance control and advanced intelligence in a smart home. To provide actional control ability, we design and developed a rule-based objective function in the EdgeX edge computing platform to control the temperature states of the smart home. Compared to cloud computing, edge computing can provide faster response and higher quality of services. The edge computing paradigm provides better analysis, processing, and storage abilities to the data generated from the IoT sensors to enhance the capability of IoT devices concerning computing, storage, and network resources. In order to satisfy the paradigm of distributed edge computing, all the services are implemented as microservices. The microservices are connected to each other through REST APIs based on the constrained IoT devices to provide all the functionalities that accomplish a trade-off between energy consumption and occupant-desired environment setting for the smart home appliances. We simulated our proposed system to control the temperature of a smart home; through experimental findings, we investigated the application against the delay time and overall memory consumption by the embedded edge system of EdgeX. The result of this research work suggests that the implemented services operated efficiently in the raspberry pi 3 hardware of IoT devices.

Keywords: embedded edge system; objective function; EdgeX; microservice; optimization; smart home

Citation: Xu, R.; Jin, W.; Hong, Y.; Kim, D.-H. Intelligent Optimization Mechanism Based on an Objective Function for Efficient Home Appliances Control in an Embedded Edge Platform. *Electronics* **2021**, *10*, 1460. https://doi.org/10.3390/electronics10121460

Academic Editor: Kah Phooi Seng

Received: 3 May 2021
Accepted: 11 June 2021
Published: 18 June 2021

1. Introduction

Every year, diverse new devices with advanced features and intelligence are presented and launched to realize the concept of the IoT. A plethora of IoT applications such as smart meters, video surveillance, healthcare monitoring, parcel or asset tracking and transportation are making a significant contribution to the growth of devices and connections. IoT connections are one of the fastest-growing categories of devices and connections. According to statistics, IoT will grow nearly 2.4 times over the forecasted period (an average annual growth rate of 19%), reaching 14.7 billion connections by 2023 [1]. The Iot can be specified as a network of intelligent devices that have wireless or wired built-in connectivity or sensors, actuators, and any other functions that can perceive and transfer

the environment information to the cloud [2,3]. The idea of the IoT is to use the Internet as the basis of a communication system to create intelligent interactions between people and surrounding objects while simultaneously bringing different "things" together as a whole. Cloud computing is an important part of the IoT, which provides valuable services to IoT-related implementations in many application areas [4]. The IoT is playing an important role in many areas and domains to improve our quality of life. These applications include smart homes, industrial automation, and disaster relief for natural and man-made disasters where human decisions are difficult to make [5].

During the past few decades, the global IoT market has grown enormously. The growing adoption of cloud computing can be attributed to advance data analytics, cost efficient connected devices and the flexibility achieved through integration of advance principal technologies, where computing, storage, and network management functions are operated in a centralized manner [6]. The cloud-integrated IoT applications offer powerful computing resources, on-demand data storage, and offline analysis of large amounts of data [7]. However, with the rapid development of applications for the mobile internet and IoT, the existing centralized cloud computing architecture-based application faces major challenges. Mobile devices connected to centralized remote cloud servers attempt to receive advanced applications that add additional load to both the radio access network (RAN) and the backhaul network and cause high latency [8]. The IoT presents challenges such as severe latency, capacity limitations, devices with limited resources, uninterrupted services with intermittent connections, and increased security that cannot compete with the legacy of centralized cloud computing architecture [9]. There is an urgent need to develop an advanced cloud computing paradigm to break the centralized architecture and reduce the capacity and latency limitations to meet these challenges.

In edge computing, due to bandwidth and power consumption issues, massive amounts of data collected by various types of IoT devices can be analyzed at the edge of the network, rather than sent to a centralized cloud infrastructure. Compared to cloud computing, edge computing can provide faster response and higher quality of services. Edge computing is better suited to integrate with the IoT paradigm to provide secure and well-organized services to amounts of end-users, and the edge computing architecture can be seen as the future IoT infrastructure [10]. The edge computing paradigm is perceived as a mechanism to satisfy low latency and efficient bandwidth that not only generates data at the edge of the network but also processes large amounts of data. The sensed data can be analyzed with the edge computing paradigm at the edge of the network [11–13]. The edge of the network implicates that the unit is equipped with advanced computing capabilities that consist of networking, storage, computing, and other core functions to generate and process data. There are other similar paradigms, such as Fog Computing [14,15] and Mobile Edge Computing [16,17] that not only provide effective solutions for large-scale computing but also improve the user experience.

The smart home is focused on improving the living environment of users and saving energy consumption. The smart home technique uses electric home appliances and IoT sensors with central control devices to effectively manage energy consumption. The author of [18] presented a smart home system that integrates IoT devices with cloud computing to provide a remote control with smartphones and networks. The author of [19] proposes an IoT and cloud computing-based EMS, which can be accessed remotely by the consumers with a network. These studies ensure remote control with a smartphone application or web browser, which is being challenged with intelligence. The authors of [20,21] present a control mechanism with machine learning algorithms to understand residents' behavior in energy exploitation and needs and provide accurate forecasts in response to residents' demand without wasting energy. However, machine learning-based controllers lack flexibility because they depend on user behavior of fixed schedule [22].

In this paper, we propose an intelligent optimization mechanism that utilizes optimization capability and fuzzy logic in the edge computing framework to provide a trade-off between energy consumption and the desired environment setting in a smart home. To

Improve the intelligence, we design and Implement a rule-based objective function in the EdgeX edge computing platform to control the temperature states of the smart home. We use the edge computing paradigm to enhance the IoT device capability of computing, storage, and network. The edge computing framework is an open-source, interoperable, software platform at the edge of the network, that interacts with the smart home sensors and actuators to provide intelligent home environment control with optimization mechanism and fuzzy logic algorithm.

The remainder of this paper is constructed as follows. Section 2 is the related work that gives a concise introduction to edge computing that moves functionality from the cloud to the edge of the network. Section 3 introduces the proposed intelligent optimization mechanism architecture for smart home environment control. Section 4 provides some insight into the implementation and results of the proposed system. Section 5 reports the performance evaluation of the proposed work. Finally, Section 6 concludes the paper and discusses some directions for future research.

2. Related Work

The recent research of smart home applications can provide a number of benefits; for example, it uses energy efficiency while improving the user experience. However, it is still a difficult task to design a common communication model for each entity of the integrated smart home system.

The author [23] proposed a smart home control system based on a ZigBee network with the help of a coordinator. The proposed work comprises of energy control system that consists of a home appliance control system and a light control system supported by a ZigBee sensor connected to each home appliance and a light sensor placed in the smart home. An intelligent management system has been developed to effectively control the operating time of electronic devices of a smart home. The author [24] proposed a smart home system design based on the communication technology of ZigBee and Powerline Carrier. A communication gateway has also been designed. The author [18] presented a smart home system that integrates ZigBee devices, smart Wi-Fi plugs, IR devices, and cloud computing. Customers can use smartphones and networks to control devices with different communication protocols. The author [25] proposed the design of a smart home environmental monitoring system based on the ARM Cortex-A53 core Raspberry Pi, B / S architecture, and wireless network functionality. The system consists of the smart home controller, EnOcean / Wi-Fi wireless sensor control network, and smart home client. The smart home client allows users to connect to the smart home controller through the client PC. The system uses a wireless EnOcean and Wi-Fi network to query the working status of the home environment in real-time.

In a typical smart home, residents can control the devices in the apartment, e.g., TVs, lamps, radios, fans, air conditioners and monitor living conditions. These devices have many different communication protocols such as ZigBee, Wi-Fi, and Ethernet. It is therefore necessary to integrate heterogeneous network protocols in smart homes. However, the smart home network integrates sensors, embedded devices, wireless technology, distributed data processing, etc. Heterogeneous communication is important, but embedded computing and distributed information processing have great significance. However, the authors generally do not care about the scalability and flexibility of the system in the smart home.

We also explore several open-source platforms that can be used to develop edge computing. EdgeX Foundry is a vendor-neutral interoperability platform for edge computing of IoT. It hosts microservices at edge nodes such as gateways, and routers [26]. It provides connectivity with various sensors and actuators via several protocols, controls them, senses information from them, and transfers the data to an application that is located at the edge of the cloud for advanced processing. EdgeX can be operated at various hardware and operating system. In order to shift analytics from cloud to edge devices, Microsoft Azure presents Azure IoT Edge as a cloud service provider [27]. The edge devices are routers, gateways,

or other nodes, which support computing resources. The user who has their applications in the cloud can move them to edge devices through Azure IoT Edge to reduce latency. It can simplify the development complexity of applications of edge devices. In addition, users can make use of Azure functions, Azure ML, and Azure stream analytics to install complex jobs on the edge nodes such as machine learning, image recognition, and other services related to AI. Apache Edgent [28] is a programming model and lightweight runtime that can be running in small nodes such as gateways and routers. It is dedicated to data analytics at the network of edge, trying to accelerate the implementation of data analysis. CORD [29] is an ONF project, initiated by AT&T and presented for network operators. Network equipment suppliers provide network infrastructure with closed proprietary-integrated systems. With the dependency of network equipment suppliers, the network capability is hard to manage with network operators. Therefore, computing and networking resources are wasted. CORD tries to use software-defined networks, network function virtualization, and cloud technologies to reconstruct the infrastructure of edge networks to build data centers. It divides the resources of reconstructed datacenters such as computing, storage, and network to provide clouds at the edge network.

We propose an intelligent optimization mechanism for efficient home appliances control based on embedded edge computing for a smart home. IoT devices require heterogeneous capabilities related to protocols, operating systems, and applications. Table 1, provides a summary of the open-source edge computing platforms. we compare them considering several aspects such as operating systems, programming languages, goals, and system characteristics. EdgeX Foundry is agnostic to hardware, the CPU, operating systems, supports several programming languages, and is designed to provide interoperability, which can be used to satisfy the requirements of heterogeneity of IoT devices.

Table 1. Comparison of Open Source Platforms for Edge Computing.

Categories Platforms	EdgeX Foundry	Azure IoT Edge	Apache Edgent	CORD
Operating System	Various OS	Various OS	Various OS	Ubuntu
Programming languages	Go lang, Python, java etc.	Java, .NET, C, Python, etc.	Java	Shell script, Python
Goals	Provide Interoperability for IoT edge	Support hybrid cloud-edge analytics	Support process of data analysis	Transform edge of the operator network into agile service delivery platforms
System characteristics	A common API for device management	Powerful Azure services	APIs for data analytics	Widespread edge clouds

3. Intelligent Optimization Mechanism for Efficient Home Appliances Control

In this section, we describe the structure of our proposed system that is integrated into the smart home. To realize enhanced intelligence and make up for the shortcomings of IoT devices, the objective function, and fuzzy logic control logic are also explained.

3.1. Proposed Optimization Mechanism Architecture for Home Appliances Control

We propose an intelligent optimization mechanism based on embedded edge computing that employs optimization functions and fuzzy logic in an edge computing framework to balance energy consumption and environmental factors required in a smart home. As shown in the following Figure 1, the proposed system consists of an intelligent edge computing platform and a smart home environment.

The smart home environment includes temperature sensors, and heating and cooling actuators. The temperature sensors sense the temperature of the smart home and send it to intelligent edge computing platform to provide automatic temperature control through execution of a cooler or heater properly. The sensors and actuators are located in a smart home to support connection with intelligent edge computing platforms via the internet.

