SITE: The Simple Internet of Things Enabler for Smart Homes

Basim Hafidh

Thesis submitted in

partial fulfillment of the requirements for the

PhD in Electrical and Computer Engineering degree

Ottawa-Carleton Institute for Electrical and Computer Engineering

School of Electrical Engineering and Computer Science

Faculty of Engineering

University of Ottawa

Abstract

This thesis presents the Simple Internet of Things Enabler (SITE), a smart home solution that allows users to specify and centrally control IoT smart objects. Unlike most existing systems, SITE supports End-User Development (EUD). It includes features that make the system accessible to users that do not possess a background in Information Technology (IT). Hence, it defines a simple language for the specification of control rules for smart objects. It also provides a user interface to graphically illustrate data received from smart objects.

Furthermore, we present the SITE architecture and describe the components that enable users to define, register, and operate smart objects within a smart home environment. Since deploying applications on the cloud renders many advantages pertaining to data security, robustness, and elasticity of resources, we additionally propose a cloud-based architecture for SITE. In this case, SITE acts as a service hosted on a cloud platform that realizes monitoring and control of a smart home remotely.

Moreover, since most of the objects in any environment are not inherently smart, we propose a framework that affords “everyday” objects the necessary modules to measure and report their state. Hence, users realize the smart objects using a transducer network framework that supports the amalgamation of multiple transducers into a single smart object. To make these objects easily reconfigurable, we apply a plug and play mechanism to enable the clustering of any number of transducers. We propose an algorithm that dynamically detects added and removed transducers from a smart object.

To assess the usability of SITE, we conduct an empirical study involving 20 participants belonging to two user groups: users with technical training (IT users) and users without technical training (Non-IT users). We demonstrate that both user groups can satisfactorily build smart objects and define control rules in a smart home environment using SITE.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Abdulmotaleb El Saddik for his constructive advice, academic support, encouragement, valuable guidance, and financial support to accomplish this research.

Special thanks are due to my co- supervisor Professor Hussein Al Osman who made his strong technical background ready for my utilization. His invaluable help, continuous support and careful revisions for the last five years reflected in a successful research achievement.

Also, I would like to thank the members of the Multimedia Communication Research Laboratory (MCRLab), for their suggestions and cooperation during my research work.

I owe my deepest gratitude towards my better half for her eternal support and understanding of my goals and aspirations. Her infallible love and support have always been my strength. Her patience and sacrifice will remain my inspiration throughout my life. Without her help, I would not have been able to complete much of what I have done and become who I am. It would be ungrateful on my part if I thank Bushra in these few words. I am thankful also to my sons Ali and Mustafa, whose consistent encouragements, support, and endless love have taken me through all hardships incurred during my studies and research journey.

As always it is impossible to mention everybody who had an impact on this work however there are those whose moral support is even more important. I feel a deep sense of gratitude for my father and mother who formed part of my vision and taught me good things that really matter in life. Their infallible love and support have always been my strength. Their patience and sacrifice will remain my inspiration throughout my life. I am also very much grateful to all my family members for their constant inspiration and encouragement.

This thesis is dedicated to

the memory of my beloved father, ***Abed Hafidh.***

It is your shining example that I try to emulate in

all that I do.

Thank you for everything.

Table of Contents

List of Figures

List of Tables

List of Acronyms

μ	Membership functions
ActuatorML	Actuator Model Language
CVC	Central Visualization and Contro1
DID	Dynamic Information Database
EUD	End-User Development
FD	Fuzzy Database
GPTN	General-Purpose Transducers Network
GUI	Graphical User Interface
I^2C	Inter-Integrated Circuit
IaaS	Infrastructure as a Service
IP	Internet Protocol
IT	Information Technology
IoT	Internet of Things
MAC	Media Access Control
PaaS	Platform as a Service
RFID	Radio Frequency IDentifier
SaaS	Software as a Service
SCL	Simple Control Language
SensorML	Sensor Model Language
SID	Static Information Database
SITE	Simple Internet of Things Enabler
SO	Smart Object
UI	User Interface
UIDL	User Interface Description Language
UML	Unified Modeling Language
VM	Virtual Machine

Chapter 1 **Introduction**

1.1 Background

The Internet of Things (IoT) is a worldwide network of smart objects or "things" (e.g. Radio Frequency Identifier (RFID) Tags, Home appliances, surveillance cameras, sensors, actuators, mobile phones) uniquely addressable, based on standard communication protocols [1-4]. Through the Internet, these objects can communicate their sensory data while being remotely monitored and controlled by users or autonomous applications [1]. The IoT concept was initially popularized by the MIT Auto-ID labs where researchers proposed the use of a wireless sensor network and RFID technology for object localization [5]. The International Telecommunication Union (ITU) in their 2005 report formally defined the IoT as the set of all objects that can communicate with each other via networks [6]. The IoT paradigm can be applied to a multitude of domains (see Figure 1.1) such as mobile healthcare and industrial automation, smart energy monitoring and control, elderly assistance, public security, urban management, infrastructure construction, business services, and smart homes [3, 4, 7, 8]. In this thesis, we are interested in the latter application of IoT.

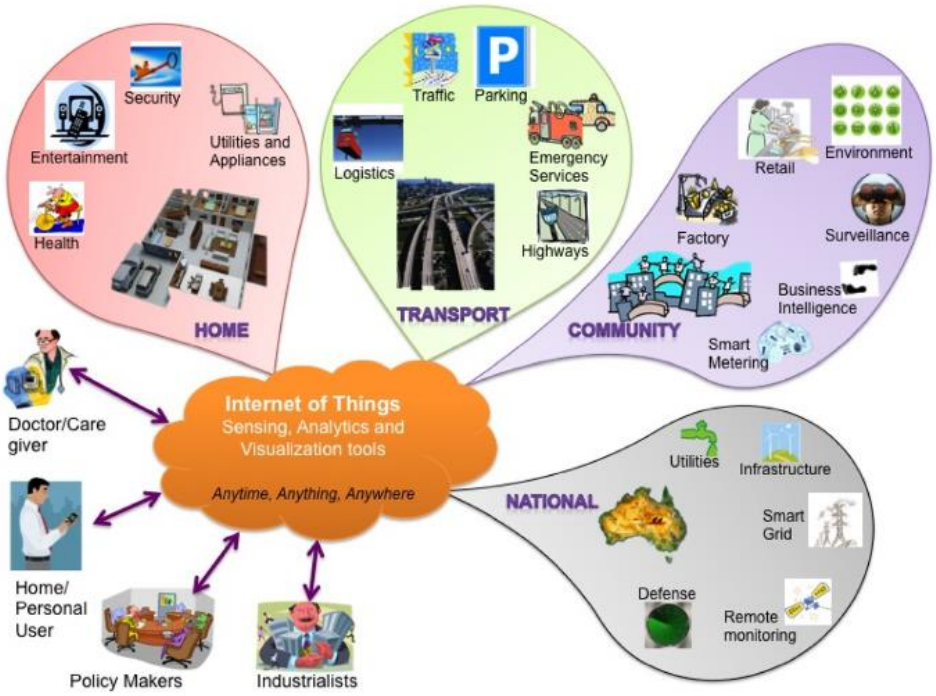


Figure 1.1 Various application domains of IoT [8]

The term smart home has evolved from exclusively referring to the centralized and semi-automated control of environmental systems, such as heat and lights, to the use of technology to monitor and control any compatible object in the home environment. Typically, the goal of smart home systems is to provide comfort, health care, security, safety, and energy consumption reduction services [8-11].

Smart objects (SOs) are the foundational building blocks of IoT [12, 13]. However, most "regular" objects in our everyday environments are not smart (i.e. equipped with remote monitoring, control, and communication capacities). To extend the IoT beyond inherently smart objects, we have to append transducers (sensors or actuators), and network connectivity components to "regular" objects. Such undertaking would add a level of intelligence to these objects by allowing them to communicate their status (e.g. temperature or pressure applied). Also, they can be potentially controlled remotely through actuators

(e.g. switch to turn an electrical device on or off). Therefore, a smart home can be seen as a combination of SOs that exchange their information among each other and also transport this information to a central server that makes it possible to interact or monitor them [14]. For this purpose, we propose the General-Purpose Transducers Network (GPTN) that allows the creation of network-enabled clusters of transducers that can be attached to various objects to form SOs. These clusters are formed through a plug and play mechanism.

Today, highly technical divisions, using programming languages that are not accessible to most end-users, program the logic pertaining to the control of SOs in smart homes. The relatively recent perforation of technological devices into our daily lives has compelled users, including those who are not technically trained, to seek an active role in technical development. This allows them to design, configure, modify, or realize technologies that are better tuned to their individual needs. This phenomenon has been dubbed in the literature as End-User Development (EUD) [15].

EUD refers to a set of activities, techniques, and tools that allow end-users to configure, modify and control software and hardware artifacts [16-17]. These artifacts should be fully "plug and play". We propose a system that enables EUD for smart homes. Individuals, after all, personalize all aspects of their home, and therefore, it is valuable to equip them with the necessary tools to configure, modify, and control their smart home systems. Furthermore, these tools would allow them to adapt these systems to their needs or preferences as they change over time.

In this thesis, we introduce the Simple Internet of Things Enabler (SITE), a system composed of hardware and software components that allows end-users to design and configure a smart home system that responds to their needs. The system is designed to support two broad classes of users: IT and Non-IT users. We define IT users as those that

possess an undergraduate degree in a discipline that includes intermediate or advanced courses in software development, hardware development or both, or have undergone equivalent training. Non-IT users refer to users that have not undergone any training in software and/or hardware development. To support Non-IT users, we propose the Simple Control Language (SCL) for the central control of SOs in a smart home. SITE controls SOs according to predefined Fuzzy logic rules. Hence, two algorithms are proposed to convert the user SCL rules into Fuzzy logic ones. The first algorithm generates membership functions and the second generates Fuzzy rules. We demonstrate through a usability study that both user classes can:

- Build SOs out of everyday objects using the GPTN framework; and
- Specify SCL rules that permit a central server to interact with these SOs.

