SPACE: An Empirical Approach towards a User-Centric Smart Campus

Thammineni Prathyusha, Vipul Jindal, Saurabh Gangwar, Anand Konjengbam and Kotaro Kataoka

Indian Institute of Technology Hyderabad, India
Email: {ee14btech11034, es14btech11022, cs15mtech11019, cs14resch11004, kotaro}@iith.ac.in

***Abstract*—Making a campus smart involves a wide variety of devices, users, and other stakeholders. This gives rise to many issues including scalability, extensibility, user-centricity, and cost in terms of both deployment and maintenance. To overcome these issues, we propose a novel platform called SPACE that (1) enables end-to-end commanding and execution of tasks on a smart campus operation in a distributed and modulated manner, and (2) attempts to mitigate the above-mentioned issues. As an initial research testbed, the proposed SPACE has been deployed for different scenarios in a university campus including a classroom, a faculty office, and a lab. This paper empirically reports how the SPACE system enables a smart campus testbed and further lessons from the deployment.**

Keywords–Internet of Things (IoT); IoT Platform; Smart Building; Cloud-based IoT.

I. Introduction

While many research and development efforts were made under the concepts of ubiquitous computing, pervasive computing and Internet of Things (IoT), we have not seen dominating research in the area of IoT platforms that addresses the challenges affiliated with providing seamless user-experience and ease of maintaining the system [1]. Moreover, many of the papers on IoT platform assume that the devices (e.g., bulb, fan, Air Conditioner (AC), etc.) are already smart and have communication capabilities [2][3]. There are virtually unlimited number of "things" that can be connected to the Internet under the umbrella of IoT. Hence, an IoT platform that can connect and accommodate such "things" is needed.

A university campus is an interesting and challenging premises to apply the concept of IoT. A university campus involves many smart contexts [4][5]. Many stakeholders including students, professors, working staff, guests, and administrators are involved there and they have various needs to be satisfied. Professors and students need features such as user-centricity, universal access, and multi-purpose space sharing. Administrators would prefer an economically viable and easy to maintain IoT-enabled campus. A smart campus provides intelligent facilities like universal access, location awareness, user-centricity, and support for heterogeneous devices to the campus community, while the operational cost gets reduced.

This paper presents a platform called SPACE for making a university campus smart by connecting various types of things and facilities (e.g., AC, fan, and bulb) in one system. The proposed platform enables 1) integration of existing campus facilities into SPACE in a cost-effective manner, 2) end users with personalized access to the campus facilities, and 3) an extensible platform where end users can easily assemble new components such as new end devices, controllers, and new communication modules supporting different protocols and standards.

In this paper, we present the implementation and deployment of the proposed platform, and a reference model of end-user oriented and ad-hoc deployment of a smart campus that has not been originally designed and constructed to be smart and user-centric. We introduce the concept of Device Interfacing Gateway (DIGW) and used it to connect any campus facility to the system.

The rest of the paper is organized as follows. Section II provides an overview of related works. Then, we discuss the problems of modelling the system design in Section III, and present the implementation details in Section IV. We evaluate and validate the system in Section V and finally, we conclude the paper and discuss the future works in Section VI.

II. Related Work

Zhang et al. [6] addressed the problem of lack of a versatile and cost-effective software platform for smart buildings. They developed an open source software called Building Energy Management Open Source Software (BEMOSS) that works on a single board computer for monitoring and controlling energy consumption of a building. The system included features like plug and plays using device discovery, and interoperability of different communication protocols. However, the supported devices were limited to already smart ones.

Sánchez et al. [7] implemented IoT applications and services in the city of Santander, Spain, with a physical deployment of more than 2000 sensors. They presented a high-level architectural model supporting real-world IoT experimentation facilities on different devices. In order to address the problem of scale and heterogeneity of devices and application domains, they divided their architecture into three tiers; IoT device, gateway, and server. Georgakopoulos et al. [8] presented an IoT architecture with the concept of service discovery and on-demand integration of devices, storage and computing resources over the cloud. They incorporated data analytics and visualization to create an on-demand IoT application. Although progress has been made on implementation of large-scale IoT, the implementations are generic and personalization of the system is not considered.