The intelligent edge computing platform services are running in an embedded system that is equipped with a WIFI network to provide connectivity.

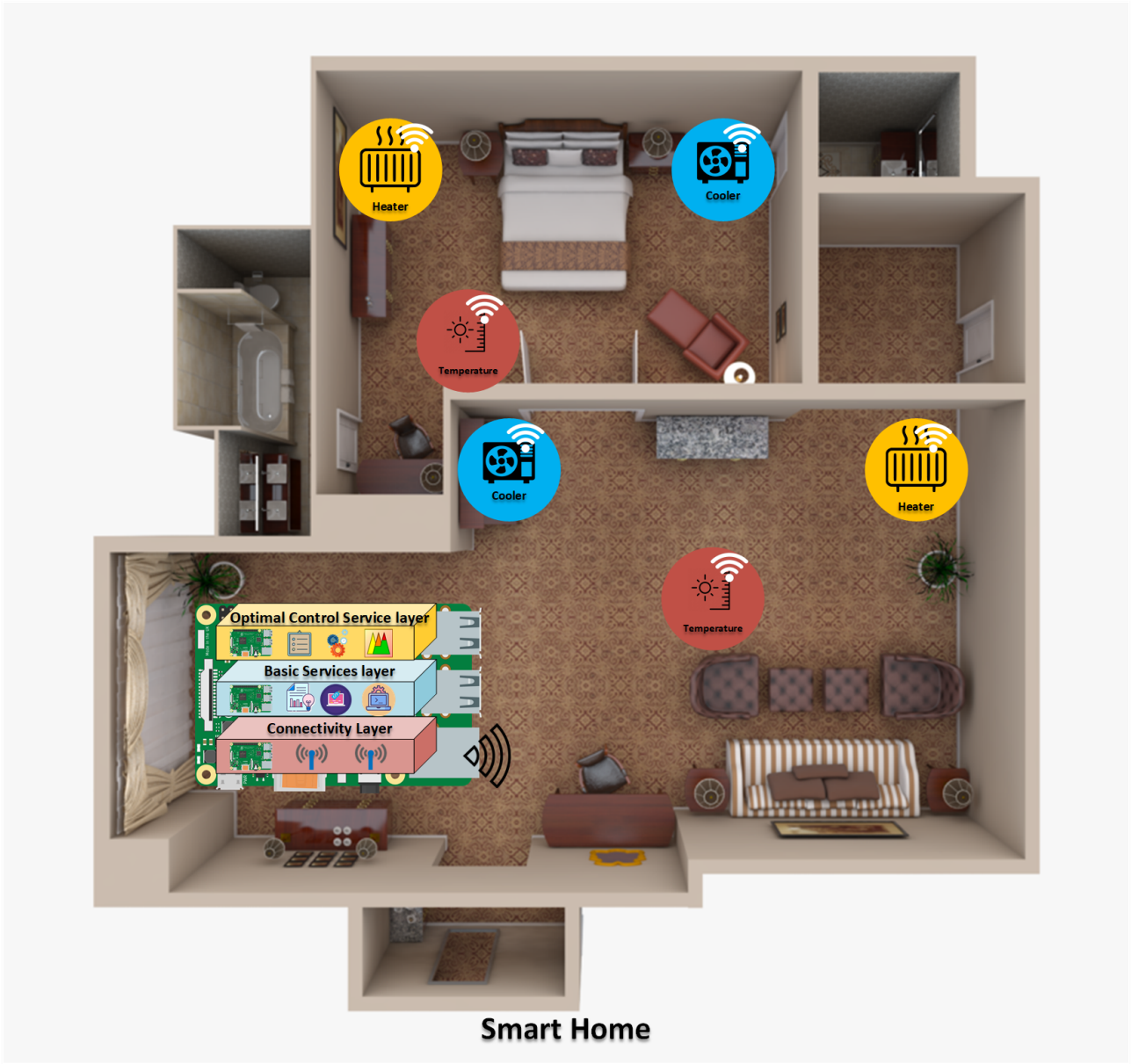


Figure 1. Conceptual Distributed Edge Computing Architecture of a Home Appliances Environment.

The intelligent edge computing platform consists of several fine-grained and self-contained microservices with individual functionalities such as storage service, an intelligent service, processing service, and internet capabilities. The microservices provide services and communicate with each other through a well-defined message interface, such as REST APIs. The microservices can be independently developed with individual technologies and programming languages that can deploy on the embedded system such as raspberry pi. The Raspberry Pi contains a System-on-a-Chip that is as powerful as a regular computer and can run a full operating system like Linux. Hence, It is possible to develop the applications on the machine itself, making embedded development easier.

3.2. Architecture of Distributed Intelligent Edge Computing in Home Appliances Environment

The proposed system functions are separated into three tiers including the client, intelligent edge computing platform, and IoT networks as shown in Figure 2.

The client tier imposes the edge client to provide information about the smart home to the users. It interacts with the intelligent edge computing platform through the visualization capability of the client to present readings of device information and sensing data in various styles.

The intelligent edge computing tier includes a connectivity layer, basic services layer, optimal control service layer, and client support layer to distribute computing processes of IoT devices at the edge of the network based on an intelligent method. In order to derive a user-desired result, the optimization engine and fuzzy control jointly control the IoT device. For interacting with IoT devices, the intelligent edge computing platform provides a command transfer service, and data repository to manage device information, storage sensing data, and issues a command to the device using connectivity layer services to link the IoT devices to the cyber world. In addition, the client support layer services allow users to access the displayed virtual object for accessing the actual device over the Internet.

The IoT network tier is constructed with various small single-board computers, sensing devices, and actuators. The IoT devices are deployed in a smart home to collect data from the physical world and operate the actuator to control electric appliances. Typically, sensors and actuators do not have network functionality. Therefore, to transfer collected data from sensors to the intelligent edge computing platform, they directly connect with IoT devices through the native interface. The single-board computer as an IoT device is able to provide wireless or wired connection ability to communicate with intelligent edge computing framework through internet protocols such as Hypertext Transfer Protocol (HTTP) and Constrained Application Protocol (CoAP).

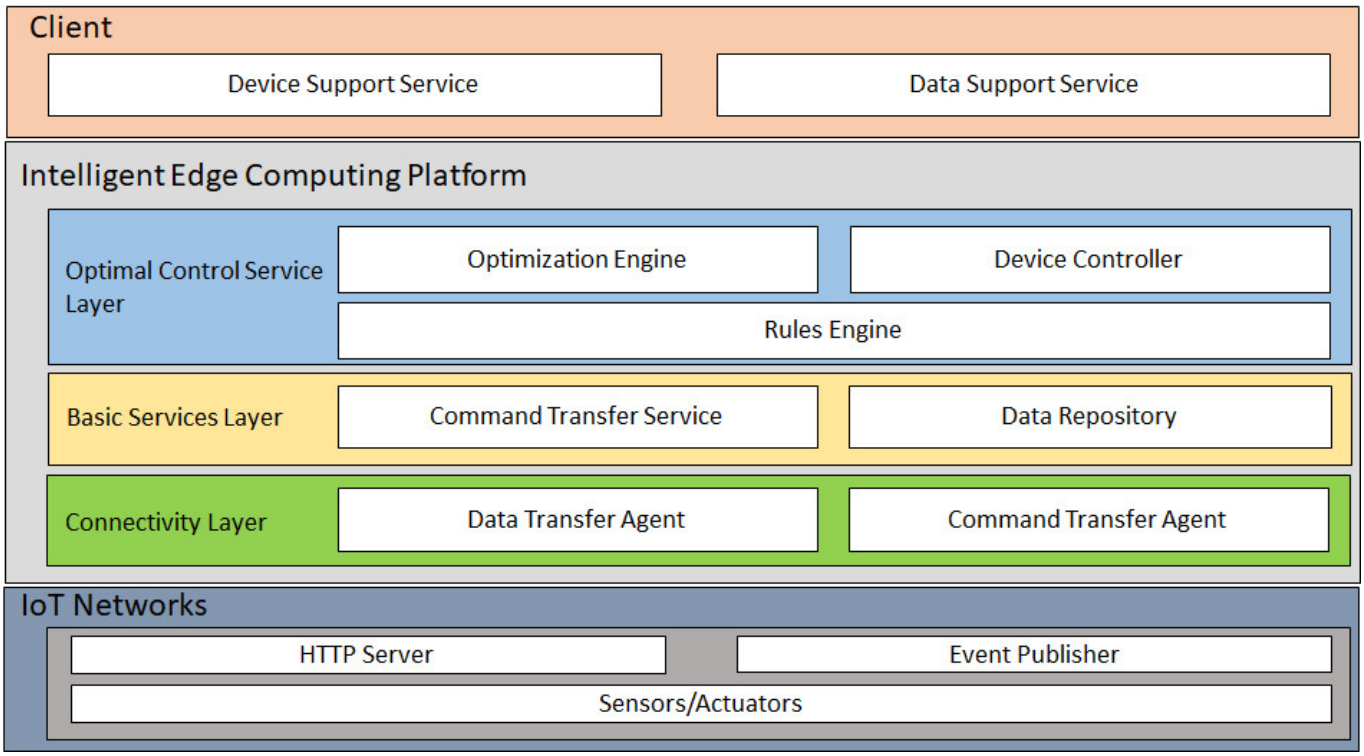


Figure 2. Proposed Distributed Intelligent Edge Computing Functionalities for Home Appliances Control.

3.3. Development Model of Proposed Microservices of Intelligent Edge Computing Platform

Figure 3 illustrates the development model of client services. To improve user convenience, we designed the client services with device support service and data support service. With the device support service, users can gain information about the device list managed by the intelligent edge computing platform, device details provide a detailed profile of the device, and device command that is served by this device. The data support

service provides statistic data, history data, and real-time data which are generated by the devices.

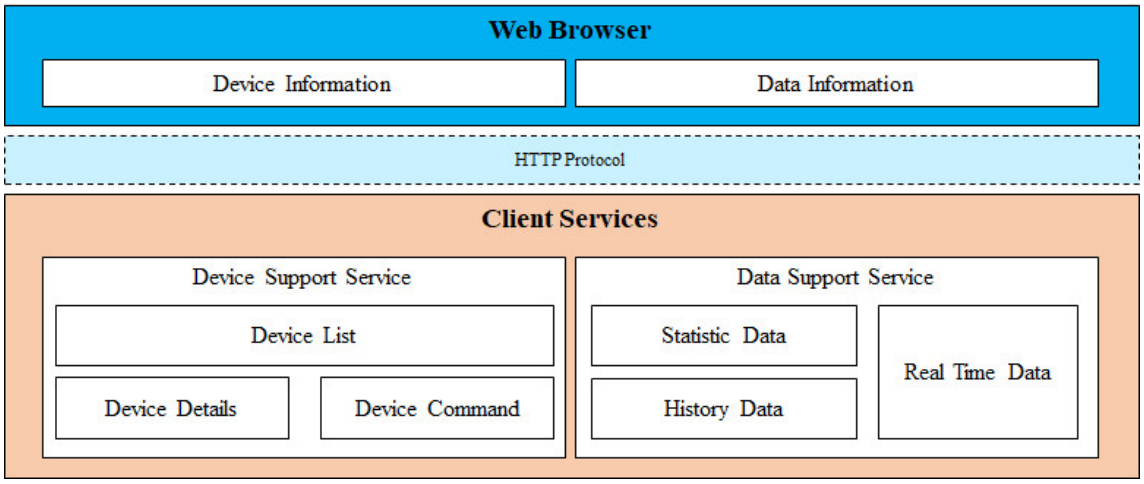


Figure 3. Development Model of Client Services.