1.2 Research Contributions

SOs are fundamental building blocks for smart homes. However, most everyday objects are not instrumented with sensors, actuators, and communication capacities, which are prerequisite components to realize SOs. Moreover, smart home system development necessitates a large amount of effort by skilled designers. Hence, we want to build a framework that allows users, including those that do not have an expertise in information technology, to build and control smart objects in a smart home setting.

In this thesis, we introduce the Simple Internet of Things Enabler (SITE), a complete system, composed of hardware and software components that allows users to:

- Build smart objects,
- Specify rules to control smart objects, and

- Store and present the data produced by smart objects.

The system is designed to support various types of users, including those that do not possess a background in Information Technology (referred to in this thesis as Non-IT users).

Consequently, the following are the contributions of this thesis:

- Design and implementation of a general-purpose transducer network that permits users to transform everyday objects into smart ones that are controlled by a central server
- Design and implementation of a rule-based language we call SCL (Simple Control Language) to allow end-users to define the interaction behavior between the central server and SOs based on Fuzzy logic.
- Design and development of algorithms to convert SCL rule into Fuzzy rules.
- Development of a communication protocol between the central server controlling the smart home (referred to in this thesis as the Central Visualization and Control) and computing devices that interact with the smart home's SOs.

1.3 Scholarly Achievements

The following papers were published during my Ph.D. studies:

Refereed Journal Publications:

[1] Basim Hafidh, Hussein Al Osman, Juan Arteaga-Falconi, Haiwei Dong, and Abdulmotaleb El Saddik, "SITE: The Simple Internet of Things Enabler for Smart Homes", *IEEE Access*, 5, (2017): 2034-2049.

[2] Basim Hafidh, Hussein Al Osman, Haiwei Dong, and Abdulmotaleb El Saddik, "A Framework of Reconfigurable Transducer Nodes for Smart Home Environments", *IEEE Embedded Systems letters*, IEEE 7, no. 3 (2015): 81-84.

[3] Mohamad Hoda, Yehya Hoda, Atif Alamri, Basim Hafidh, and Abdulmotaleb El Saddik, "A Novel Study on Natural Robotic Rehabilitation Exergames Using the Unaffected Arm of Stroke Patients," *International Journal of Distributed Sensor Networks*, Article ID 590584, 2015 (2015): 5.

[4] Mohamad Hoda, Yehya Hoda, Basim Hafidh, and Abdulmotaleb El Saddik, "Predicting Muscle Forces Measurements from Kinematics Data Using Kinect in Stroke Rehabilitation," *Multimedia Tools and Applications*, [DOI 10.1007/s11042-016-4274-5].

Refereed Conference Publications:

[1] Mohamad Hoda, Basim Hafidh, and Abdulmotaleb El Saddik, "Haptic Glove for Finger Rehabilitation", *Fifth ICME International Workshop on Multimedia Services and Technologies for E-health*, MUST-EH 2015, accepted for publication.

[2] Basim Hafidh, Hussein Al Osman, and Abdulmotaleb El Saddik. "SmartInsole: A foot-based activity and gait measurement device." *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on*. IEEE, 2013.

1.4 Thesis Organization

This thesis is organized as follows:

- Chapter 1 introduces the topic, presents the motivation, and describes the major contributions of the thesis.

- Chapter 2 presents an abstract view of the smart home architecture and reviews related existing work on SOs and smart home systems.
- Chapter 3 describes the proposed GPTN framework and specifies its various components.
- Chapter 4 details the proposed SITE system design by describing its components and their functionality. It also showcases the user interface of the proposed system.
- Chapter 5 presents an evaluation of the SITE system. The obtained results are analyzed and discussed in this chapter.
- Finally, Chapter 6 summarizes the thesis and presents ideas for future work.

Chapter 2 **Background and Related Works**

In this chapter, we discuss some existing work on smart homes and SOs technologies. In particular, we present an abstract view of smart homes that describes the major composing elements of these systems. We also introduce the concept of SOs and the existing mechanisms to convert everyday objects into "smart" ones.

2.1 Abstract View of Smart Home Architecture

A typical smart home is composed of several SOs that communicate with a central application. This application is referred to in various works as the Distributed Services Oriented Middleware [11], E-Servant [18], Controller [19], Home Gateway [20], Gateway and Integrator [21], ZigBee-based Intelligent Self-Adjusting Sensor (ZiSAS) [22], Home Server [23, 24], etc. However, the principle functions of such application are similar and we can summarize them as follows (see Figure 2.1):

- It receives sensory information from the SOs deployed in the smart home environment.
- It controls the smart environment through commands sent (typically wirelessly) to a subset of the existing SOs deployed in that environment. These commands are usually sent in response to sensory information obtained from the environment, a user command entered onto the system or a pre-configured timer.

Also, this kind of application optionally allows the user to visualize the collected sensory information at various levels of granularity. In this thesis, we will refer to such application as the Central Visualization and Control (CVC) application.

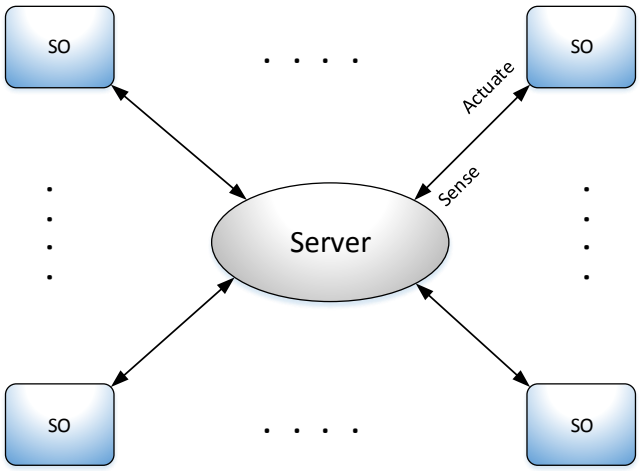


Figure 2.1 The smart home's server high-level architecture

2.2 Existing Smart Home Systems

Several works presented smart home systems for monitoring and controlling SOs [11, 18, 19, 21, 25]. These systems are composed of hardware and software components and typically allow the user to seamlessly, locally or remotely control the house brightness, ventilation, temperature, humidity, doors and windows, and so forth. For example, a European project called "Easy Line" implemented and evaluated a smart kitchen that increases the autonomy of elderly and disabled people in their kitchen-related activities [18]. Household appliances, sensors, interfaces, and communication standards were integrated into the system's modular architecture. The software architecture of the system was based on the Open Services Gateway initiative (OSGi) which entails the construction of complex systems from highly specialized small modules. Similarly, Folea *et al.* [25] presented a system that transforms a normal house into a smart one with the aid of a wireless sensor network. The system measures different physical properties such as temperature, humidity, light, smoke, and pressure, and ensures the house security. It can

control the alarm system, the rooms' temperature and lighting, and monitor smoke, humidity and pressure in the rooms. It provides a security service through the use of tilt sensors to monitor doors.

The work in [11] presented smart home applications based on IoT architecture and middleware (a software layer linking the infrastructure and the CVC applications using it). The system integrated temperature and light sensors, and the user interacts with them by means of middleware called "LinkSmart" (Figure 2.2) [26] to control the brightness of the lights, the air, and entertainment systems of the house. The IoT middleware was used in the implementation of a smart home to satisfy some requirements, such as security, scalability, and capability to deal with the heterogeneity of devices.

Other researchers used mobile devices to execute the CVC application through which the user can remotely control and monitor the house SOs [19-21]. In these works, a home gateway or hub is used to aggregate all the data coming from the SOs before being

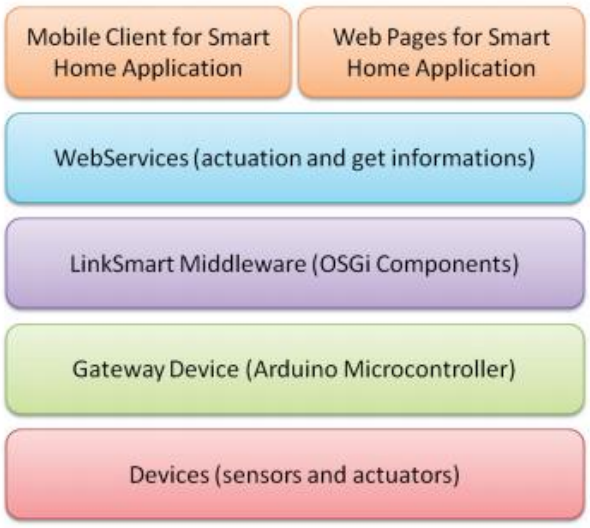


Figure 2.2 Layered architecture of the smart home application [11]

forwarded as a single stream to the CVC application. For example, the work in [20] uses a

micro-web server for home control and monitoring, with IP connectivity to remotely access and control appliances.

2.3 Cloud-Based Smart Home Systems

Cloud or cloud computing is defined as services that are delivered and consumed in real-time over the internet. These services include computation, software, data storage and access, enabling users to achieve a low-cost, scalable, and available platform [27, 28]. The services are classified into three layers as shown in Figure 2.3: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). SaaS layer communicates with developers and users in order to provide services [27, 28, 30]. PaaS is considered the middleware of the cloud and is responsible for the resource and security management to protect data and restrict user access. The IaaS layer is responsible for providing physical resources including computing power and storage space [31].

There is a complementary relationship between the cloud computing and IoT, where the IoT generates huge amounts of data, and the cloud computing infrastructure provides a pathway for that data to travel to its destination [32].