The following works addressed the lack of standards for IoT and discussed frameworks for integrating various smart devices. Puatru et al. [9] presented a solution for connecting different types of home appliances to one platform. They focused on how different platforms like Google Nest and Philips Hue can be operated via a mobile device with a Web browser for easy accessibility and better user experience. Nati et al. [10] worked on user-centric IoT for integrating

 ISBN: 978-1-61208-676-7

embedded heterogeneous smart devices in a real-life office environment. The solution offered modular implementation of an open source smart home system using Intel Edison board [11] as a gateway. Mozzami et al. [12] proposed a smart phone based platform to handle heterogeneous devices and multi-vendor smart home appliances in home environment without the need for painstaking configuration and custom programming. They developed an Android application with an open specification for XML driver support. Hernandez et al. [13] proposed a framework for the development of IoT applications where smart objects exhibit autonomy in regards to platforms and human users through management functions. These works assumed that such smart devices can be IP-proxied and capable of sensing and actuating, and device-specific applications are already available.

Hentschel et al. [14] proposed a campus-wide sensor network using Raspberry Pi. They defined supersensors as sensors (light, temperature, motion, sound, and Wi-Fi) attached to Raspberry Pi that are capable of local computer operations and data transmission to a centralized database. The use of Raspberry Pi reduced the cost of procurement compared to standard expensive IoT sensors and has a small footprint. They considered that different sensors have different communication modes and tried to cope with a heterogenous environment. They also discussed the advantages of their method which include intelligent filtering of sensor data as well as capture and buffering of incoming data while the network connectivity is disrupted. Joshi et al. [15] designed a low-cost basic home automation system to control multiple appliances that can be globally monitored and accessed using low cost Raspberry Pi, Arduino and web server.

Some researchers have carried out works related to personalized and user-centric IoT systems. Jayatilaka et al. [16] proposed an assisted living system where appliances exhibit seamless social interactions with people. They embedded multiple sensors such as thermometer, hydrometer, and scale into appliances (refrigerator, microwave, and trash bin). The appliances used Twitter as a messaging system to send notification messages to authorized people. Wu et al. [17] developed a framework for human-system interaction by analyzing the interactive relationship among services, spaces, and users in a smart home environment. Lee and Lin [18] implemented a situation-aware IoT based system that can detect user activities in a room and control the devices in the room accordingly. Alam et al. [19] worked on predicting user behavior based on human activity pattern. Their approach used episodes of events of home appliances that have on-off states (such as lights, fans, heater, and window blinds) as an input to a sequence prediction algorithm to predict the next activity from previous history. Using multi-sensor data streams, Chen et al. [20] worked on a knowledge-driven real-time continuous activity recognition using multi-sensor data streams in a smart home environment. The approach uses domain knowledge, ontologies, semantic reasoning and classification for activity recognition. Our work is related to developing a platform for smart and user-centric campus environment that provides facilities such as universal access, user-centricity, and support for heterogeneous devices.

III. System Design

A. System Requirement Organization

In this section, we describe the system requirements of our proposed SPACE platform and explain the approaches for implementing a smart campus.

Universal access: End devices and campus facilities should be accessible from anywhere so that users can control devices without physically being present near the devices. SPACE provides universal access by enabling end users to operate all the end devices using a single mobile application.

User-centricity: The system should be simple in terms of deployment, usage, and maintenance for the user. As a consequence of user-centricity, SPACE provides pleasing user experience and user-friendly interface.

Cost effectiveness: The cost of a system includes its deployment, maintenance and the cost of upgrading/adding new components. The cost of deploying SPACE over the existing infrastructure is considerably lower than replacing all the existing components (e.g., lights) in the infrastructure with smart components (e.g., smart bulbs). Physical contact with switches is generally required to control the components. Repeated physical contact results in wear and tear of switches and may lead to accidents. Such accidents can be avoided by replacing physical switches with virtual switches that are controlled through a mobile/desktop application.

Support for heterogeneous IoT devices: Various heterogeneous devices are present in an environment. Such devices have different operations and need different protocols to control them. It is desirable to identify, integrate, and control such devices via a single application so that the user need not use a different application for each end device.

Extensibility: The system needs to be extensible regarding support of both non-smart and already smart devices. It should accommodate various kinds of communication and processing technologies as per the need of the environment in the deployed system. Users should be able to integrate devices with minimum support from the system administrator.