Figure 4, illustrates the development model of the intelligent edge computing platform using EdgeX. The model is separated into six functionalities including the device service, EdgeX core data, EdgeX core command, rules engine, optimization engine, and device controller.

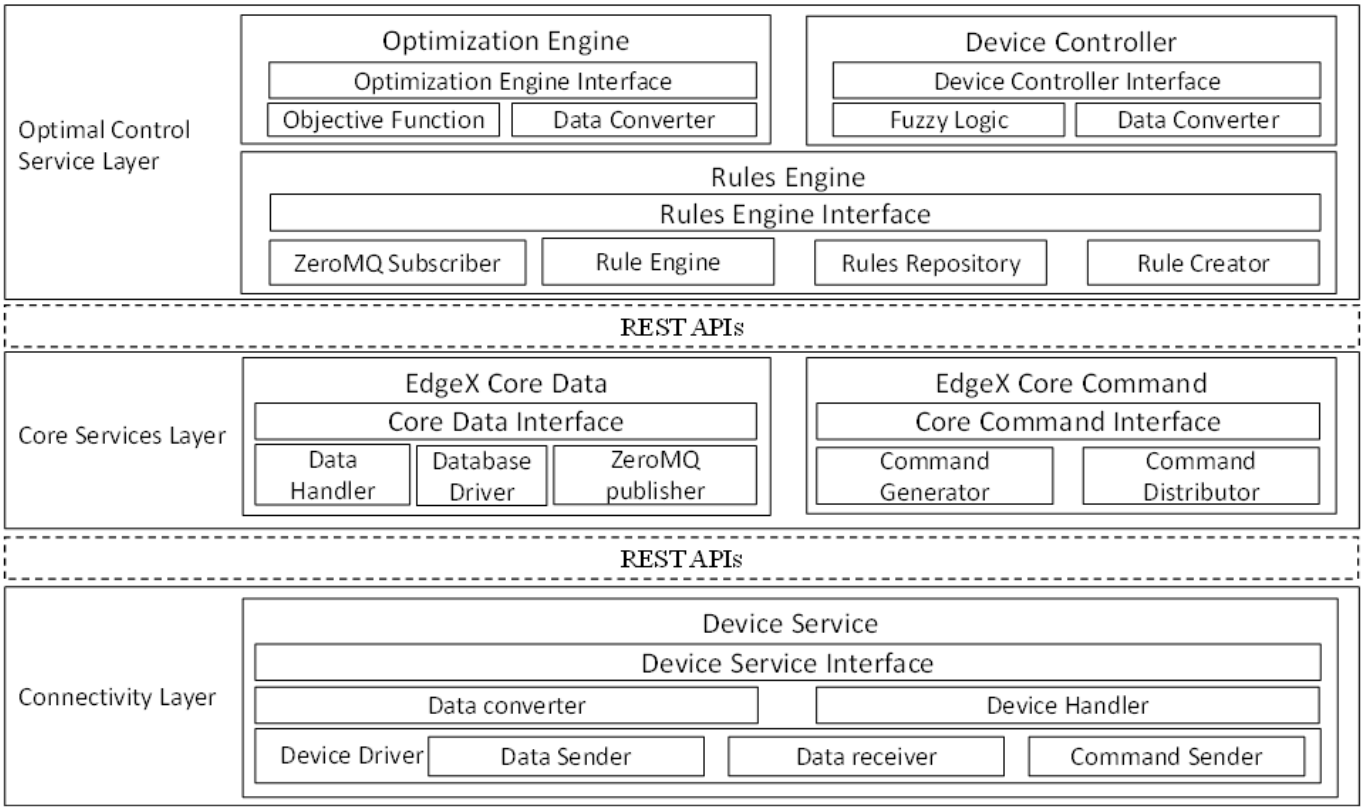


Figure 4. Development Model of the Intelligent Edge Computing Platform based on the Proposed Optimization Mechanism using EdgeX.

The device service is the edge device connector interacting with physical devices that are managed by the intelligent edge computing platform. The sensing data provided by physical devices are ingested to other services and converted into common EdgeX data

formats through device service. The device service exposes REST APIs with the device service interface component to transfer command from other services in the intelligent edge computing platform.

The EdgeX core services such as core data, and the core command are supported by EdgeX. The core data are responsible for storing and managing service for data collected from the device service. The core commands facilitates and controls device actuation requests from other services. To provide the intelligent optimization mechanism in embedded edge computing, we modified the rule engine, and designed an optimization engine and a device controller. The rule engine includes a ZeroMQ subscriber, rule engine, rule repository, and rule creator functions. The rule creator generates rules requested by the users. The generated rules are stored in the rule repository. Afterwards the ZeroMQ subscriber receives flow data provided by the devices and then fed to the rule engine for further processing. The optimization engine consists of an objective function and a data converter. The data converter of the optimization engine converts the data of optimization request to unified formats of the optimization engine and delivers them to the objective function to generate optimized parameters. The device controller is responsible for generating the actuation mechanism of devices based on parameters generated by the optimization engine through fuzzy logic.

To provide sensing data to the intelligent edge computing platform, the simulated IoT device is designed in raspberry pi 3 model B. The designed model is launched on the ubuntu operating system and libraries as an application. The model composes an event generator, event publisher, device handler, and simulated device interfaces. The simulated device interface receives requests from the device service of the intelligent edge computing platform, then invokes proper functions of the device handler to generate an event and finally publishes the event with the event publisher. Detailed Information is depicted in Figure 5.

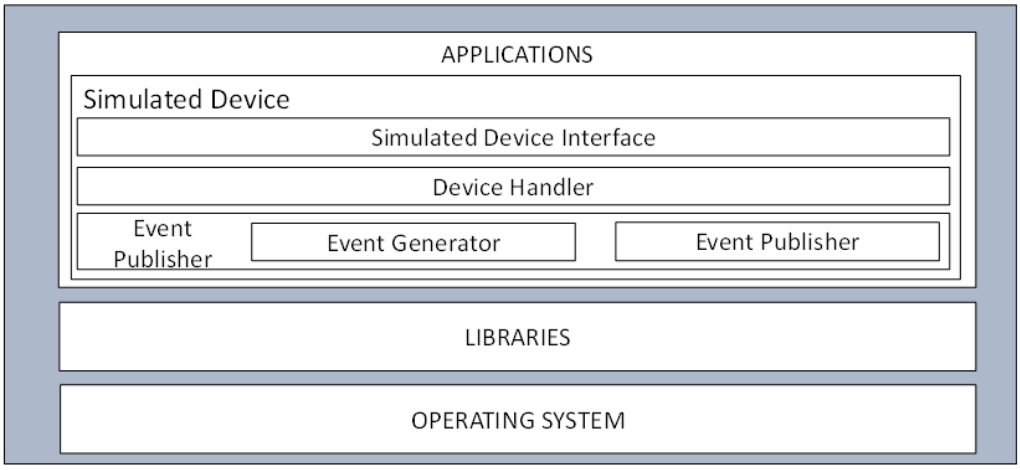


Figure 5. Development Model of the Simulated IoT Device.

3.4. Sequence Diagram of Smart Home Environment Control Using the Intelligent Edge Computing Platform

The sequence diagram of the proposed intelligent edge computing platform for controlling the smart home environment is depicted in Figure 6. The sensors in the IoT networks detect temperature information of a smart home. Then, previously transfer data to the data repository by REST APIs are exposed by the data repository. When the data repository receives the data sent from sensors of IoT networks, it will issue the data with an event and store them. The rules engine is listening to the event with the ZeroMQ subscriber. When the data repository issues an event through the ZeroMQ publisher it will deliver to receivers of this event. If data match any rules that are registered in the rule engine, they will invoke a related function to execute further processing. In our scenario, when the received event brings the current temperature of the smart home, the rules engine

will invoke an optimization engine to obtain the optimized temperature that is a trade-off responds energy consumption and the desired environment setting of residents in a smart home. When an optimization engine response with an optimized temperature, the rules engine invokes the device controller to properly control the smart home. Finally, the device controller calls the command transfer service to send the commands to actuators of IoT networks to maintain the status of the smart home.

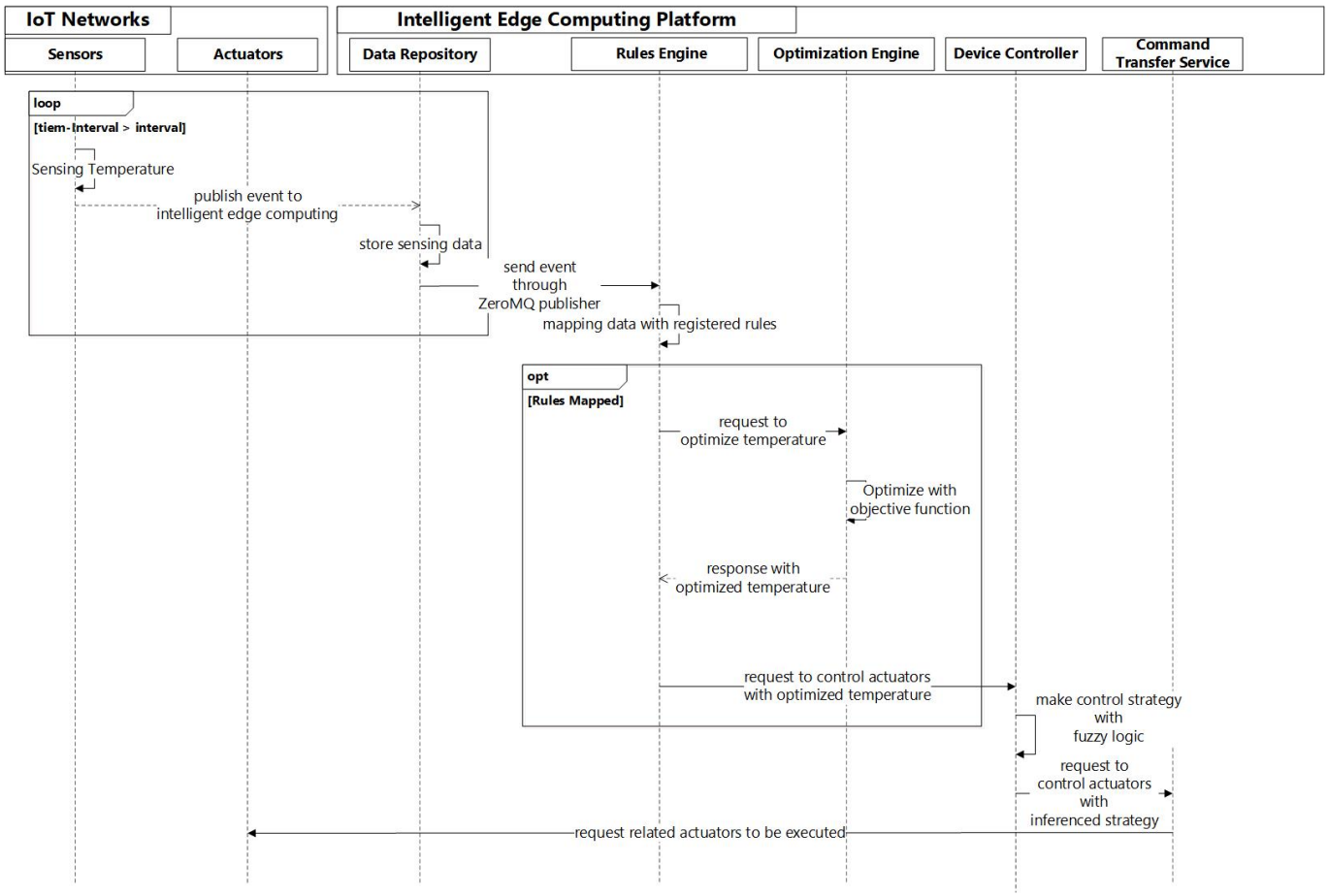


Figure 6. Optimization Mechanism-based Sequence Diagram for Home Appliances Control.