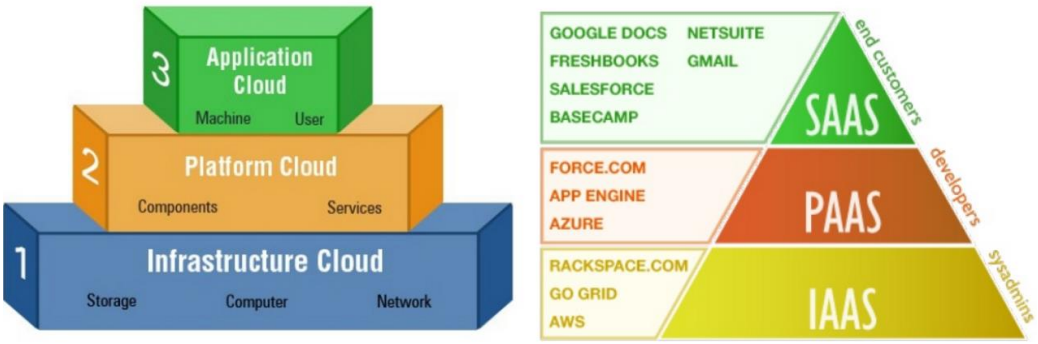


Figure 2.3 Architecture of Cloud Computing [29]

Several works focus on developing smart home devices and deploying them in the cloud environment. For example, the system in [33] integrates mobile devices with cloud networking using wireless communication for home automation. A mobile application is used as an interface with the cloud to control different appliances at home. The system in [34] measures home conditions using sensors, and manages and controls home appliances through commands sent via the cloud. The cloud is also used for storing data measured by the home devices. The work in [35] presents a smart home framework that uses a cloud data center for processing and analysis of the data collected from the home's temperature, infrared, and light sensors. The framework implements various energy consumption reduction schemes. For example, the light intensity is reduced when the surrounding light is high or the light is shut down when residents leave the room.

2.4 Smart Objects (SOs)

In general, a SO is an item that can interact with other computerized items or humans in its environment. SOs can be employed within the home environment for automation (e.g. automatically adjusting the heat in each room), monitoring (e.g. measuring carbon monoxide levels), and control (e.g. switching lights through mobile phone) to achieve a so-called smart home that optimizes comfort, security, and energy savings.

Most objects in our environment do not fall under the above-presented definition of a SO. Hence, developing mechanisms to make these objects compatible with smart homes is necessary. Some researchers have tackled this challenge. For instance, several works describe general-purpose sensor/actuator boards that can be attached to everyday objects. For example, in the "smart-Its" project, Holmquist *et al.* [36] developed a small "stick-on" computer that can be attached to un-assembled furniture parts. The purpose is to gather

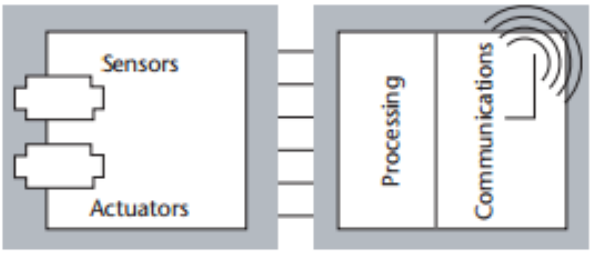


Figure 2.4 Smart its hardware consists of two separate components: a core board and sensor board [36]

information about the assembly process. Hence, two separate boards were designed (see Figure 2.4): the core board, which is composed of a processor and wireless transceiver, and the sensors board, which includes light, sound, pressure, acceleration, and temperature sensors. Two more sensors can be optionally added: camera and gas sensors. Tapia *et al.* [37] introduced a low-cost sensor called "state-change sensor". This tap-on sensor measures a change in an object's state in a home environment to recognize a user's activity. As an example, various tap-on sensors were added to a bed to track the movement and position of the user. A similar approach was taken by Kameas *et al.* [38]. They embedded hardware boards in objects (e.g., chairs, lamps, coffee jars, alarm clocks, and desks) to enable interaction between the objects and the user.

As opposed to creating a general-purpose sensor board, some researchers focused on appending sensors and/or actuators to a particular object with the purpose of employing it in a specific application. For instance, Antifakos *et al.* [39] designed a smart door handle for user identification and room access control (Figure 2.5). They used accelerometers on the door handle and the user's wrist. The person's identity is detected by measuring the

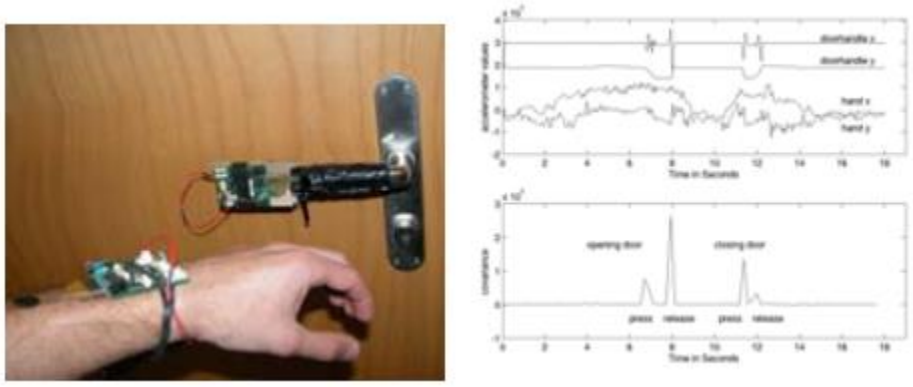


Figure 2.5 Access control example, showing the door handle and the person's wrist equipped with accelerometers [39]

correlation between the two accelerometer signals. Also, the "Mediacup" project described in [40] presents a regular coffee cup augmented with temperature and motion detection sensors, a processor, and a communication module. It studies the capture and communication of the cup's status, such as moving, stationary, full, and empty. All hardware components were integrated into a board located at the base of the cup as shown in Figure 2.6.

Figure 2.6 The Mediacup is an ordinary coffee cup with sensors, processing, and communication embedded in the base [40]

2.5 Smart Objects' Energy Consumption

SOs typically have built in or appended transducers that allow them to measure their internal processes and environment through sensors, and perform actions through actuators. In the smart home environment, SOs communicate between each other through wired or wireless mediums. Each medium type has its merits and disadvantages. Wired transducer networks are restricted by their deployment as they require physical connections snaked through walls or laid through other less aesthetically appealing methods; on the other hand, they consume less energy compared to wireless transducer networks as wired communication is much less energy demanding.

Wireless transducer networks typically have distinct energy consumption patterns that correspond to their phases of operation: sensing, processing, and communicating. The energy consumption in the first and second phases is negligible compared to that of the communicating phase. Pottie *et al.*, for example, stated that the required energy for transmitting 1KB over 100m distance is about 3 joules and is almost the same as processing 3 million operations by a 100 MIPS processor [41]. Therefore, energy consumption is considered as one of the major challenges in wireless transducer network design [42-44].

Various mechanisms are proposed by researchers to minimize or optimize the energy consumption for wirelessly communicating SOs. These mechanisms can be divided into two ways: *software* and *hardware*:

For *software solutions*, researchers have implemented algorithms in order to minimize the power consumption of wireless nodes (A SO possessing one or more transducers can be called a node). For instance, Chou *et al.* [45] presented a compression algorithm that makes use of previous readings in order to minimize communication. Other researchers

introduced power-aware routing protocols to maximize the lifetime of the transducer nodes [46, 47].

In terms of *hardware solutions*, researchers have combined transducers together in one wireless node to reduce the number of nodes in the network and consequently decrease the energy consumption of the whole network. Several researchers successfully combined multiple transducers in one wireless node. Noh *et al.* [48], for example, presented a design for a wireless node that bundles four physically interconnected transducers. Similarly, Suh *et al.* [49] proposed a wireless node composed of three transducers, while supporting the option of attaching a fourth one through a connector mechanism (Figure 2.7). In our proposed system, we adopt the hardware solution to minimize the power consumption, as we will explain later in chapter 3 and the evaluation in section 5.1.

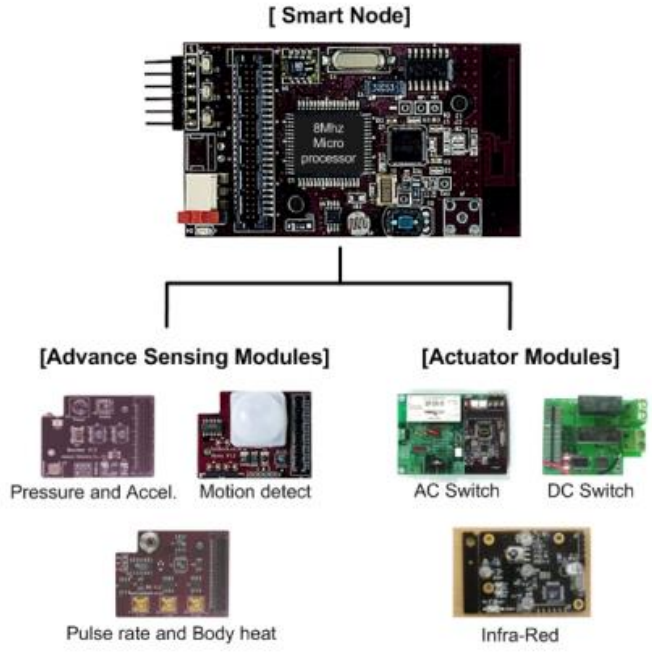


Figure 2.7 Smart node and its options modules [49]

2.6 End User Development (EUD)

The EUD paradigm has been adopted for several smart home solutions. Examples of these systems are OSCAR [50], iCAP [51], CAMP [52] and SPOK [53]. OSCAR [50] is a software tool running on a touchscreen-based tablet that allows end-users to discover and connect home network devices together, such as TV, cameras, and speakers. iCAP (interactive prototype of Context-Aware Application) [51] allows end-users to visually design smart home applications using a pen-based interface (Figure 2.8). It permits users to specify input and output devices and setup behavioral logic using "if-then" rules that describe a condition and its associated action. CAMP (Capture and Access Magnetic Poetry) [52] provides an interface consisting of words that can be clustered into rules that denote four pieces of information: who, what, where, and when. These pieces of information describe what actions the system will command in response to what stimulus and at what time. SPOK (Simple PrOgramming Kit) [53] uses a pseudo-natural language