B. System Architecture/Design Overview

The overall architecture of SPACE consists of four main components - local controller, central controller, Device Interfacing Gateway (DIGW), and mobile application. Figure 1 shows the system overview of SPACE.

As shown in the figure, the platform is divided into three layers: User Interface (UI) layer, logical control layer, and physical control layer. A local controller is deployed in each environment such as rooms, offices, and labs. It performs edge computation and it is responsible for managing its local environment. All end devices in an environment communicate with a local controller via DIGW. A DIGW is attached to each of the end devices and is among one of the most important components of the platform. The mobile application can interact with the end devices via central controller or local controller. Each of the components has some specific functions and they interact with one another to fulfill the system requirements. Figure 2 shows the information exchange among the components of SPACE.

 ISBN: 978-1-61208-676-7

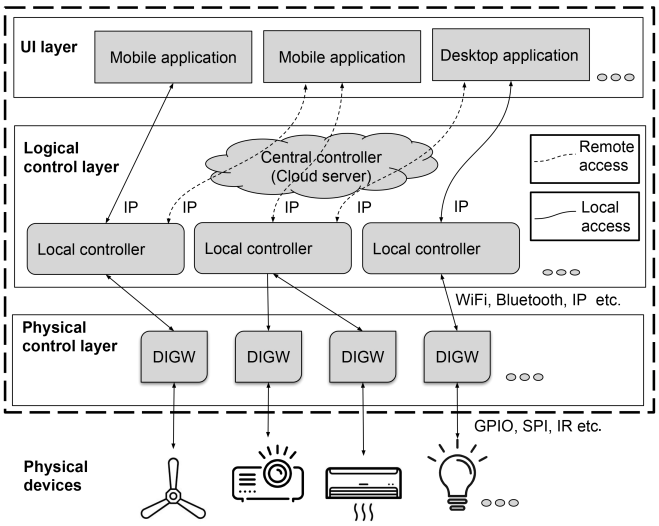

Figure 1. System overview of SPACE

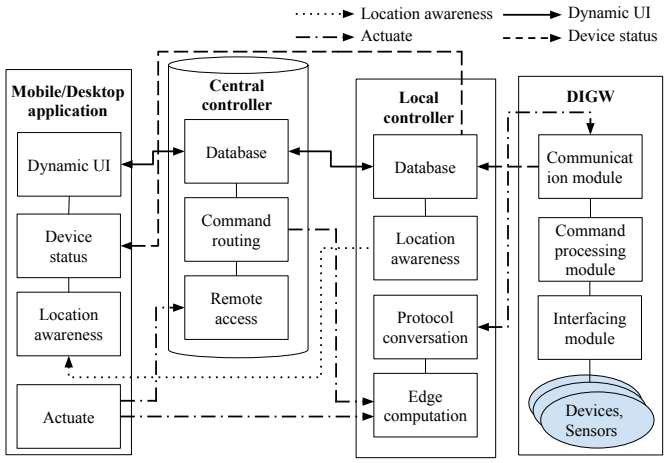

Figure 2. System block information exchange of SPACE

The **Device Interfacing Gateway (DIGW)** is used to proxy and control end devices. Generally, end devices are not capable of communication. To connect these end devices to a network, a DIGW is plugged into each end device. DIGW receives commands from local controller and sends data/feedback from sensors and end devices to local controller, as illustrated in Figure 2. It also enables discovery of end devices. DIGW is comprised of four modules: communication module, command processing module, interfacing module, and power module. The details of each module are explained below.

- The **communication module** takes care of receiving and sending signals over any particular communication protocol like ZigBee, Bluetooth, and Wi-Fi. It does necessary processing of signals to convert them into a standard format understandable by the command processing module.
- The **command processing module** receives commands from communication module, interprets the commands and accordingly calls the right functions that are stored in its memory. The functions then trigger the interfacing module to actuate the command. It also takes care of internal optimization regarding power consumption.
- The **interfacing module** is the physical interface between an end device and DIGW. It consists of actuators such as switches, valves, and infrared (IR) modules that can control the end device. Actuators provide functions such as turning ON/OFF of end devices, modulating control changes, and switching action between different states.
- The **power module** takes care of the power consumed by all the DIGW modules. It is interchangeable with different parts such as adapter, battery, and solar cell.