3.5. Proposed Energy Efficiency Objective Function Model for Home Appliances Control

We illustrate the block diagram of the proposed objective function model of the optimization mechanism in Figure 7. The proposed model is comprised of input data, output, system constraints, temperature requirements, and an optimization engine. The input data inject the current temperature into the optimization engine; with the system constraints and temperature requirements of users, the optimization engine generates optimized temperature.

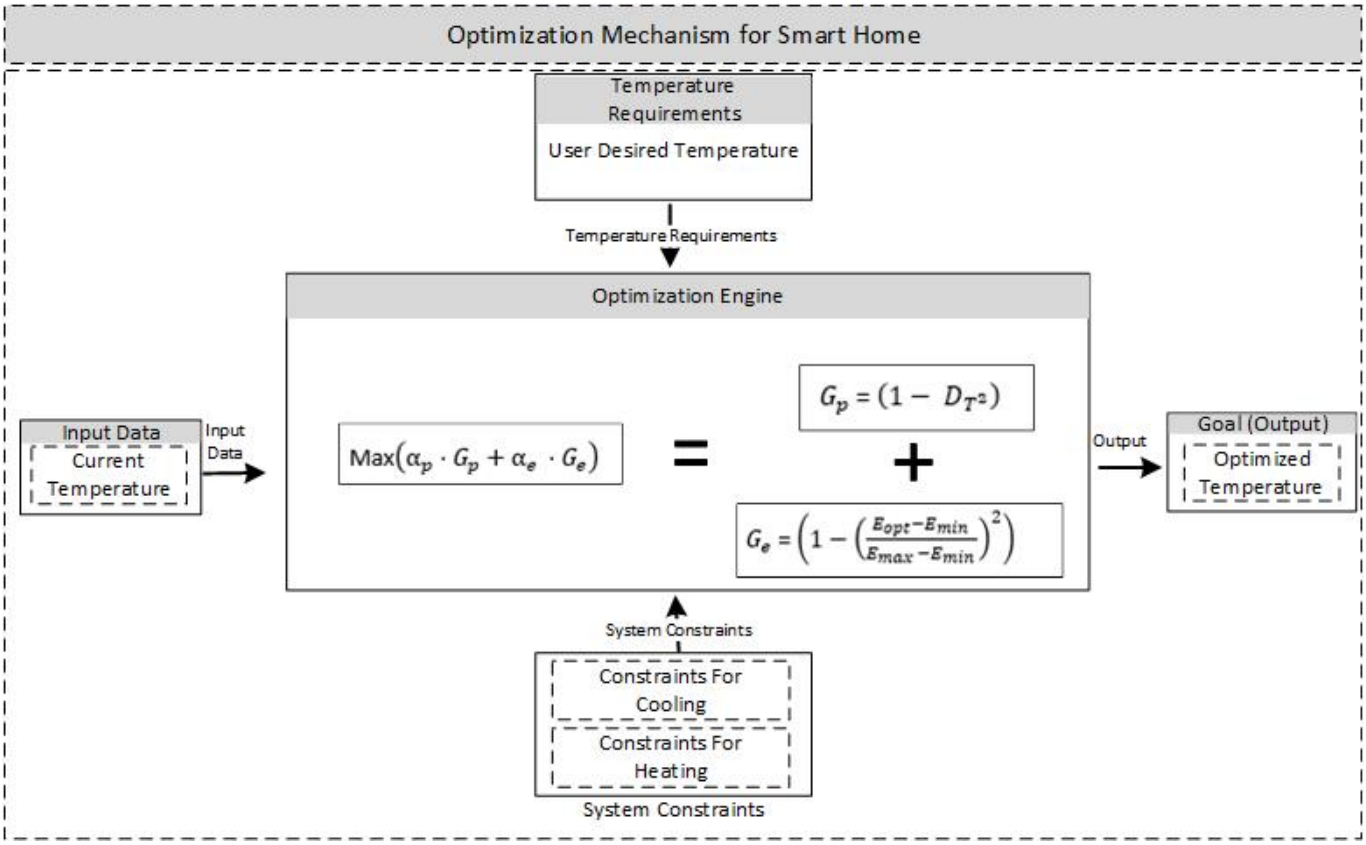


Figure 7. Objective Function Model of the Intelligent Optimization Mechanism.

In order to infer the functional expression, the following assumptions are made. The current sensing data for temperature in the smart home are expressed as T_c. In order to express the user's acceptable range of temperatures, we denoted them as T_d. The user preferred maximum and minimum values for temperature are represented as $T_d = [T_{min}, T_{max}]$. It is assumed that within the expected range, the maximum value represents the maximum expected value of the user's comfort temperature, and the minimum value represents the minimum acceptable value of the smart home parameter for indoor spaces. If energy costs are not an influencing factor, the occupant wants certainly to keep the indoor parameters in a smart home at their maximum level. Setting the indoor parameters to the maximum value in a smart home would incur higher energy costs, so there is a trade-off between the energy consumption and the environmental settings required in the smart home. To be concise, we set the length of the temperature range as $\triangle T = [T_{max} - T_{min}]$. In this scenario, two smart home actuators (heater, cooler) should be used to maintain the required temperature settings in the smart home. Moreover, there are constraints to operate actuators, such as if $T_c < T_{min}$, then the heater is operated to increase smart home temperature. Similarly, if $T_{max} > T_c$, then the cooler will be operated to cool down the indoor temperature. We considered that the consumption by the heater and cooler in per unit change in temperature is estimated by P_h and P_c, respectively. Being unaffected by the high-cost constraint associated with more power consumption, the user can simply set temperature as per their comfort zones and the user desired parameters but, in practice, we cannot afford this all the time. Therefore, we need to find some trade-off between the user desired setting in the smart home and energy consumption. Table 2 presents a brief description of the various notations used in this formulation.

Table 2. Proposed Objective Function Notations.

Notation		Description
SHP_d	T_d $\frac{T_{min}}{T_{max}}$	Desired smart home parameter for Temperature with ranges
SHP_c	T_c	Current smart home parameter for Temperature
SHP_o	T_o	Optimal smart home parameter for Temperature
Energy Consumption	E_{opt}	Total energy consumption in optimal conditions
	E_{min}	Total energy consumption in minimum parameter settings
	E_{max}	Total energy consumption in maximum parameter settings
α_p		User preference for indoor parameter settings
α_e		User preference for energy saving
P_a		Power consumption in per unit change in indoor parameters $\alpha \in \{heater, cooler\}$
D_T		Deficiency in achieving optimal settings for parameter T
G_p		Gain in optimal parameter settings
G_e		Gain in energy saving

We assume T_o is the optimal parameter that can compromise the required environmental settings and energy consumption. With the user's acceptable range of temperatures, the optimal parameter $T_o \in [T_{min}, T_{max}]$. With the above assumption the total energy to maintain the optimal setting E_opt in the smart home is expressed with the following formula.

$$E_{opt} = \begin{cases} P_h * (T_o - T_c) & \text{if } T_c < T_{min} \\ P_c * (T_c - T_o) & \text{if } T_{max} < T_c \end{cases} \quad (1)$$

Similarly, we can obtain the minimum and maximum power consumption with the following formula.

$$E_{min} = \begin{cases} P_h * (T_{min} - T_c) & \text{if } T_c < T_{min} \\ P_c * (T_c - T_{max}) & \text{if } T_{max} < T_c \end{cases} \quad (2)$$

$$E_{max} = \begin{cases} P_h * (T_{max} - T_c) & \text{if } T_c < T_{min} \\ P_c * (T_c - T_{min}) & \text{if } T_{max} < T_c \end{cases} \quad (3)$$

We want to attain the maximum comfort status of the smart home with minimum energy consumption. We specify that G_p and G_e are the gain in making optimal environmental settings and gain in energy saving, respectively. If the user-defined preferences for the optimal environmental setting and energy saving are α_p and α_e, respectively, then the objective function is represented through the following equation.

$$Max(\alpha_p G_p + \alpha_e G_e) \quad (4)$$

There is a constraint which is $\alpha_p + \alpha_e = 1$. The gain in having the optimal environmental setting G_p can be expressed as follows.

$$G_p = (1 - D_{T^2}) \quad (5)$$

There is D_T which is the deficiency in having an optimal setting for temperature. If we want to maximize the gain through the optimal preference setting G_p, the deficiency

component for the temperature parameter should be minimized. The formula is given by the following.

$$D_T = \begin{cases} \frac{T_{max} - T_o}{\triangle T} & \text{if } T_c < T_{min} \\ \frac{T_o - T_{min}}{\triangle T} & \text{if } T_{max} < T_c \end{cases} \tag{6}$$

When $T_c < T_{min}$ then the optimal preference setting should be to set $T_o \approx T_{max}$. When $T_o \to T_{max}$ then $D_T \to 0$, that will help in maximizing the desired G_p. The gain in energy saving G_e can be expressed as the following formula.

$$G_e = \left(1 - \left(\frac{E_{opt} - E_{min}}{E_{max} - E_{min}}\right)^2\right) \tag{7}$$

If we want to maximize energy saving, the energy consumed required for optimal setting should be minimized, i.e., E_{opt}. When $E_{opt} \to E_{min}$ then $G_e \to 1$, which is helpful to maximize the energy-saving component G_e. Finally, the objective function can be expressed with the following formula.