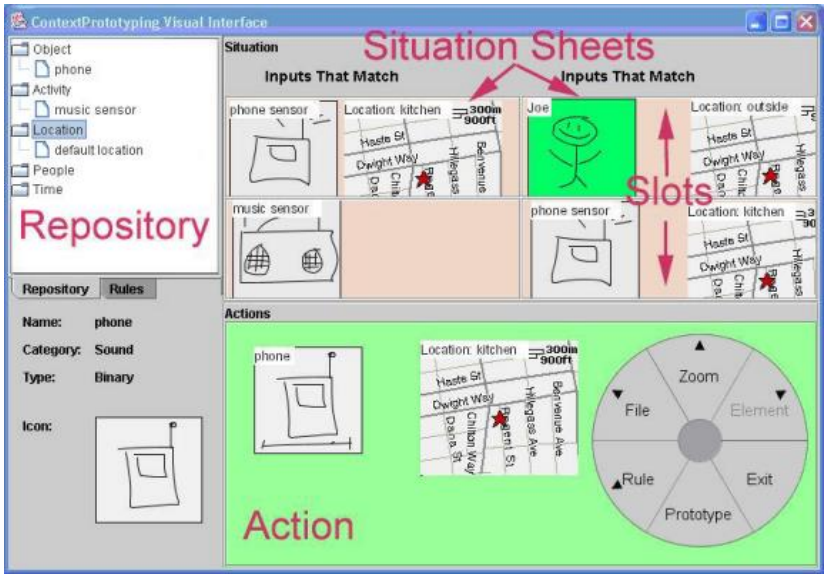


Figure 2.8 The iCAP user interface [51]

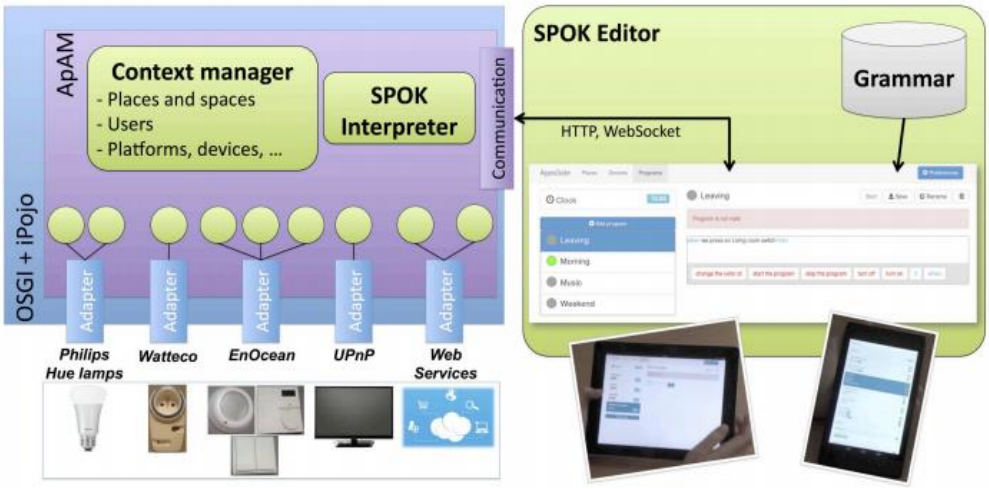


Figure 2.9 Overall software architecture and the SPOK editor running on client interaction devices [53]

(combines rule-based and imperative programming) to define event-condition-action rules to configure and control appliances (see Figure 2.9).

In addition, several commercial and freeware home boxes are now available to provide end-users with rule-based programming tools [54]. Among them are IFTTT [55], and Zipato [54, 56]. IFTTT (IF This Then That) is a web and smartphone application, uses a rule-based language mixed with text and icons. It allows the user to write "if-then" rules called "recipes" to control smart devices. The drawback of this system is that the user can define for each rule only one condition and one action. Zipato is another rule-based home management system used for home control. Like SPOK, Zipabox has been developed using Scratched-based language [57].

Nichols and Myers [58] presented a method to automatically generate user interfaces that expose CVC functions. In particular, they developed the User Interface Descriptive Language (UIDL) to describe the functionality of SOs. Using these descriptions, they devised a scheme to automatically generate interfaces to monitor and control corresponding

SOs. The user interface first downloads a description of the appliance's functions written in UIDL and then automatically creating an interface. Similarly, Mayer et al. [59] presented a modality-independent user interface generation method for IoT. Interface generation is enabled by detailed descriptions of the atomic interactive components of SOs that also capture the semantics of that interaction.

2.7 Gap Analysis

Although the concept of IoT can be applied to many applications, smart home systems are some of the most studied by researchers. Smart homes typically rely on SOs to realize the intelligence needed to monitor and control the home environment. We have surveyed several SO technologies and focused on general-purpose sensor/actuator boards that can be attached to objects [37, 38]. These technologies are comparable to the SO specification feature of the proposed system. However, the existing technologies have strict limitations on the number and type of sensors that a board can support.

Furthermore, we analyzed several typical smart home systems, the CVC logic responsible for controlling the SOs is defined programmatically. This makes the update or maintenance of such systems difficult and costly. Moreover, it would pre-empt Non-IT users from constructing such systems. The interface generation schemes presented by [58] and [59] allow users to directly monitor and control SOs. However, these systems do not support automated control of SOs based on rules of actuation.

To respond to the limitations of programmatically defined CVC logic, several EUD systems have been proposed [50, 51, 53]. These systems support rule-based behavior description. The rules define principally a condition and an action. The conditions are tests

on sensor values and the actions are commands to be sent to actuators. Existing EUD rules use crisp sensor and actuator values to define conditions and actions.

In this thesis, we present a complete smart home monitoring and control system. We call this system the Simple Internet of Things Enabler (SITE). SITE resolves the challenges stated above as follows:

- SOs can be realized by appending sensor/actuator clusters built through the GPTN onto "regular" objects. We describe the GPTN in Chapter 3.
- SOs can be controlled through the CVC using SCL or Fuzzy logic rules. The SCL rules are designed to simplify the control process. We describe the CVC, along with the proposed rule specification schemes, in Chapter 4.
- SCL advocates the use of qualitative expressions as opposed to quantitative measures for rule definition. Humans assess their environment qualitatively. For example, to describe the temperature of an object, they use qualifiers such as very hot, hot, cold, etc. Hence, intuitive qualitative expressions can simplify rule construction for end-users.

Table 2.1 summarizes some of the related work presented in the previous sections in the lights of the features discussed above.

Table 2.1 Summary of Related Work

Features	Holmquist *et al.* [36]	Tapia *et al.* [37]	Beigl *et al.* [40]	Noh *et al.* [48]	Souza *et al.* [11]	Yan *et al.* [21]	Folea *et al.* [25]	Nichols *et al.* [58]	Mayer *et al.* [59]	Dey *et al.* [51]	Truong *et al.* [52]	Coutaz *et al.* [53]	Dickey *et al* [33]	Soliman *et al.* [34]	Bin *et al* [35]	Proposed System
General purpose transducers boards attached to objects	✓	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-	✓
Plug and play mechanism in building SOs	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-	✓
End-user can construct the system hardware	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	✓
CVC logic is defined by end-user	-	-	-	-	X	X	X	X	X	✓	✓	✓	**X**	X	X	✓
Automatic control of SOs using rules	-	-	-	-	X	X	X	X	X	✓	✓	✓	X	X	X	✓
Qualitative measures for rules definitions	-	-	-	-	X	X	X	X	X	X	X	X	X	X	X	✓
Cloud-based solution	-	-	-	-	X	X	X	X	X	X	X	X	✓	✓	✓	✓

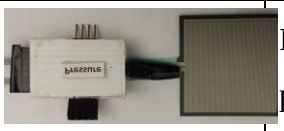
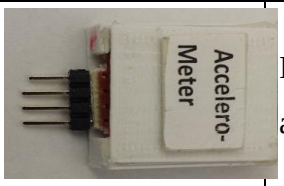
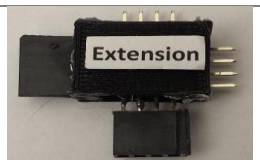
Chapter 3 **The General-Purpose Transducers Network (GPTN)**

3.1 Overview

One of the contributions of this thesis is the design and development of a general-purpose transducers network (GPTN). The GPTN permits users to convert regular objects into SOs that can wirelessly communicate with a central server. In this Chapter, we describe the GPTN, which allows the creation of network-enabled clusters of transducers that can be appended to various objects to form SOs. Each node is associated with a single network address. The clusters are formed through a plug and play mechanism where a variety of transducers can be connected to a mainboard (see Table 3.1). Therefore, the mainboard can recognize the added and removed transducers dynamically. The network supports commonly used sensors such as temperature, pressure, light, acceleration, and CO, and commonly used actuators such as on/off switch, dimmer, and vibrotactile actuator as shown in Table 3.1.

All components used to create a SO are designed as small blocks that can be interconnected in many possible configurations as shown in Figure 3.1. Each node organizes its transmitted data into Sensor Model Language (SensorML) [60] and receives control data in Actuator Model Language (ActuatorML) format as will be described later.