The **central controller** is responsible for data storage, data processing, command routing, and remote access. The central controller is a cloud server through which all the local controllers and mobile applications can interact. All the information about end devices, room environments, and user information (such as user authentication, and location) are stored in the database of the central controller. The mobile application uses the information present in the database of the central controller for functions such as device status and dynamic UI. The use of cloud service is to provide scalable computing and storage power for developing, maintaining, and running multiple services simultaneously.

The **local controller** is a control unit present in all the environments such as rooms and labs. It is responsible for all the activities happening in its local environment apart from the authentication process, which is taken care of by the central controller. It stores the relevant information of end devices present in its local environment. It supports different communication protocols and acts as a bridge between the DIGW and the central controller. It is responsible for the real-time processing of sensor data in its environment. It is also responsible for protocol conversion. For example, consider a scenario where some end devices speak Zigbee via DIGW, and some other devices speak Wi-Fi and remaining devices speak Bluetooth and Radio-Frequency Identification (RFID). The local controller converts the command request from the mobile application/central controller to the supported protocol and sends the formatted command to the respective DIGW and vice-versa. The local controller broadcasts its location information along with a list of end devices that can be controlled within its environment. This information helps mobile applications to be aware of the location and improves user experience (explained in the next section). If the user is near a local controller, commands from the mobile application reach the local controller earlier before reaching the central controller, and so the command gets executed faster. It performs edge computing which helps in optimizing various latencies such as command execution time.

The **mobile application** gets access to end devices via the central controller or the local controller depending on the location (remote/local) of the user. It has a dynamic UI that changes depending on the location of the user. This feature helps in minimizing the number of manual operations required to perform an action, hence providing a good user experience. The UI of the application reflects consistent information regardless of the technology used at lower layers. For example, two different temperature sensors have the same

 ISBN: 978-1-61208-676-7

type of UI even though they are from different manufacturers and use different protocols such as ZigBee or Bluetooth. The mobile application also supports voice commands to control devices.

IV. System Implementation

This section details the implementation of SPACE inside a university (Indian Institute of Technology Hyderabad, India) as a testbed. We assume that, initially, the end devices are not capable of any communication.

A. Technical Components

TABLE I. Software and hardware components of SPACE

Component	Software	Hardware
Mobile application	Operating System (OS): Android K,L,M; Google Speech Application Program Interface (API)	Any smartphone with Internet connectivity
Central Controller	OS: Ubuntu 16.04; Database: PostgreSQL	2.40 GHz quad core Intel Xenon CPU; 8GB RAM
Local Controller	OS: Raspbian	Raspberry Pi (v2,v3), Bluetooth Low Energy (BLE) module
DIGW for air conditioner	Real-Time Operating System (RTOS) and ATtention (AT) commands API; Arduino Interactive Development Environment (IDE)	Espressif Systems (ESP8266EX) [21], IR module
DIGW for room lights	RTOS and AT commands API; Arduino IDE	ATMEGA328P [22]; SPDT relay

Table I presents a concise summary of the various software and hardware elements used by components for implementing the SPACE platform. It may be noted that implementation is done mostly using open-source software.

Figure 3 shows a cropped interface of the mobile application. The sidebar displays information about the user,

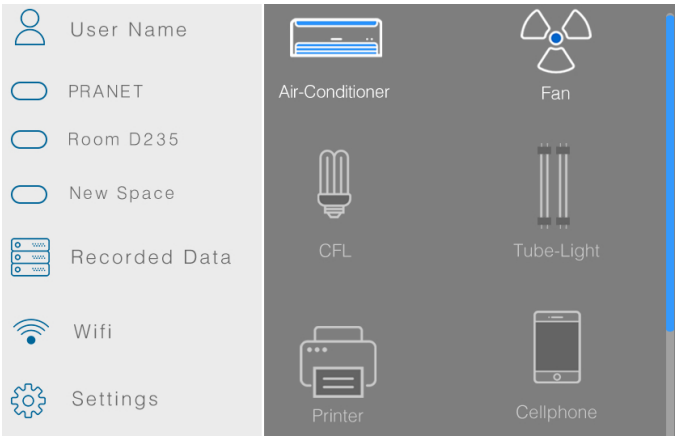

Figure 3. UI of mobile application - side menu and main menu buttons

rooms, recorded data, and available settings. The main menu has clickable icons to control the devices present in a room. The icon changes from grey to blue when the corresponding device changes its state from OFF to ON and vice versa.