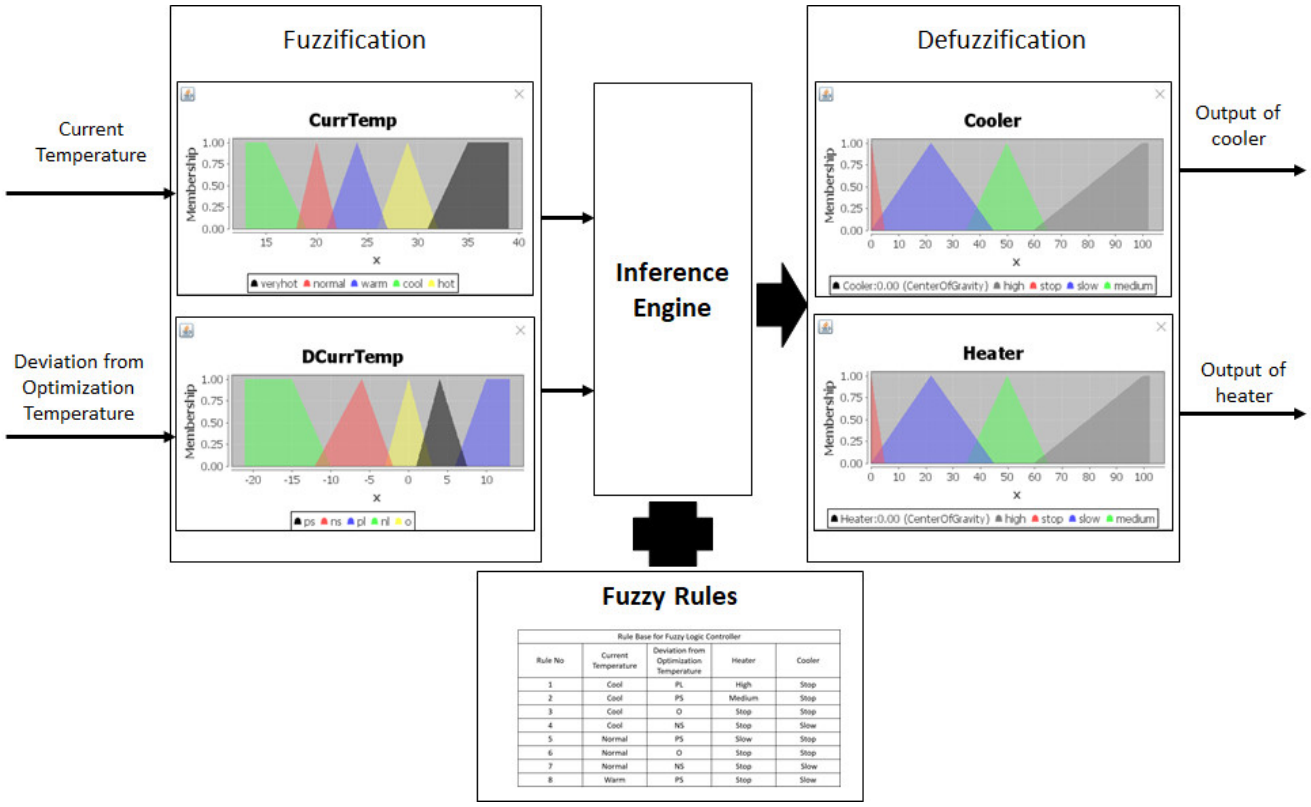
$$Max\left(\alpha_p\left(1 - D_{T^2}\right) + \alpha_e\left(1 - \left(\frac{E_{opt} - E_{min}}{E_{max} - E_{min}}\right)^2\right)\right) \tag{8}$$

There are some constraints such as :
For the cooling case : $T_c < T_{min} \leq T_o \leq T_{max}$
For the heating case : $T_{min} < T_o \leq T_{max} < T_c$
$0 < E_{min} \leq E_o \leq E_{max}$

3.6. Proposed Fuzzy Logic-Based Home Appliances Control Model

The given Figure 8, illustrates a fuzzy knowledge-based model that is used to control the smart home environment. The fuzzification comprises of the process of transforming values such as current temperature, deviation from optimized temperature into grades of membership for linguistic terms of fuzzy sets. The current temperature is divided into a fuzzy set such as Cool, Normal, Warm, Hot, and Very Hot. The deviation from the optimization temperature is divided into a fuzzy set that includes Negative Long (NL), Negative Short (NS), Optimal (O), Positive Short (PS), Positive Long (PL). The fuzzy rule-based inference engine performs the inference operations based on the rules with input values to map corresponding linguistic terms of heater and cooler. The defuzzification transforms the fuzzy results of the fuzzy rule inference engine into a crisp output.

The Algorithm 1 represents an algorithm that generates the level of cooling and heating using the current temperature and deviation from the optimized temperature. In order to generate a proper level of cooling and heating, we leverage a fuzzy logic. Fuzzy logic is much closer in spirit to human thinking and natural language than the traditional logical systems. Basically, it provides an effective means of capturing the approximate, inexact nature of the real world. The Fuzzy logic takes the current temperature and its deviation from optimized temperature as inputs and outputs the appropriate heating and cooling levels based on the fuzzy rules. Finally, an output class is created to include the result. Through the fuzzy logic-based method, the device control mechanism could provide a proper control level of the cooler and heater.



Rule Base for Fuzzy Logic Controller				
Rule No	Current Temperature	Deviation from Optimization Temperature	Heater	Cooler
1	Cool	PL	High	Stop
2	Cool	PS	Medium	Stop
3	Cool	O	Stop	Stop
4	Cool	NS	Stop	Slow
5	Normal	PS	Slow	Stop
6	Normal	O	Stop	Stop
7	Normal	NS	Stop	Slow
8	Warm	PS	Stop	Slow

Figure 8. Home Appliances Control Mechanism Using Fuzzy Logic.

Algorithm 1: Device Control Algorithm.

Input: current temperature, and optimized temperature.
Output: level of cooler, and heater.
Initialize double *dot* to store deviation from the optimization temperature.
Initialize double *ht* to store the level of the heater.
Initialize double *cl* to store the level of the cooler.
Initialize class *output* to store the level the of cooler and heater.
Initialize fuzzy logic inference engine as fis to generate the level of the cooler and heater.
$dot =$ optimized temperature – current temperature;
fis set the variable current temperature with the current temperature;
fis set the variable store deviation from the optimization temperature with dot;
fis evaluate to generate the level of the cooler and heater;
$ht = fis$ obtain the variable of the heater;
$cl = fis$ obtain the variable of the cooler;
output set the heater with *ht*;
output set the cooler with *cl*;
return *output;*

The following Table 3, is the membership function for the current temperature. We assume the proposed fuzzy logic controller of temperature works perfectly at any temperature within range 13 °C–39 °C.

Table 3. Membership Functions for Current Temperature.

Membership Functions	Range (°C)
Cool	13–19
Normal	18–22
Warm	21–27
Hot	26–32
VeryHot	31–39

The following Table 4, is the membership functions for deviation from the current temperature. It gives the difference between the user's preferred temperature and the current temperature of the room as recorded by the temperature sensor in the room. As this model can work in temperature range 13–32 °C and the user can set the desired temperature from 18 to 26 °C, so the temperature difference between the current and the optimized temperature cannot exceed −21°C (18 °C–39 °C) and 13 °C (26 °C–13 °C). Thus, (−21 °C) and (+13 °C) are the lower and upper limits of the input variable "deviation from optimized-temperature".

Table 4. Membership Functions for Deviation from the current Temperature.

Membership Functions	Range (°C)
NL	−21 to −10
NS	−12 to −2
O	−3 to 3
PS	1 to 7.5
PL	6 to 13

The heater can either be in the ON or OFF state depending on the temperature preference in the room. The heater settings are categorized into 1. STOP 2. SLOW 3. MEDIUM 4. HIGH. If the current temperature of the room is below the desired temperature, the heater is turned on automatically according to temperature difference. The detailed information is presented in Table 5.

Table 5. Membership Functions for the Heater.

Membership Functions	Range (%)
Stop	0–5
Slow	0–45
Medium	35–65
High	60–100

The cooler settings are categorized into 1. STOP 2. SLOW 3. MEDIUM 4. HIGH. If the current temperature of the room exceeds the desired temperature then this cooler turns on automatically according to the temperature difference. The detailed information is presented in Table 6.

Table 6. Membership Functions for the Cooler.

Membership Functions	Range (%)
Stop	0–5
Slow	0–45
Medium	35–65
High	60–100

4. Implementation and Result

In this chapter, we present the development environment for implementing the proposed system and the detailed implementation results in pictures.

4.1. Development Model of Implemented Microservices of Intelligent Edge Computing Platform

Figure 9 illustrates the developed functionalities of intelligent edge computing platforms using EdgeX. The intelligent edge computing platform contains several microservices. The device microservice for communicating with IoT devices is located between the intelligent edge computing platform and the IoT device for transmitting data or commands in both directions. EdgeX core services provide basic services required for an intelligent edge computing platform. The core data manage and stores data, core metadata manage information of IoT devices, and all services related to the intelligent edge computing platform while the core metadata manage and deliver commands through REST APIs to the services provided by IoT devices. The rule engine, which provides basic intelligence in the intelligent edge computing platform, provides basic judgment to the intelligent edge computing platform based on pre-registered rules. In controlling the environment of a smart house, we propose an optimization engine and a fuzzy controller that supports the rule engine to minimize energy consumption while maintaining user preferences. Based on the currently collected temperature information, the optimization engine provides an optimized temperature that can satisfy user requirements while minimizing energy consumption, and the fuzzy-based controller properly controls the temperature environment of the smart house based on data of both the optimization, desired temperature and the current temperature.

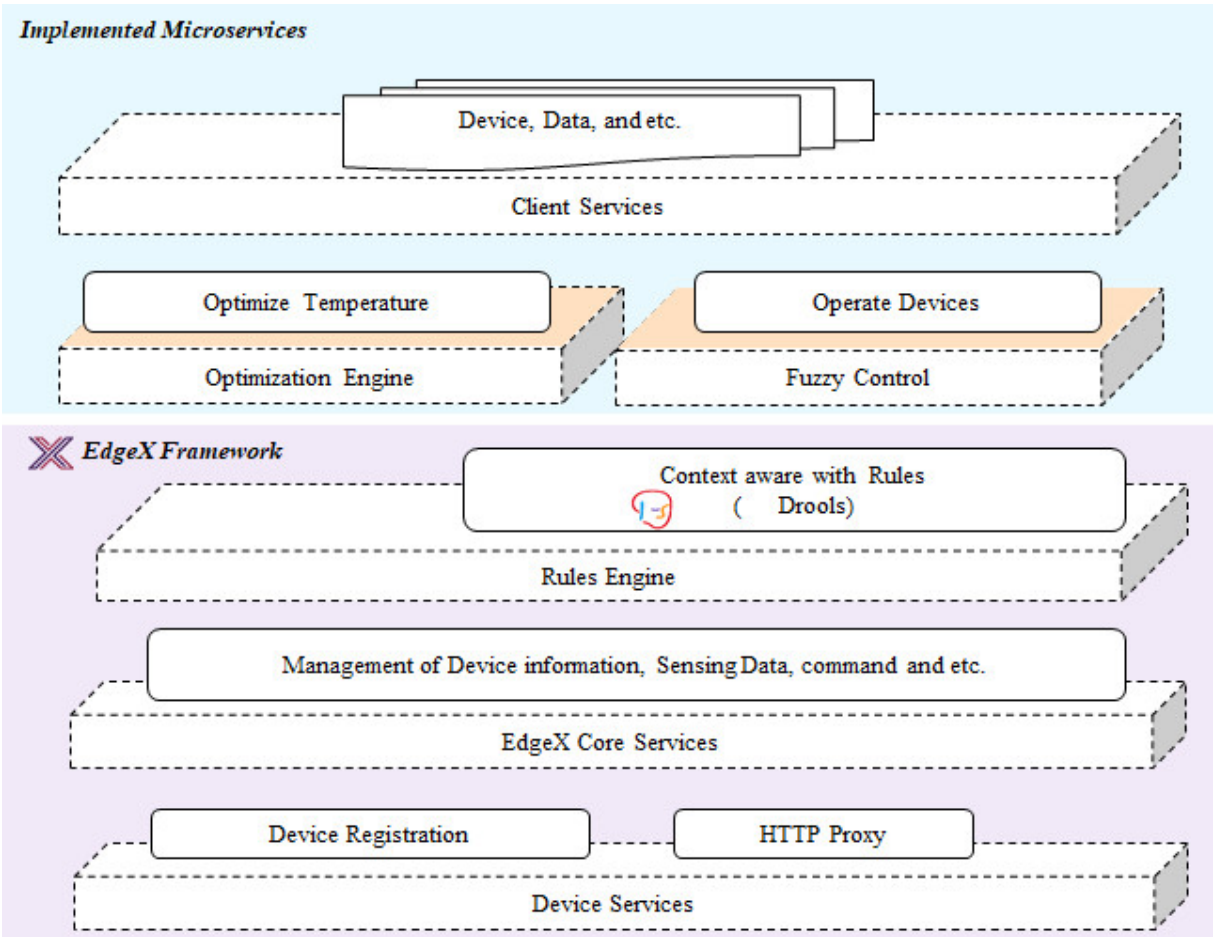


Figure 9. Developed Functionalities of the Intelligent Edge Computing Platform Using EdgeX.