Table 3.1 The Mainboard and Commonly Used Transducers

I^2C ID	Transducer	Prototype	Remarks
	Wireless Node's Core (Mainboard)		Data collection, processing, and communication
1-10	Pressure Sensor		I^2C IDs 1-10 are reserved for pressure sensors (FSRs)
72-75	Temperature Sensor		I^2C IDs 72-75 are reserved for Temperature Sensors (TMP102)
83-84	Accelerometer		I^2C IDs 83-84 are reserved for accelerometer (ADXL345)
41,57	Light Sensor		I^2C IDs 41, 57 are reserved for Light sensors (TSL2561)
11-20	Vibrotactile Actuator		I^2C IDs 11-20 are reserved for vibrotactile actuators
21-30	On/Off Actuator		I^2C IDs 21-30 are reserved for on/off actuators
31-40	Dimming Actuator		I^2C IDs 31-40 are reserved for dimming actuators
	Extension		Connected to the I^2C bus when needed

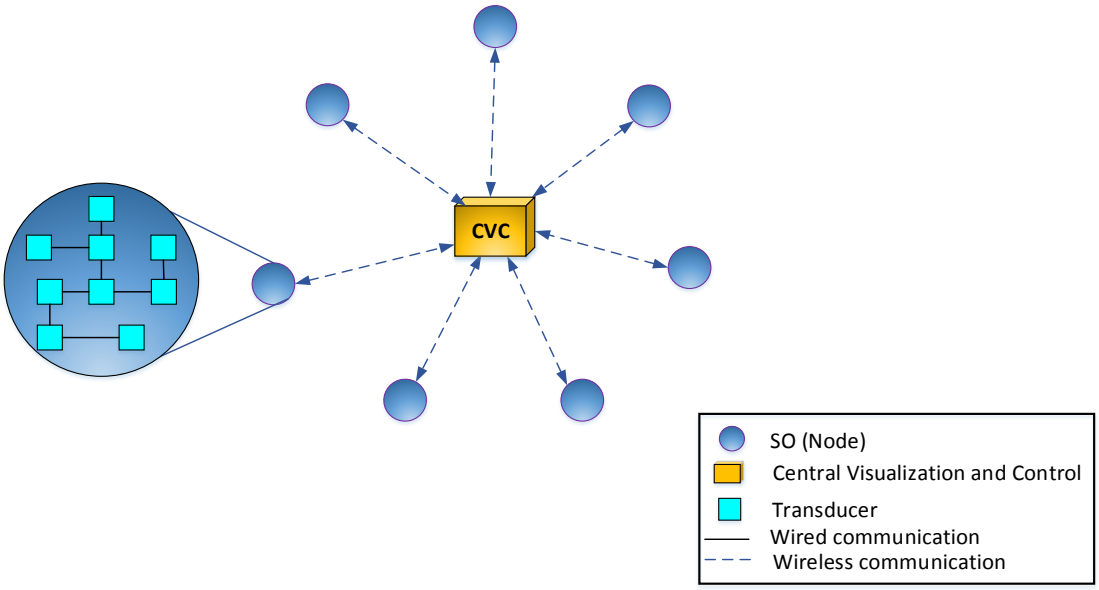


Figure 3.1 SOs with reconfigurable transducers. The transducers are interconnected to form nodes. The nodes interact wirelessly with the CVC

3.2 The GPTN Architecture and Design

Figure 3.2 shows the architecture of the GPTN. It is composed of the following five modules described below.

3.2.1 Transducers module

The transducers module represents a set of interconnected transducer units. Each unit is composed of a microcontroller, an analog to digital converter (ADC) or a digital to analog converter (DAC). A unique ID is assigned to each transducer (see Table 3.1) in order to identify it. These IDs are statically pre-assigned and are permanently stored in flash memory.

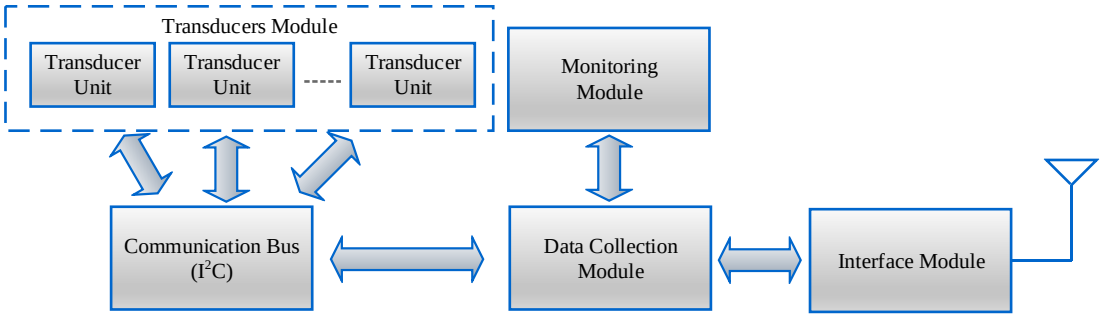


Figure 3.2 The GPTN high-level architecture

3.2. 2 Communication bus

The communication bus is responsible for providing the necessary communication between the transducers module and the data collection module. The onboard communication bus is based on the I^2C (Inter-Integrated Circuit) protocol [61].

3.2.3 Data collection module

This module provides the necessary space to collect all measured data that comes from the transducers module, and passes them to the interface module. It also collects the upcoming commands from the interface module and forwards them to the transducers module in order to activate the running actuators. Moreover, this module stores the connectivity status of the node's transducers. The connectivity status is composed of the transducers IDs and corresponding statuses (i.e., running or not).

3.2.4 Monitoring module

This module is responsible for monitoring the connection and disconnection of transducers. Figure 3.3 demonstrates the workflow of the task it performs. As shown in the figure, the module polls all assigned transducers periodically and waits for their response. When a transducer is added to the cluster, it broadcasts its unique address through the communication bus. This module recognizes this address. It records the change regarding added and/or removed transducers and sends this change to the data collection module in order to update the corresponding node's status.

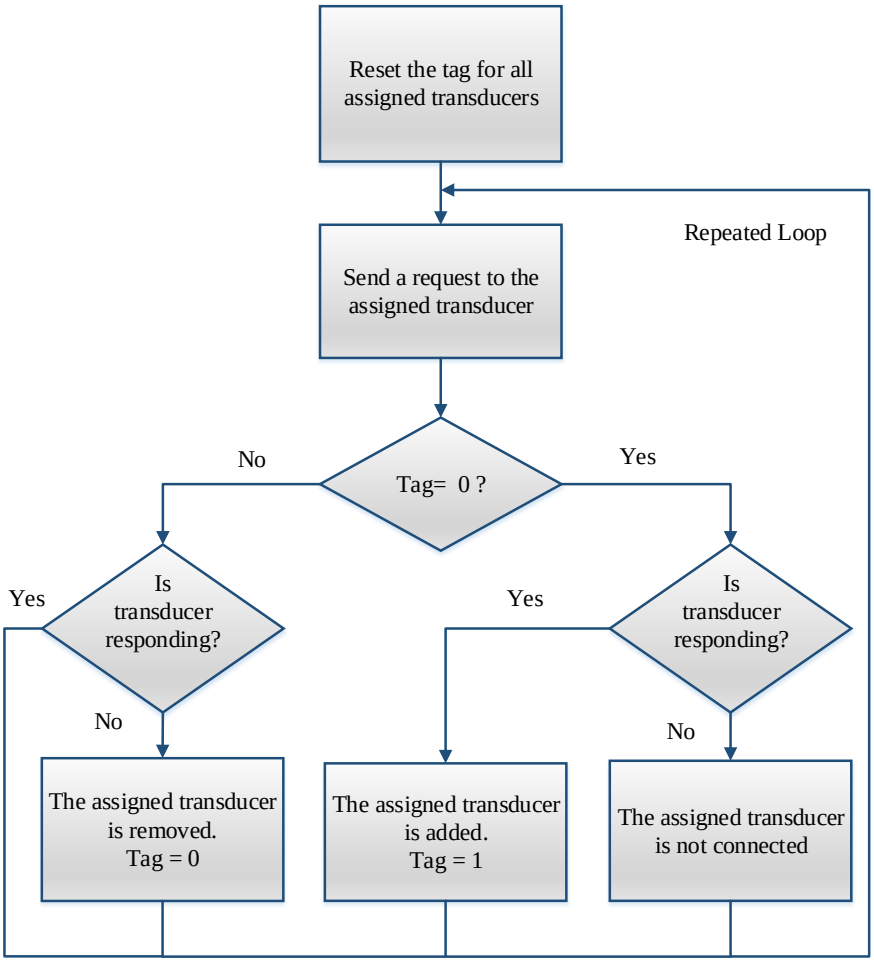


Figure 3.3 Process of monitoring the status of transducers

3.2.5 Interface module

The interface module provides the necessary communication between the transducer node and its external environment. Five possible media of communication can be used with the proposed transducer node including USB (RS232), Bluetooth technology (IEEE 802.15.1), Ethernet (IEEE 802.3), Wi-Fi (IEEE 802.11), and ZigBee (IEEE 802.15.4). Our proposed system supports Wi-Fi in order to communicate with SOs through the internet. This module is responsible for exchanging the messages between the data collection module and other nodes or the central server.

3.3 The GPTN Implementation

As previously described in section 3.1, the GPTN allows the creation of network-enabled clusters of transducers. Each cluster is composed of a mainboard and one or more plug and play transducers.

3.3.1 The GPTN Mainboard

The mainboard combines all the modules described in section 3.2 excluding the transducers module. Figure 3.4 depicts the various components that are integrated within the mainboard. The following is a description of these components:

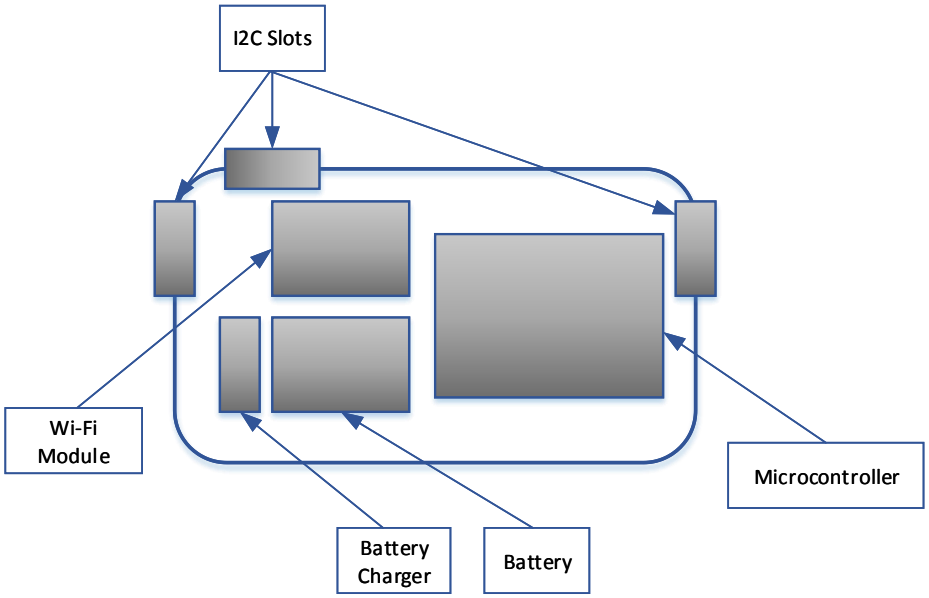


Figure 3.4 The various devices integrated into the node's mainboard

- *Microcontroller*: We used an *Arduino ProMini* [62], an open source single-chip microcontroller which has a 8-bit *Atmega 328* processor running at a 8 MHZ clock speed. The microcontroller can be programmed using a proprietary language with a syntax that is similar to the Java and C languages [35]. The Arduino software runs the data collection and monitoring modules previously described in section 3.2.
- *Wi-Fi Module*: The full-duplex communication between the mainboard (each SO has one mainboard) and the CVC is achieved using a low power *ESP8266EX Wi-Fi Transceiver* [63]. This transceiver operates in three power modes: active, sleep, and deep sleep, which makes it very widely used device in mobile, wearable, IoT and home applications and automation.
- *Battery:* The power of the whole circuitry was supplied by a *Polymer Lithium Ion* (LiPoly) Battery that produces a nominal voltage of 3.7 Volts with a capacity of 2000 mAh (milli-amp-hour) [64].