The mobile application also supports voice commands through Google Speech API. Certain key phrases are used to train the API to perform actions similar to user clicks. For example, we used the phrase "Lights on" to initiate function calls for turning on lights. The voice command control was deployed in an office. However, we removed this feature in the latest version of the mobile application as the Google Speech API was not accurately detecting the voices and required extensive samples for training data.

Figure 4 shows the block diagram of a local controller along with DIGW and ampere sensors. Here, *Raspberry Pi v3* is used as the local controller for performing edge computation and storing data. Arduino, as well as Single Pole Double Throw (SPDT) relay are used as DIGW to interact with ampere sensors and end devices. Ampere sensors detect the state of devices using the current readings.

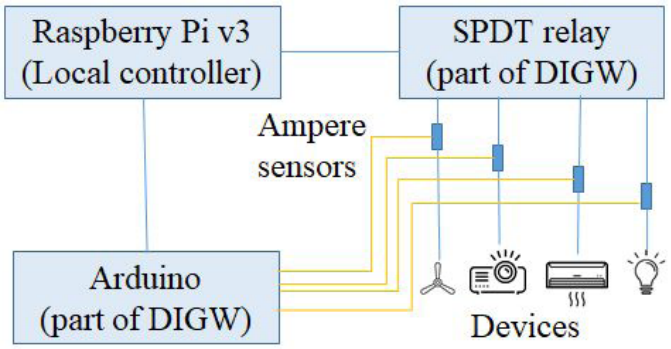

Figure 4. Block diagram of a local controller along with DIGW and ampere sensors

Figure 5 shows the different modules of DIGW used in the real-time deployment of SPACE. All these modules

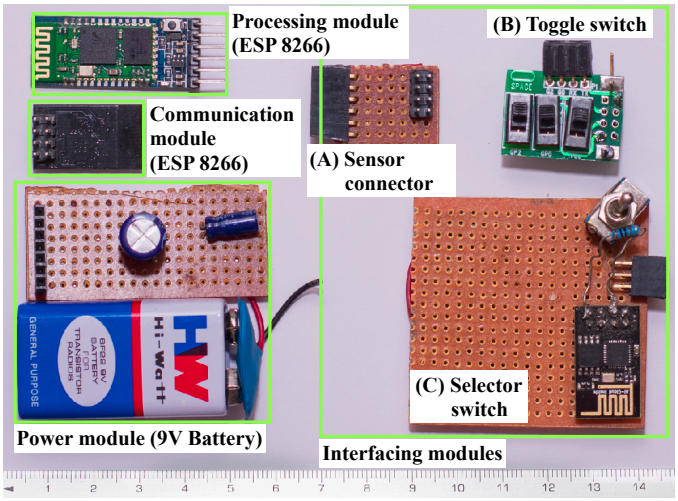

Figure 5. DIGW used in real-time deployment of SPACE

work independently, hence, making DIGW modular. The four modules communicate with each other over General-Purpose

Input/Output (GPIO) interface [23], which is a standardized mode of communication. Consider a case where, after the deployment, the user wishes to change the mode of communication from Bluetooth to Wi-Fi. The user can do so by getting a Wi-Fi module, which is compatible with DIGW and replacing the Bluetooth module by the Wi-Fi module in a plug and play fashion. This feature helps in saving the upgrading cost of the system, as we need not change the whole DIGW to change some particular features of the system.

B. Communication Protocols

Figure 6 shows the information flow with regards to time between various components of the system for executing a command.