4.2. Development Environment Specifications of the Intelligent Edge Computing Platform

To implement the embedded service, the optimization engine and device service were implemented on the desktop, and the execution environment was installed on the Raspberry. Optimization engine was implemented in C# using ASP.NET Core framework, and device service was implemented in Java language using Spring boot framework. The detailed specifications are depicting in Table 7.

Table 7. Develop and Executing Specifications of Intelligent Edge Computing Platform

Environment	Category	Items	Descriptions
Develop Environment	Hardware (Desktop)	OS	Windows 10
		CPU	Intel® core™ i5-8500
		Memory	8 GB
		Hard Disk	150 GB
	Software	ASP.NET Core, Swashbuckle	A framework to develop web applications
		Spring Boot	A framework for modern Java-based enterprise applications
Executing Environment	Hardware (Raspberry Pi 3)	OS	Ubuntu 20.04 64 bit
		CPU	Quad Core 1.2 GHz Broadcom BCM2837 64 bit CPU
		Memory	1 GB
		MicroSD Card	16 GB
	Software	Docker and Docker Compose	A platform to develop, ship, and run applications

4.3. Implemented Results of Intelligent Edge Computing Platform

As shown in Figure 10, the client services and intelligent microservices are deployed in the corresponding platform to provide services.

Figure 10a, the edge client service is a web client-server that provides the information through UIs to users. During experimentation, the Chrome web browser runs on a laptop to access the intelligent edge computing platform through the Internet. The UIs are provided by the client support provider from the intelligent edge computing platform. Furthermore, the Bootstrap and Jquery libraries are used for implementing the contents of UIs based on the Spring Boot framework.

Figure 10b, the optimization engine is developed with C# based on ASP.NET Core to provide microservices. The fuzzy control is developed with Java-based on the Spring boot framework to provide microservices. In the experiment, these modules are deployed in Raspberry Pi 3 that includes 1 GB memory to runs the Ubuntu 64 bit OS smoothly.

Figure 11, shows the deployed microservices in the corresponding platforms. Figure 11a, Spring Boot 2.1.4 is used for developing a rules engine to provide microservices. Figure 11b, EdgeX core services are implemented using Go to provide web services through multiple microservices providers. Figure 11c, the device service is implemented using Go based on the implementation template from the EdgeX foundry. The IoT device is a java application that runs on the ubuntu 20.04 operating system based on Raspberry Pi 3. We develop a temperature generation application with Java and expose that with REST APIs. When a user requests to generate random temperature, the application will be publishing evens to the core data service of the intelligent edge computing platform to trigger the fuzzy control approach based on the rules engine.

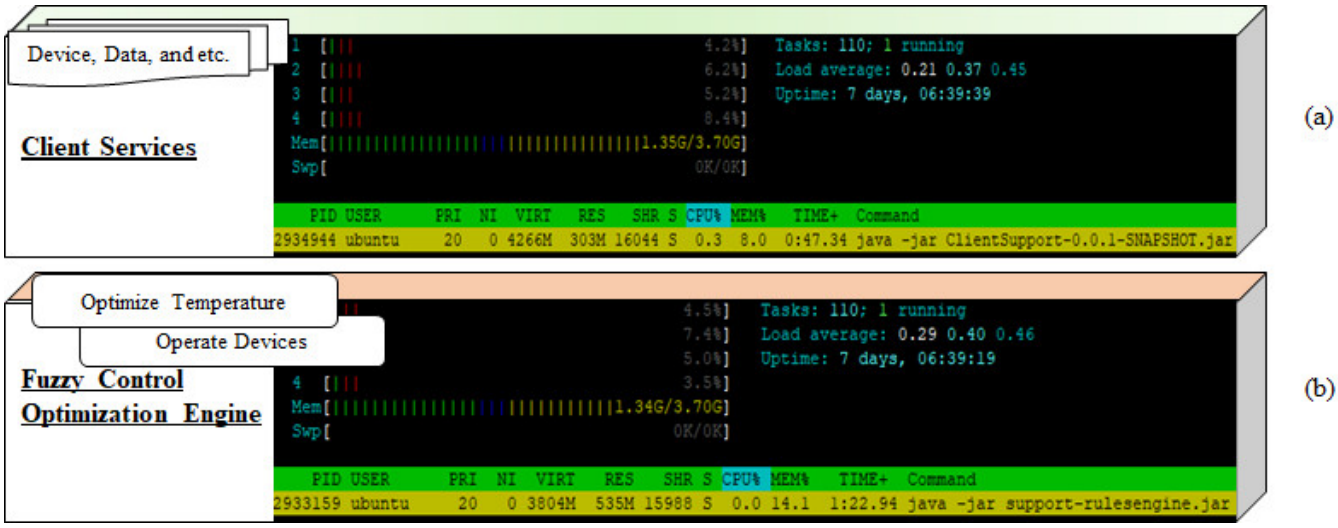


Figure 10. Implemented Client Services and Intelligent microservices of the proposed embedded edge computing. (**a**) Executed Client Services. (**b**) Executed Optimization Engine and Fuzzy Control.

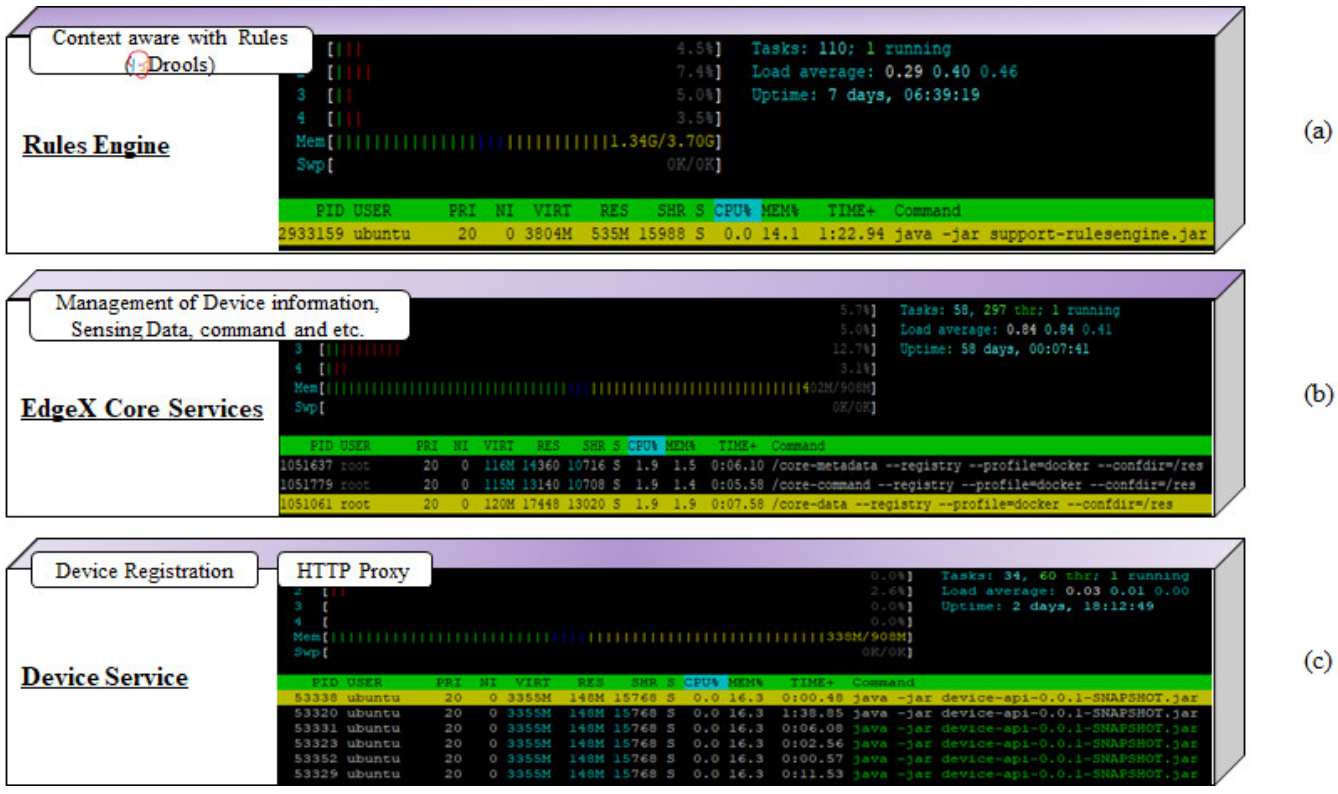


Figure 11. Implemented Rules Engine and Device Service microservices of the proposed embedded edge computing. (**a**) Executed Rules Engine. (**b**) Executed EdgeX Core Services. (**c**) Executed Device Service.

Figure 12, shows an implementation result of device management using a user interface of the web client service. The user interface displays the device list page that includes a device list and each item presents a device ID, name, and URI. The item of the device list provides the function of accessing the device detail page that presents detailed information of the clicked item. The button detail provides device detail page that presents detailed device information. The detailed device information is retrieved by the selected device ID.

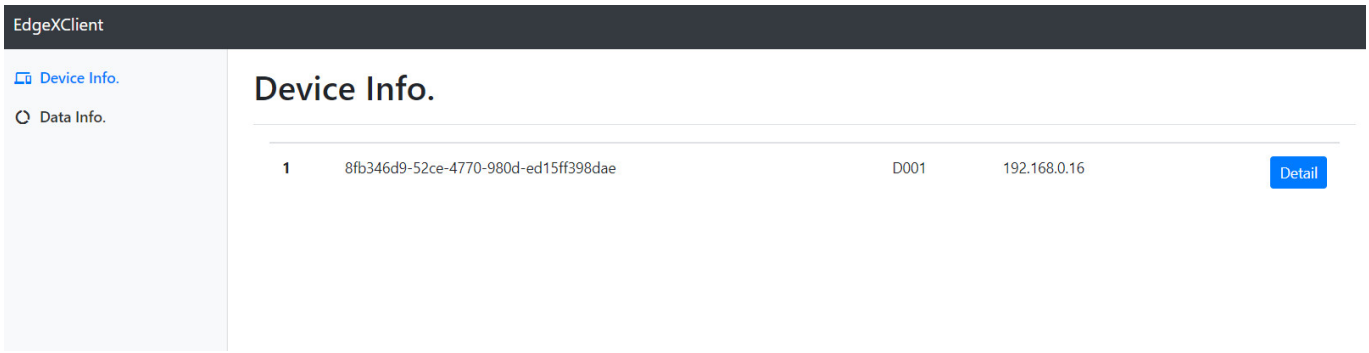


Figure 12. The Implemented Result of Device List of Client Services.