- *I^2C Slots:* In the current implementation, the transducers' node includes pressure (*FSR*, force sensitive resistor, model 402) [65], ambient light (*TEMT6000*) [66], carbon monoxide gas (*MQ-7*) [67], triple axes accelerometer (*ADXL 345*) [68], temperature (*TMP 102*) [68], and flex [69] sensors, and switch, dimmer, and vibrotactile actuators [68]. Each transducer has its assigned ID number for a unique identification and is able to communicate with each node's microcontroller through the I^2C serial communication protocol through the I^2C slots.

- *Battery Charger:* The battery charger consists of a USB LiPoly Charger that allows the charging of 3.7 V LiPo cells at a rate up to 500 mA.

3.3.2 The Home Example

To illustrate how the GPTN can be used to create and communicate SOs in a smart home environment, we introduce the example depicted in Figure 3.5. In this example, six SOs have been placed in different locations in a house. Each SO has been customized depending on the measurement environment. For example, node 1 (SO1) measures the kitchen's temperature, light intensity and carbon monoxide (CO) levels, node 2 (SO2) is placed in the refrigerator and includes pressure (placed under the eggs' tray as an example) and light (attached to the door) sensors. Node 3 (SO3) has three pressure sensors and three actuators appended on a sofa's cushions. The sensors can be used to monitor sitting patterns and durations, and the actuators can be used to buzz persons when they remain seated for too

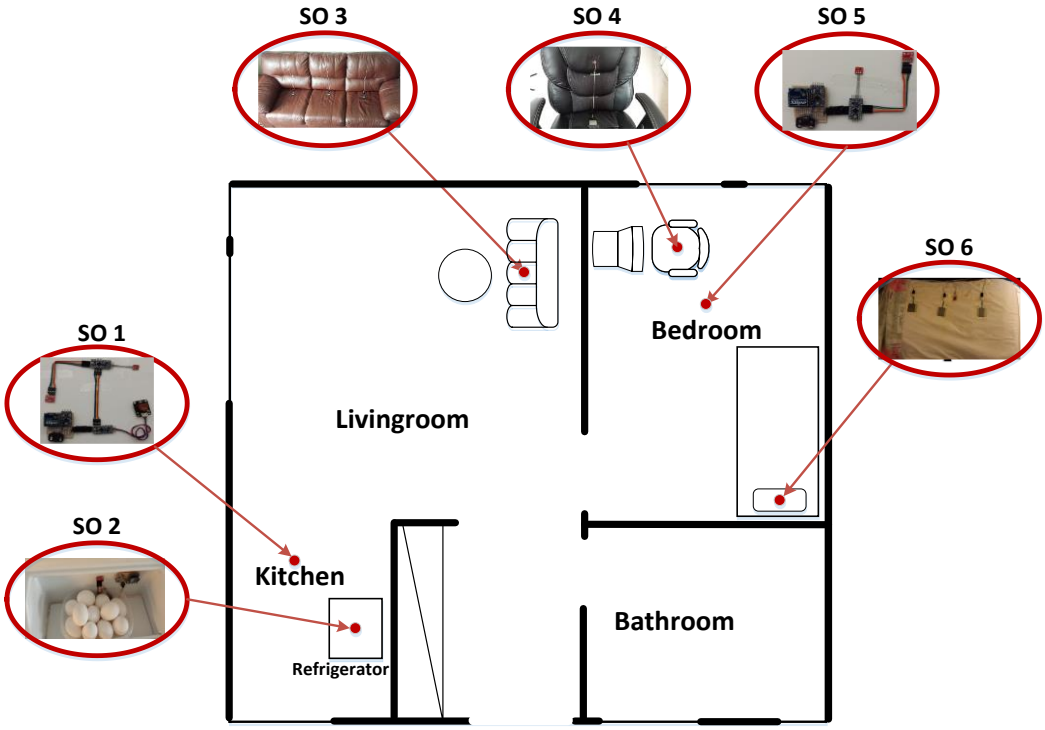


Figure 3.5 A house example with six wireless nodes with different reconfigurable transducers

long. Three nodes are located in the bedroom: node 4 (SO4) with a pressure sensor and actuator mounted on a chair seat, node 5 (SO5) measures the temperature and light intensity in the room, and node 6 (SO6) located underneath the pillow with three pressure sensors and accelerometer to measure a person's movements while sleeping. As we can see that, the same nodes' mainboard hardware and software are used with different plug and play transducers depending on the type of the physical property to be measured.

The sampling rate of the sensors used is 1Hz except for that of the accelerometer, which is 30 Hz. Each node's microcontroller sends a measurement SensorML messages to the central server through a Wi-Fi Transceiver. This SensorML message includes the node's general information (such as the communication protocol, and number and type of the

transducers involved). If we take the smart chair (SO4) as an example, Figure 3.6 shows an example of this information for this smart chair that appended with a cluster of pressure sensor, light sensor, and vibrotactile actuator. The SensorML message also includes the node's running sensors' measurement (such as the timestamp and the measured data of each sensor). Figure 3.7 shows an example of the measurement part of this message for both the pressure and light sensors for the smart chair.

The central server (CVC) monitors and stores the received measurement data and post-processes this data. The CVC (will be described in details in chapter 4) receives and stores the upcoming measured data from the nodes and prepares them for visualization to make them meaningful. It also sends ActuatorML messages to the corresponding nodes in order to activate their actuators. Figure 3.8 shows an example of the ActuatorML message that is sent from the CVC to the smart chair.

```
<generalInformation>
  <objectName>MCRLab1</objectName>
  <objectID>1</objectID>
  <noOfSensors>2</noOfSensors>
  <noOfActuators>1</noOfActuators>
  <transducer>
    <ID>1</ID>
    <type>Pressure_Sensor1</type>
    <manufacturer>VersoPoint</manufacturer>
    <samplingRate>1</samplingRate>
    <unit>Newton</unit>
    <minimum>0</minimum>
    <maximum>10</maximum>
    <sensorActuator>sensor</sensorActuator>
    <visualization>guage</visualization>
  </transducer>
  <transducer>
    <ID>57</ID>
    <type>Light_Sensor</type>
    <manufacturer>TAOS</manufacturer>
    <samplingRate>1</samplingRate>
    <unit>Lux</unit>
    <minimum>0</minimum>
    <maximum>10</maximum>
    <sensorActuator>sensor</sensorActuator>
    <visualization>tank</visualization>
  </transducer>
  <transducer>
    <ID>6</ID>
    <type>Actuator1</type>
    <manufacturer>Precision Microdrives</manufacturer>
    <samplingRate>1</samplingRate>
    <unit>NA</unit>
    <minimum>0</minimum>
    <maximum>255</maximum>
    <sensorActuator>Actuator</sensorActuator>
    <visualization>NA</visualization>
  </transducer>
</generalInformation>
```

Figure 3.6 Example of SensorML general information for a smart chair with two (pressure and light) sensors and one vibrotactile actuator

```
<measurements    time_sec= "984"    NoOfSensors= "2">
   <sensor>
      <ID>1</ID>
      <noOfValues>1</noOfValues>
      <name>Pressure_Sensor1</name>
      <value>8</value>
   </sensor>
   <sensor>
      <ID>57</ID>
      <noOfValues>1</noOfValues>
      <name>Light_Sensor</name>
      <value>3</value>
   </sensor>
 </measurements>
```

Figure 3.7 Example of SensorML measurements for the pressure and light sensors appended onto the smart chair.

```
<actuatorControl>
   <objectId>1</objectId>
   <actuatorId>6</actuatorId>
   <intensity>225</intensity>
   <frequency>50</frequency>
   <period>1</period>
 </actuatorControl>
```

Figure 3.8 Example of an ActuatorML message sent from the CVC to control the smart chair's actuator

3.4 Summary

This chapter presented the GPTN framework architecture. We have explained the plug and play mechanism, and how the framework can recognize the added and removed transducers. We also have described the hardware implementation of the framework. In addition, we presented a home example, where multiple GPTN clusters were appended to objects to form SOs. In chapter 4, we make use these SOs to interact with the SITE CVC system.

Chapter 4 **The Proposed SITE System**

4.1 SITE Overview

SITE interacts with two types of entities: users and SOs. We define a user as an individual that:

- Creates SOs using the GPTN,
- Configures a smart environment using the SITE CVC,
- Sends commands to SOs through the CVC, and/or
- Visualizes the information produced by SOs using the SITE CVC.