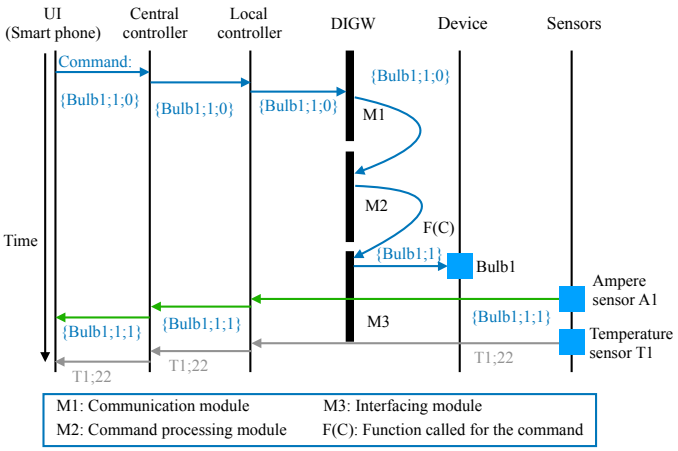

Figure 6. Information flow between components of SPACE

For illustration, we examine two types of information flow: 1) the message flow of user triggered command followed by the sensor data flow that senses the state of the end device after execution of the command, and 2) periodic update of sensor data. Consider a remote user is sending a command to turn on a light bulb using the mobile application. The application generates a command packet consisting of device ID, desired state, and reply status ({Bulb1;1;0}) and passes it to the central controller. The central controller parses the command and routes it to the appropriate local controller. The local controller identifies the light bulb from the device ID and routes the command to the appropriate DIGW, which is connected to the desired light bulb. The communication module (M1) of the DIGW receives the signal from the local controller and converts the signal into a format, which is understandable by the processing module (M2) and then passes the message to M2. M2 then calls the appropriate function F(C) to turn on Bulb1. F(C) gets executed via the interface module (M3), and the light bulb gets switched on. The ampere sensor connected to Bulb1 detects a change in the state after the command execution and sends the sensor data to the mobile phone via the local controller and the central controller, respectively. The application then automatically reflects the new state of the light bulb in the UI. Also, the temperature reading of the room is periodically sent from the temperature sensor (T1) to the mobile application via the local controller and the central controller.

C. Data Structure

Our system implementation used six data types, as shown in Table II.

TABLE II. DATA STRUCTURE OF MESSAGE PACKETS

Data Type	Fields
Authentication	Username, Password, Reply flag
Location	LocationID (IPv6 address), Location SSID
Device	DeviceID, Device name, Device type, LocationID, StateID
Sensor data	SensorID, FloatData
Command	DeviceID, DesireState, Reply
Preset	Room Type, DeviceID, DesireState

Each data type is described as follows.

- **Authentication:** *Username* and *Password* are used for storing the credentials for authorized users. *Reply flag* indicates whether a command is successful or not.
- **Location** is identified by the static IPv6 address assigned to the local controller. In addition, it consists of a human-readable name in the form of an Service Set Identifier (SSID) that in projected in each room.
- **Device** stores the details of a device. The mobile application uses this message packet to know the type and state of devices present in a room.
- **Sensor data** stores the reading of the sensors. This reading is periodically sent to the local controller.
- **Command** is a message triggered by the end user via the mobile application. It contains information regarding the requested action by the end user along with the id of the end device on which the action must be performed.
- **Preset** is used to change the setting of a group of devices for performing a combined task. For example, *lecture preset* is used to turn on AC, projector and turn off lights to prepare for a presentation.

V. EVALUATION

In this section, we discuss the output of implementing the proposed platform. For evaluation, we consider three different room environments: a personal room, a professor's office room, and a computer lab. 10 student volunteers were selected to evaluate various scopes manually.

Universal Access: To test the universal access of the system, the volunteers were asked to control various devices in the rooms via the mobile application. Three different scenarios were considered: 1) when they are inside the room and connected to the university Wi-Fi, 2) when they are at a different location of the institute and connected to the university Wi-Fi, and 3) when they are out of the institute and connected to the Internet using 3G/4G LTE. In all the three scenarios, the volunteers could successfully control all the functions of the system, monitor room temperature, motion inside the rooms, and state of the devices (light, fan, and air conditioner) in real time. This shows that the devices are accessible to the users irrespective of where they are present.

Personalization: The volunteers were asked to install the application and perform the sign-up process on their own.

 ISBN: 978-1-61208-676-7

Using the application, they were asked to add a room and all controllable devices present in the room. The volunteers could arrange the layout of the icons according to their convenience. In average, they took 8 minutes to complete the whole setup under regular Wi-Fi connectivity. This shows that the users could complete the set up and personalize the UI with ease.