Figure 13, shows an implementation result of device management for presenting device detail information to the user using a user interface of the web client service. The user interface displays the device detail page that includes device detail including the device's ID, name, profile name, service name, and resource list. The information is retrieved by the selected device ID from the Device List Page. The information is delivered as a JSON format data provided from Metadata of EdgeX framework. The detailed information of a device includes properties of the device and a list of resources. Each resource in the list has two types of method used for accessing the resource to obtain the sensing data and control actuator.

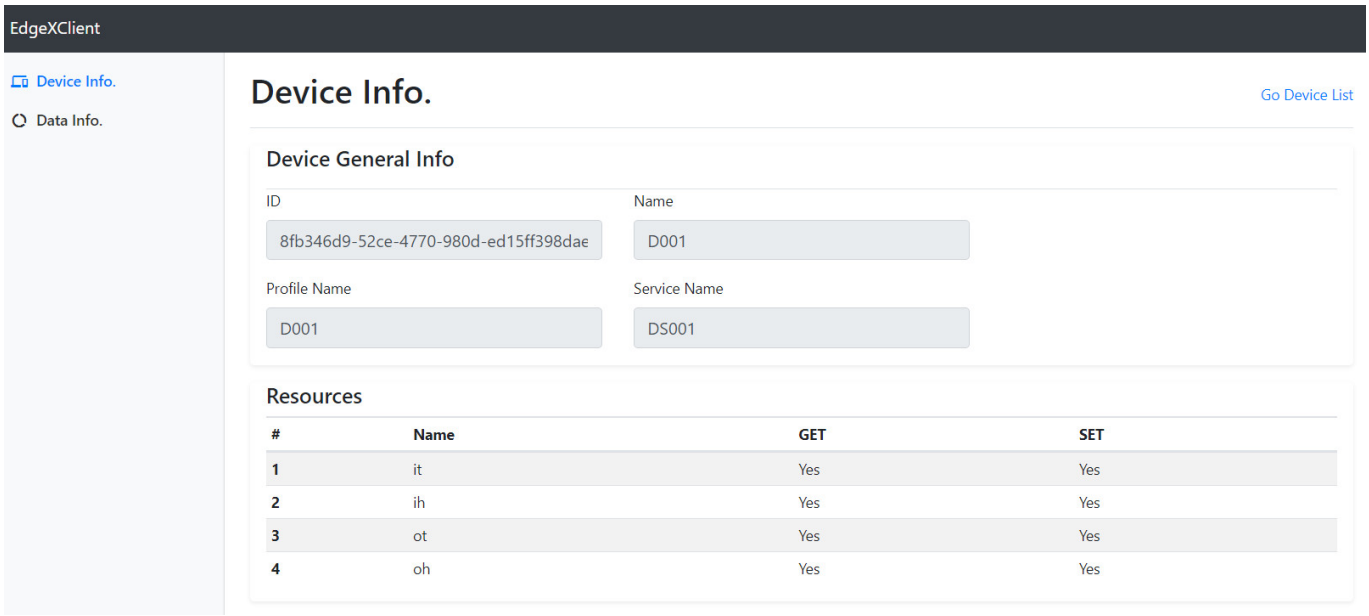


Figure 13. The Implemented Result of the Device Detail of Client Services.

Figure 14 shows the implementation result for presenting history data of a device that is selected from the device list. The selected device may have multiple resources, and data of resources are sent to the EdgeX framework through event publishing. The data are stored in the core data service of EdgeX framework, and the Web Client requests the data from the EdgeX framework.

Figure 15, represents the implementation result for presenting statistical data of a device that is selected from the device list. For each resource of the selected device, the history data are requested and calculated to be statistical data. Once a device is selected for presenting the statistical data, the Web Client requests the EdgeX framework to obtain the history data and results from the statistical data for each resource. The first statistic bar presents an average of the retrieved data, the second is a standard deviation, the third is the maximum, and the fourth one is the minimum.

Figure 14. The Implemented Result of History Data of Client Services.

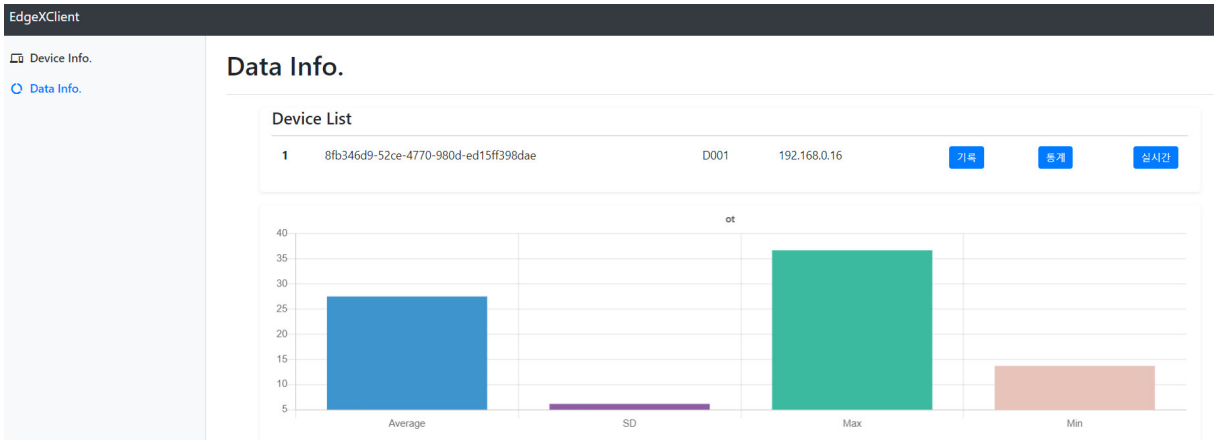


Figure 15. The Implemented Result of Static Data of Client Services.

Figure 16, shows the implementation result for presenting real-time data of a device that is selected from the device list. For presenting the data of each resource, the Web Client requests to the device and obtains the real-time data. For accessing the selected device in real-time, the Web Client requests the EdgeX framework to gain device information for sending a command to the device. The real-time data are collected by requesting the resources of the selected device. In this process, the Web Client requests the device periodically; therefore, it consumes higher network resources.

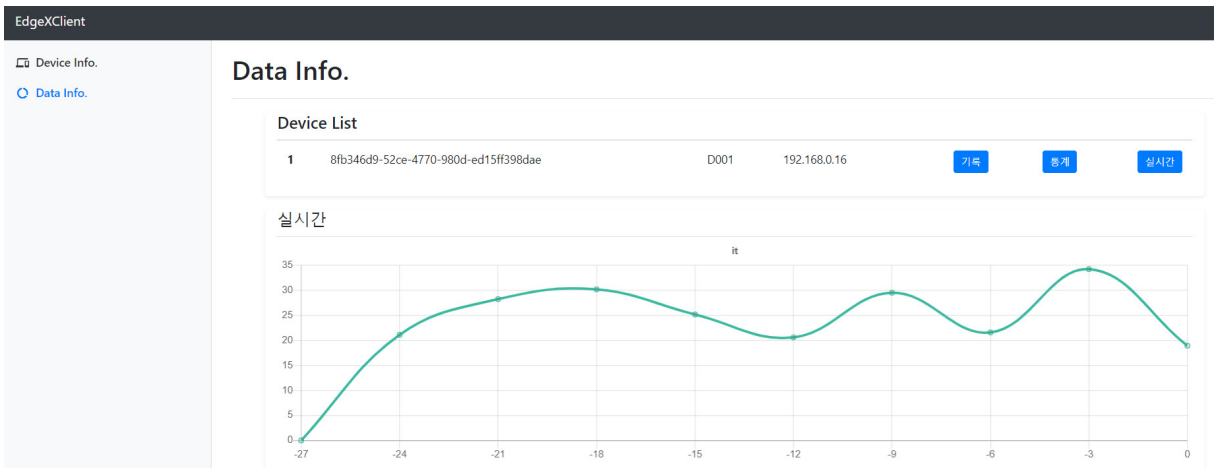


Figure 16. The Implemented Result of Real-Time Data of Client Services.

5. Performance and Evaluation

Figure 17, illustrates the memory usage of the proposed microservices of the system on the Raspberry Pi 3. We use two Raspberry Pi 3 pieces of hardware, as shown in Figure 17a,b, to execute the system. The total memory of each device is 929,996 kb, in which the running process takes 196,988 kb, and 281,128 kb, the total available memory is 1,332,356 kb. According to Figure 17c, the total memory of the microservices is 964,024 kb, and hence we have enough memory to run the microservices. The rules engine, client service provider, fuzzy controller, and simulated device microservices take most memory usage. All the services are developed with a java-based spring boot framework that supports REST APIs for other services and clients. The optimal engine takes 63,532 kb to provide optimization capability to the intelligent edge computing framework and developed with the ASP.net Core framework which is based on the C# language. The EdgeX core services, such as core data, core command, and core metadata running on the Docker container are implemented with Go to provide basic services for the intelligent edge computing framework through consuming 44,948 kb of memory. The device service is also implemented with Go to provide services that take 3536 kb of memory. According to the statistics, it is evident that Go-based services consume less memory compared to C#- and Java-based services.

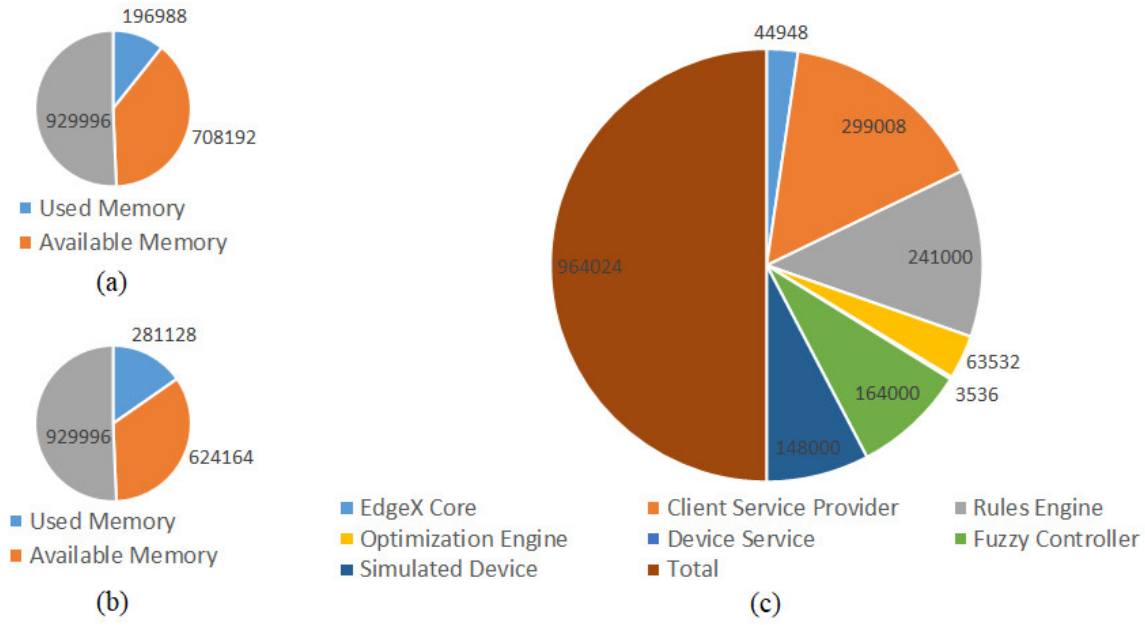


Figure 17. Proposed Microservices Memory Usage on Raspberry Pi 3. (**a**) A Raspberry Pi 3. (**b**) A Raspberry Pi 3. (**c**) Total Memory of Microservices.