To configure a smart environment, the user specifies what SITE CVC should do in response to the data received from the SOs' sensors. For example, consider the following SO: an office chair equipped with a pressure sensor and a vibrotactile actuator, both integrated into its seat cushion. The user can configure SITE CVC to send a command to the actuator to vibrate if the pressure is detected to be high (i.e. a person is sitting on the chair) for a couple of hours. The vibration would remind the seated person to take a walk. Figure 4.1 shows the high-level use cases supported by SITE. In order to setup a smart environment, the user performs the following:

- Build SOs using the GPTN and deploy them in the environment.
- Use the SITE CVC to:

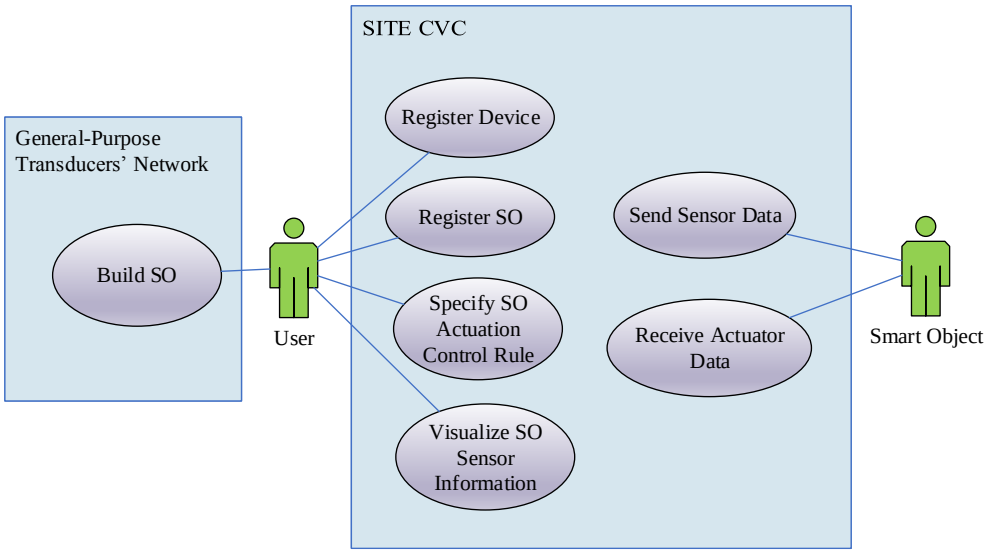


Figure 4.1 SITE UML use case diagram.

- Register the computing devices that have the authority to access the CVC by supplying their MAC addresses.
- Register the available SOs by supplying their name, IP address, and geographical coordinates.
- Configure all or a subset of the registered SOs by defining rules to control them through the CVC.
- Send Commands through the CVC to control SOs.
- Visualize SO sensors' information.

4.2 CVC Communication

In the proposed SITE system, the SOs are communicating directly with the CVC. The CVC can run on a local server or cloud platform. The user can configure and control these SOs through a computing device such as mobile phone, tablet, or PC. The computing device

forwards user commands to the CVC, and the CVC, in turn, relays them to SOs (see Figure 4.2). Therefore, we must define a communication protocol between the CVC and computing device.

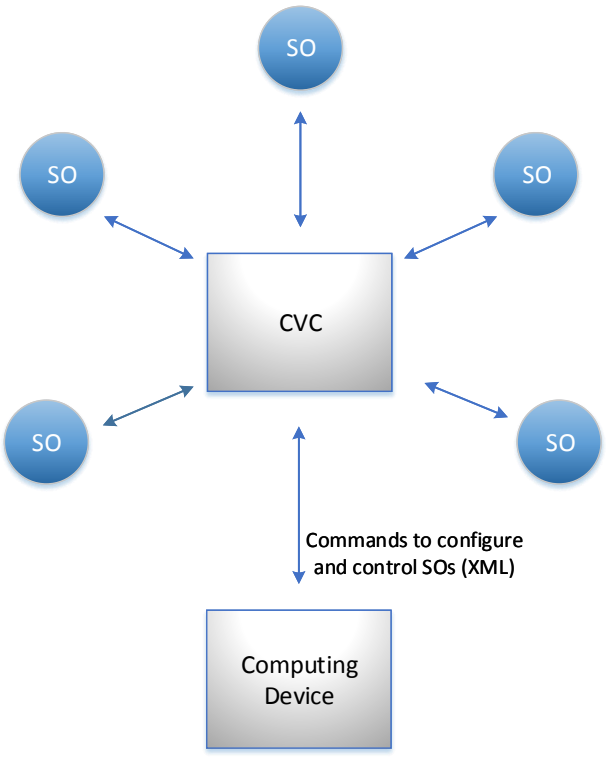


Figure 4.2 Controlling and configuring SOs by a controlling device through the SITE CVC

4.2.1 CVC-Computing Device Communication Protocol

Figure 4.3 illustrates the proposed communication protocol between a computing device and the SITE CVC. The protocol defines three principle message sequences:

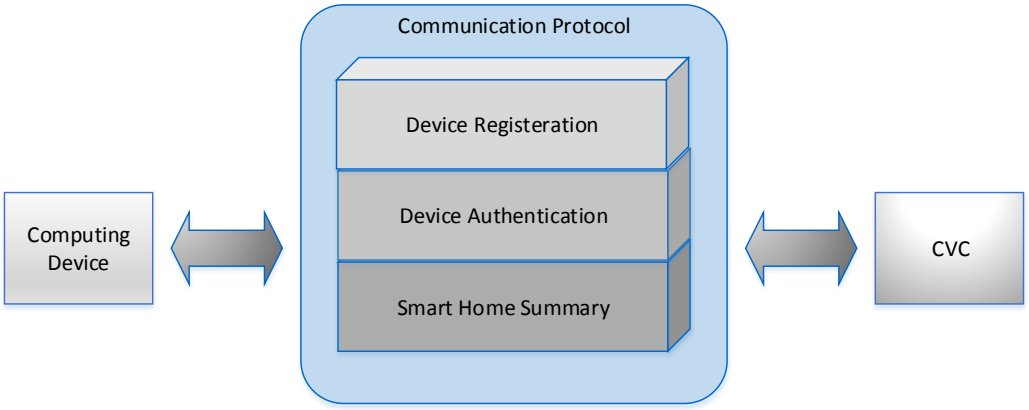


Figure 4.3 Communication protocol between a computing device and the SITE

1. *Device registration*. This message sequence allows computing devices such as laptops, smart phones, and tablets to register as authorized devices with the CVC. This will allow these devices to control SOs inside the smart home. We define two categories of authorized devices: administrator and standard devices. The administrator device is capable of configuring the CVC's controller, and adding or removing authorized standard computing devices. The administrator and standard computing devices can both view SOs' sensor measurements and send actuation commands to SOs. Initially, one administrator device is registered with the CVC at deployment time. From there, other devices' registrations are always initiated by the administrator device, which sends the MAC address of the device to be registered to the CVC. The CVC in turn stores that address in its database, as will be explained in section 4.3.
2. *Device Authentication*. When a computing device requests access to the CVC, the CVC requests the MAC address of the device. Upon receiving the MAC address, the CVC lookups this address in its Static Information Database (SID) (will be described in detail in section 4.3.10). Based on the results received from the database, the CVC

will accept or deny the request. Figure 4.4 shows the sequence diagram of this authentication process.

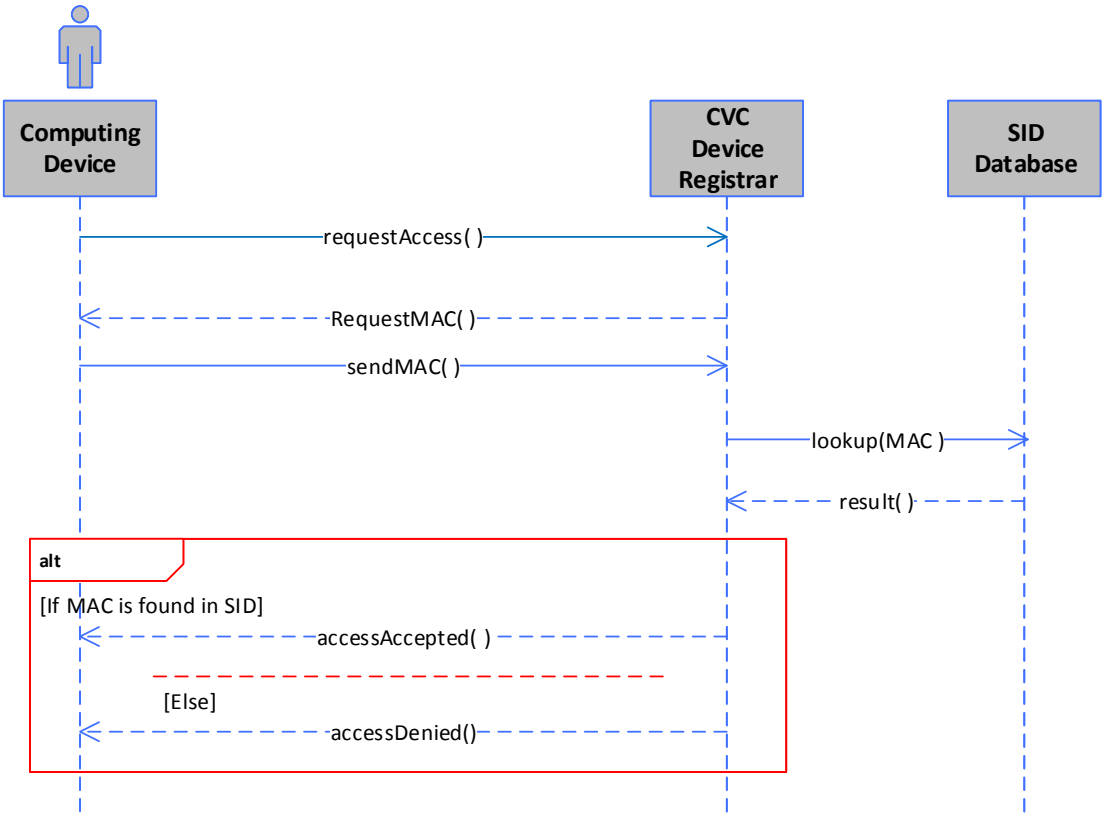


Figure 4.4 The sequence diagram of a computing device authentication process

3. *Smart home summary*. The CVC sends a smart home summary message (in XML format) describing the status of the smart home environment. This message is similar to the SensorML information message discussed in section 3.3.2. The only difference is that the previous SensorML message was describing general information for only one SO, whereas the summary message is for all the SOs in the smart home environment. Figure 4.5 shows an example of a summary message of a smart home composed of two SOs, each SO is appended with two transducers. The summary