Location Awareness: Global Positioning System (GPS) was not used to detect location because of its high power consuming property. A unique SSID, which was projected by all the local controllers, was used to detect the present location of the user. The transmission power of Wi-Fi was controlled to be minimal so that its signal did not penetrate the walls of the room and was available only inside the room. Once a user entered the room, the mobile application picked up the SSID from the local controller and the application dynamically changed its UI, depending on the location (e.g., room, lab, etc.) of the user. This ensured that the user was shown the UI of the room along with devices that could be controlled, rather than seeing the UI of some other room. Consistent observations were made in all the three different room environments. This feature of dynamically changing the UI of the application reduced the number of manual operations a user needs to turn on a device from three to just one.

Cost Effectiveness: Most of the existing IoT solutions use expensive components that are designed for some specific purpose. On the contrary, our implementation uses general readily available inexpensive components such as Raspberry Pi, Arduino, etc. This brings the current cost of the deployed platform to $51 only per room. Each deployed system can accommodate up to 20 end devices. Table III shows the breakdown of the cost of deployment per room.

TABLE III. Cost for deploying the platform in a room

Gadget	Price (in USD)	# devices supported
Raspberry Pi v3	35	20
Arduino	6	17
ESP8266	5	1
SPDT	5	8
Misc. devices	5	1
Total	**51**	

Scalability of storage space and processing power: The use of central controller and cloud infrastructure makes the platform scalable in terms of storage space and processing power. Using cloud infrastructure, the storage space and processing power of the central controller can be increased as per the scale of deployment. The storage space and processing power for a local controller depend on the specification of the Raspberry Pi used.

Reaction Time: To test the reaction time and concurrent execution of the commands, we present three instances of command request and the time delays in different stages of execution. The details of the performance of command request through a local controller are shown in Table IV. In the table, the *Number of requests* represent the parallel requests made to the same local controller simultaneously, from different end devices. *Pi execution time* represents the time taken by the local controller to process and forward the command to the DIGW after receiving it from the mobile application. *Communication delay* represents the time taken from the command to go from the mobile application to the local controller. *Net latency* represents the total latency, i.e., the sum of Pi execution time and communication delay. The same user may make multiple requests (e.g., turn on light and AC), but each user is asked to make a different request at the same time. More parallel requests mean more processing load on the local controller and more probability of request drops, hence increasing the reaction time. We observe that the net latency increases with the number of parallel requests. The increase in latency is because of 1) drop of packets by the local controller due to overloading, and 2) Wi-Fi signal delay between the mobile and the local controller. The total delay is consistently low and the system works well even in the situation where 30 different command requests were made concurrently.

TABLE IV. Service processing time comparison of three instances of command execution

Instance	Number of requests	Pi execution time (ms)	Communication delay delay (ms)	Net latency (ms)
1	5	0.002	0.165	0.167
2	10	0.002	0.213	0.215
3	30	0.003	0.250	0.253

VI. Conclusion and Future Works

This paper presented a novel framework for making a smart campus environment called SPACE. Our platform offers user-centric functionality based on user location and preference for controlling devices in the surrounding space through a mobile application. We validated the performance of the SPACE platform through implementation and practical use-cases in a university campus. The modularization property of the DIGW allows SPACE to integrate various modes of communication, power supply, and devices. The proposed platform has high potential to support a wide variety of services and applications on it.

The future work involves polishing the hardware by 3D-printing custom-designed circuit boards and casing with standard design guidelines. This may also help in downsizing and reducing the cost of the local controller and the DIGW. Another direction of future work is optimizing the behavior of the smart campus based on learning of user activity data. Artificial Intelligence and data mining algorithms may be used to predict user activities and actuate the campus facility. Privacy preservation of the user-generated data must be considered for using such data.

References

[1] J. Bergman, T. Olsson, I. Johansson, and K. Rassmus-Gröhn, "An exploratory study on how internet of things developing companies handle user experience requirements," in International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, 2018, pp. 20–36.

[2] M. Swan, "Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0," Journal of Sensor and Actuator Networks, vol. 1, no. 3, 2012, pp. 217–253.

[3] T. Sztyler, "Towards real world activity recognition from wearable devices," in Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on. IEEE, 2017, pp. 97–98.