Figure 18, shows request delays for executing the proposed system based on the intelligent edge computing framework in the network edge. For evaluating the performance of the executing latency, the executing time of the optimization engine and fuzzy control is measured.

Figure 18a presents the execution time of the optimization engine that is requested by the rules engine. The optimization engine is implemented with the C# language to provide optimization capability to the intelligent edge computing framework. According to the statistics, we can understand that the minimum time is 0.152 s, averaging at 0.177 s, and the maximum time is 0.207 s.

Figure 18b presents the execution time of fuzzy control that is invoked by the rules engine. The fuzzy control is implemented with the java language to provide a control

function to the managed devices of the intelligent edge computing framework. According to the statistics, the minimum time is 0.169 s, average time is 0.199 s, and the maximum time is 0.252 s.

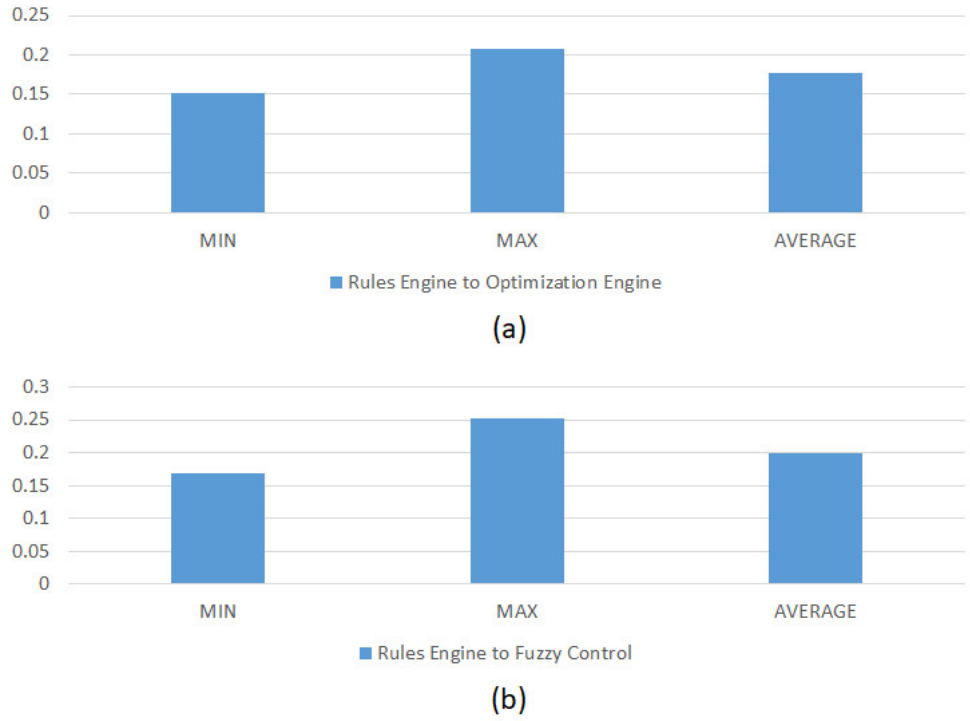


Figure 18. Proposed Microservices Memory Usage on Raspberry Pi 3. (**a**) Optimization Engine. (**b**) Fuzzy Control.

6. Conclusions

In this paper, We conducted research on improving the intelligence of smart control systems and supplementing the inherent deficiencies of IoT using embedded edge computing and rule-based objective functions. In order to control the smart home temperature according to user desired parameters and the comfort index, an optimization engine and fuzzy control service are implemented. The optimization engine is developed with the C# language to provide optimization ability to intelligent edge computing. The fuzzy control service is developed with the java language to support the control mechanism to the devices managed by intelligent edge computing. When looking at the simulation results, it can be seen that all services that control the temperature of the smart home operate automatically on the two Raspberry devices without human intervention. We have overcome the deficiencies of IoT with edge computing and have improved smart home controller intelligence with a rule-based optimization mechanism so that it can be executed automatically without human intervention. In future work, we are considering implementing the optimization algorithm with other factors affecting the smart home and testing it in a real smart home environment.

Author Contributions: R.X. conceived the idea for this paper, implemented the optimization engine and device control based on fuzzy logic to provide a trade-off between energy consumption and the desired environment setting of residents in a smart home, designed the experiments, and wrote this paper; W.J. assisted in implementing the client services; Y.H. assisted in implementing the device control service; D.-H.K. conceived the overall idea of the paper and proofread the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2021-0-00188, Open source development and standardization for AI enabled IoT platforms and interworking), and

this research was supported by Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2018-0-01456, AutoMaTa: Autonomous Management framework based on artificial intelligent Technology for adaptive and disposable IoT). Any correspondence related to this paper should be addressed to Dohyeun Kim.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cisco. Cisco Annual Internet Report (2018–2023) White Paper. 2020. Available online: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (accessed on 10 June 2021).
2. Bresciani, S.; Ferraris, A.; Del Giudice, M. The management of organizational ambidexterity through alliances in a new context of analysis: Internet of Things (IoT) smart city projects. *Technol. Forecast. Soc. Chang.* **2018**, *136*, 331–338. [CrossRef]
3. Mehmood, Y.; Ahmad, F.; Yaqoob, I.; Adnane, A.; Imran, M.; Guizani, S. Internet-of-things-based smart cities: Recent advances and challenges. *IEEE Commun. Mag.* **2017**, *55*, 16–24. [CrossRef]
4. Ray, P.P. A survey of IoT cloud platforms. *Future Comput. Inform. J.* **2016**, *1*, 35–46. [CrossRef]
5. Kashif, H.; Khan, M.N.; Awais, Q. Selection of Network Protocols for Internet of Things Applications: A Review. In Proceedings of the 2020 IEEE 14th International Conference on Semantic Computing (ICSC), San Diego, CA, USA, 3–5 February 2020; pp. 359–362.
6. Ai, Y.; Peng, M.; Zhang, K. Edge computing technologies for Internet of Things: A primer. *Digit. Commun. Netw.* **2018**, *4*, 77–86. [CrossRef]
7. Zhang, J.; Chen, B.; Zhao, Y.; Cheng, X.; Hu, F. Data security and privacy-preserving in edge computing paradigm: Survey and open issues. *IEEE Access* **2018**, *6*, 18209–18237. [CrossRef]
8. Peng, M.; Zhang, K. Recent advances in fog radio access networks: Performance analysis and radio resource allocation. *IEEE Access* **2016**, *4*, 5003–5009. [CrossRef]
9. Chiang, M.; Zhang, T. Fog and IoT: An overview of research opportunities. *IEEE Internet Things J.* **2016**, *3*, 854–864. [CrossRef]
10. Lin, J.; Yu, W.; Zhang, N.; Yang, X.; Zhang, H.; Zhao, W. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet Things J.* **2017**, *4*, 1125–1142. [CrossRef]
11. Morabito, R.; Petrolo, R.; Loscri, V.; Mitton, N. LEGIoT: A lightweight edge gateway for the Internet of Things. *Future Gener. Comput. Syst.* **2018**, *81*, 1–15. [CrossRef]
12. Villari, M.; Fazio, M.; Dustdar, S.; Rana, O.; Ranjan, R. Osmotic computing: A new paradigm for edge/cloud integration. *IEEE Cloud Comput.* **2016**, *3*, 76–83. [CrossRef]
13. Satyanarayanan, M. The emergence of edge computing. *Computer* **2017**, *50*, 30–39. [CrossRef]
14. Liu, Y.; Zhang, J.; Zhan, J. Privacy protection for fog computing and the internet of things data based on blockchain. *Cluster Comput.* **2020**, 1–15. [CrossRef]
15. Ghanavati, S.; Abawajy, J.H.; Izadi, D. An Energy Aware Task Scheduling Model Using Ant-Mating Optimization in Fog Computing Environment. *IEEE Trans. Serv. Comput.* **2020**, 1–15. [CrossRef]
16. Qadir, J.; Sainz-De-Abajo, B.; Khan, A.; García-Zapirain, B.; De La Torre-Díez, I.; Mahmood, H. Towards Mobile Edge Computing: Taxonomy, Challenges, Applications and Future Realms. *IEEE Access* **2020**, *8*, 189129–189162. [CrossRef]
17. Tian, Z.; Wang, Y.; Sun, Y.; Qiu, J. Location privacy challenges in mobile edge computing: Classification and exploration. *IEEE Netw.* **2020**, *34*, 52–56. [CrossRef]
18. Huang, F.L.; Tseng, S.Y. Predictable smart home system integrated with heterogeneous network and cloud computing. In Proceedings of the 2016 International Conference on Machine Learning and Cybernetics (ICMLC), Jeju, Korea, 10–13 July 2016; Volume 2, pp. 649–653.
19. Wang, X.; Mao, X.; Khodaei, H. A multi-objective home energy management system based on internet of things and optimization algorithms. *J. Build. Eng.* **2021**, *33*, 101603. [CrossRef]
20. Azuatalam, D.; Lee, W.L.; de Nijs, F.; Liebman, A. Reinforcement learning for whole-building HVAC control and demand response. *Energy AI* **2020**, *2*, 100020. [CrossRef]
21. Gao, G.; Li, J.; Wen, Y. DeepComfort: Energy-Efficient Thermal Comfort Control in Buildings via Reinforcement Learning. *IEEE Internet Things J.* **2020**, *7*, 8472–8484. [CrossRef]
22. ALiero, M.S.; Qureshi, K.N.; Pasha, M.F.; Jeon, G. Smart Home Energy Management Systems in Internet of Things networks for green cities demands and services. *Environ. Technol. Innov.* **2021**, doi:10.1016/j.eti.2021.101443. [CrossRef]
23. Khan, M.; Silva, B.N.; Han, K. Internet of things based energy aware smart home control system. *IEEE Access* **2016**, *4*, 7556–7566. [CrossRef]
24. Liu, R.; Ge, Y. Smart home system design based on Internet of Things. In Proceedings of the 2017 12th International Conference on Computer Science and Education (ICCSE), Houston, TX, USA, 22–25 August 2017; pp. 444–448.
25. Wen, X.; Wang, Y. Design of smart home environment monitoring system based on raspberry Pi. In Proceedings of the 2018 Chinese Control And Decision Conference (CCDC), Shenyang, China, 9–11 June 2018; pp. 4259–4263.
26. Foundry, E. EdgeX Foundry. 2021. Available online: https://www.edgexfoundry.org/ (accessed on 10 June 2021).
27. Microsoft. Azure IoT. 2021. Available online: https://azure.microsoft.com/en-us/overview/iot/ (accessed on 10 June 2021).
28. Foundation, A.S. Apache Edgent. 2016. Available online: https://edgent.incubator.apache.org/ (accessed on 10 June 2021).
29. Foundation, O.N. Cord. 2021. Available online: https://opennetworking.org/cord/ (accessed on 10 June 2021).