 ISBN: 978-1-61208-676-7

[4] A. Alghamdi and S. Shetty, "Survey toward a smart campus using the internet of things," in Future Internet of Things and Cloud (FiCloud), 2016 IEEE 4th International Conference on. IEEE, 2016, pp. 235–239.

[5] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," IEEE Communications Surveys & Tutorials, vol. 16, no. 1, 2014, pp. 414–454.

[6] X. Zhang, R. Adhikari, M. Pipattanasomporn, M. Kuzlu, and S. R. Bradley, "Deploying iot devices to make buildings smart: Performance evaluation and deployment experience," in Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on. IEEE, 2016, pp. 530–535.

[7] L. Sánchez, V. Gutiérrez, J. A. Galache, P. Sotres, J. R. Santana, J. Casanueva, and L. Muñoz, "Smartsantander: Experimentation and service provision in the smart city," in Wireless Personal Multimedia Communications (WPMC), 2013 16th International Symposium on. IEEE, 2013, pp. 1–6.

[8] D. Georgakopoulos, P. P. Jayaraman, M. Zhang, and R. Ranjan, "Discovery-driven service oriented iot architecture," in Collaboration and Internet Computing (CIC), 2015 IEEE Conference on. IEEE, 2015, pp. 142–149.

[9] I.-I. Pătru, M. Carabaş, M. Bărbulescu, and L. Gheorghe, "Smart home iot system," in RoEduNet Conference: Networking in Education and Research, 2016 15th. IEEE, 2016, pp. 1–6.

[10] M. Nati, A. Gluhak, H. Abangar, and W. Headley, "Smartcampus: A user-centric testbed for internet of things experimentation," in Wireless Personal Multimedia Communications (WPMC), 2013 16th International Symposium on. IEEE, 2013, pp. 1–6.

[11] Hardware Guide, Intel Edison Kit for Arduino. San Val, 2015.

[12] M.-M. Moazzami, G. Xing, D. Mashima, W.-P. Chen, and U. Herberg, "Spot: A smartphone-based platform to tackle heterogeneity in smart-home iot systems," in Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on. IEEE, 2016, pp. 514–519.

[13] M. E. P. Hernández and S. Reiff-Marganiec, "Towards a software framework for the autonomous internet of things," in Future Internet of Things and Cloud (FiCloud), 2016 IEEE 4th International Conference on. IEEE, 2016, pp. 220–227.

[14] K. Hentschel, D. Jacob, J. Singer, and M. Chalmers, "Supersensors: Raspberry pi devices for smart campus infrastructure," in Future Internet of Things and Cloud (FiCloud), 2016 IEEE 4th International Conference on. IEEE, 2016, pp. 58–62.

[15] J. Joshi, V. Rajapriya, S. Rahul, P. Kumar, S. Polepally, R. Samineni, and D. K. Tej, "Performance enhancement and iot based monitoring for smart home," in Information Networking (ICOIN), 2017 International Conference on. IEEE, 2017, pp. 468–473.

[16] A. Jayatilaka, Y. Su, and D. C. Ranasinghe, "Hotaal: Home of social things meet ambient assisted living," in IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops). IEEE, 2016, pp. 1–3.

[17] C.-L. Wu and L.-C. Fu, "Design and realization of a framework for human–system interaction in smart homes," IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, vol. 42, no. 1, 2012, pp. 15–31.

[18] S.-Y. Lee and F. J. Lin, "Situation awareness in a smart home environment," in Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on. IEEE, 2016, pp. 678–683.

[19] M. R. Alam, M. B. I. Reaz, and M. M. Ali, "Speed: An inhabitant activity prediction algorithm for smart homes," IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, vol. 42, no. 4, 2012, pp. 985–990.

[20] L. Chen, C. D. Nugent, and H. Wang, "A knowledge-driven approach to activity recognition in smart homes," IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 6, 2012, pp. 961–974.

[21] E. S. C. Platform, "Esp8266," Espressif Systems, 2013.

[22] W. Kunikowski, E. Czerwiński, P. Olejnik, and J. Awrejcewicz, "An overview of atmega avr microcontrollers used in scientific research and industrial applications," Pomiary Automatyka Robotyka, vol. 19, 2015.

[23] S. Balachandran, "General purpose input/output (gpio)," Michigan State University College of Engineering. Published, 2009, pp. 08–11.

 ISBN: 978-1-61208-676-7