message is a one-way communication from the CVC to the computing device. When a user adds or removes SOs, through the administrator computing device, or even updates the number of sensors or actuators on a SO, the CVC will recognize the update and send a new summary message.

```
<smartHomeSummary>
    <noOfSOs>2</noOfSOs>
    <SO>
        <objectName>SmartChair</objectName>
        <objectID>1</objectID>
        <noOfSensors>1</noOfSensors>
        <noOfActuators>1</noOfActuators>
        <transducer>
            <ID>1</ID>
            <type>Pressure_Sensor1</type>
            <manufacturer>VersoPoint</manufacturer>
            <samplingRate>1</samplingRate>
            <unit>Newton</unit>
            <minimum>0</minimum>
            <maximum>10</maximum>
            <sensorActuator>sensor</sensorActuator>
            <visualization>guage</visualization>
        </transducer>
        <transducer>
            <ID>6</ID>
            <type>Actuator1</type>
            <manufacturer>Precision Microdrives</manufacturer>
            <samplingRate>1</samplingRate>
            <unit>NA</unit>
            <minimum>0</minimum>
            <maximum>255</maximum>
            <sensorActuator>Actuator</sensorActuator>
            <visualization>NA</visualization>
        </transducer>
    </SO>
    <SO>
        <objectName>SmartFridge</objectName>
        <objectID>2</objectID>
        <noOfSensors>1</noOfSensors>
        <noOfActuators>1</noOfActuators>
        <transducer>
            <ID>57</ID>
            <type>Light_Sensor</type>
            <manufacturer>TAOS</manufacturer>
            <samplingRate>1</samplingRate>
            <unit>Lux</unit>
            <minimum>0</minimum>
```

```
            <maximum>500</maximum><sensorActuator>sensor</sensorActuator>
            <visualization>tank</visualization>
            <minimum>0</minimum>
            <maximum>500</maximum>
            <sensorActuator>sensor</sensorActuator>
            <visualization>tank</visualization>
        </transducer>
        <transducer>
            <ID>7</ID>
            <type>Actuator2</type>
            <manufacturer>Precision Microdrives</manufacturer>
            <samplingRate>1</samplingRate>
            <unit>NA</unit>
            <minimum>0</minimum>
            <maximum>255</maximum>
            <sensorActuator>Actuator</sensorActuator>
            <visualization>NA</visualization>
        </transducer>
    </SO>
</smartHomeSummary>
```

Figure 4.5 Example of a home summary message in XML format for smart home composed of two SOs, each SO has two transducers

4.2.2 Cloud-Based CVC

The proposed SITE CVC is a framework that interacts with smart devices in the smart home environment. Therefore, the CVC is deployed on a server that is constantly operational. The server can be a local (inside the house) or on the cloud. Deploying the CVC on the cloud has many benefits that are inherent to cloud computing compared to local servers, including:

- Data security,
- Robustness, and,
- The elasticity of resources.

Therefore, multiple CVCs for different smart homes can be running on Virtual Machines (VMs) on a cloud platform such as amazon [70] or Google [71] cloud as shown

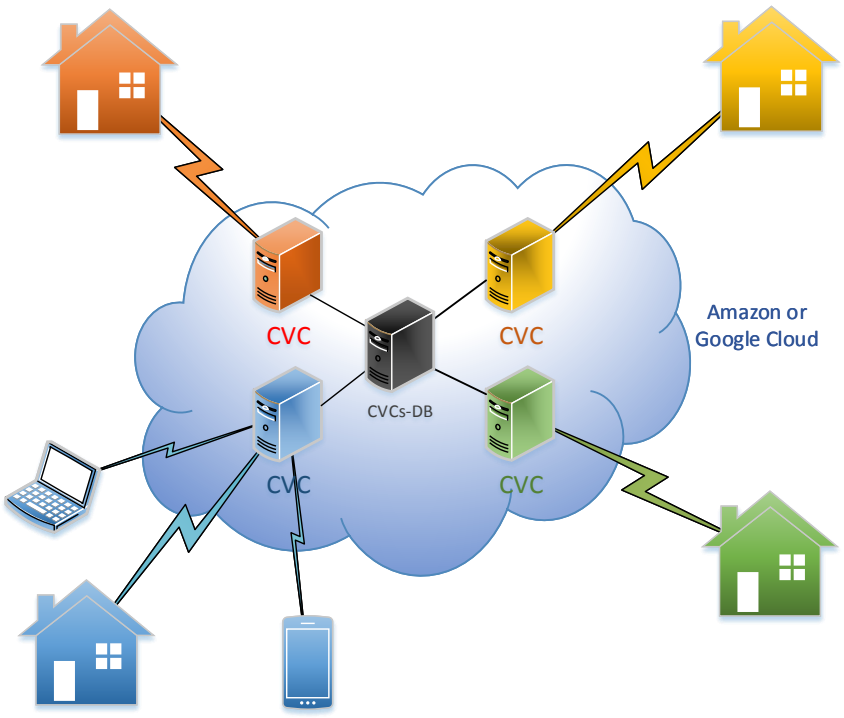


Figure 4.6 Cloud-based CVC

in Figure 4.6. The user can login to a VM, setup rules (will be described later in section 4.3) to control SOs automatically and register a mobile device to control the smart home. Therefore, any command sent from the mobile device goes to the CVC running on the cloud. In response, the CVC will send commands back to the house SOs.

4.2.3 Cloud-Based CVC Security

Cloud computing is considered a viable solution to shift the cyber security responsibility from the home users to the cloud computing service providers [72]. Two kinds of security concerns have to be addressed when deploying the CVC on the cloud. The first concern is how to prevent unauthorized computing devices from accessing the CVC. The second

concern is how to protect SOs of a smart home from being registered into the CVC of another smart home.

As previously described in section 2.4, one of the responsibilities of the second layer (PaaS) of the cloud computing architecture is managing the security of the applications to be deployed on cloud, where the service provider of the cloud implements application–level security control.

To improve the security of SOs in the smart home environment, every SO should be segmented into its network and has network access restricted. Therefore, the second security concern can be resolved by integrating a database on the cloud which can be accessed by all the CVCs deployed in the cloud. We call this database the CVCs-DB (see Figure 4.6). The CVCs-DB stores the global IP address of all the registered SOs along with the corresponding CVC. Therefore, when a user wants to register a new SO to his/her home CVC, the CVC will first access the CVCs-DB to make sure that this IP address is not registered by any other CVC. The CVC then stores this IP address in that database. This registration process is provided by the CVC's SO registrar described in the following section.

4.3 CVC Architecture

The CVC architecture is designed in a manner to be independent of whether the CVC Server is located on the cloud or locally inside the house. When the CVC is deployed as a local server, static IP addresses are assigned to the SOs. However, when the CVC is deployed on the cloud, each SO is addressed by its unique global IP address. Figure 4.7 illustrates a high-level architecture of the proposed SITE CVC, which is composed of

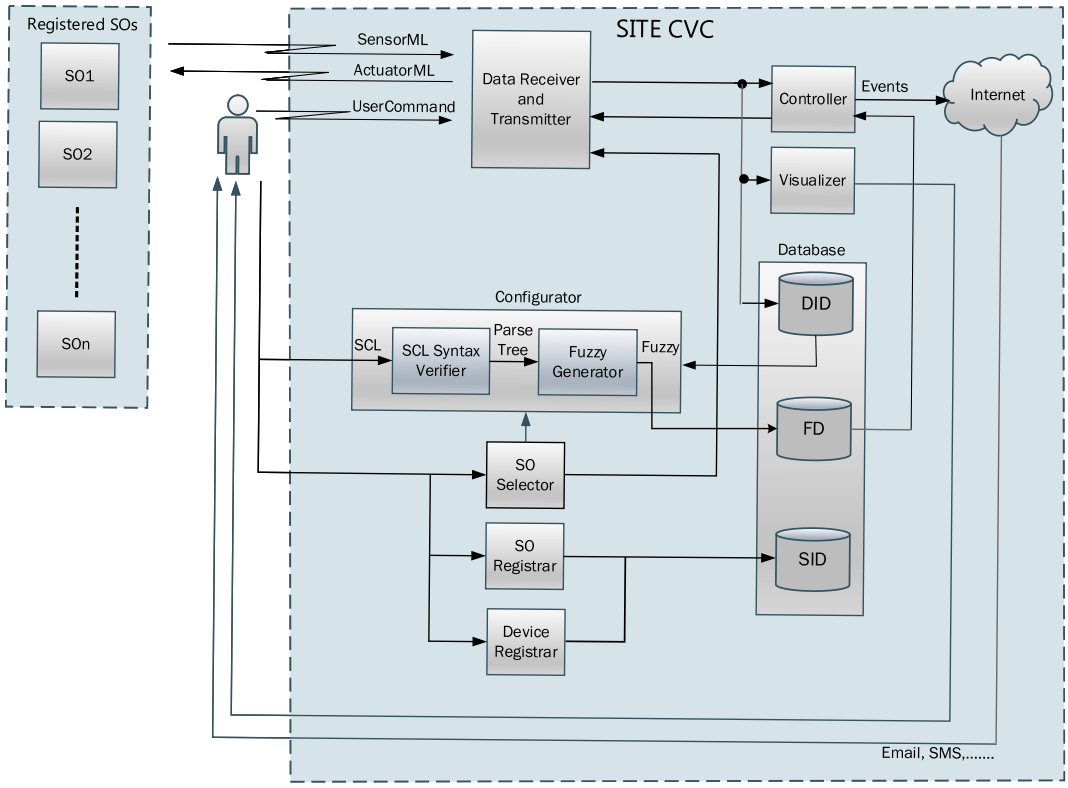


Figure 4.7 CVC high-level architecture

several functional components that we describe in the following sections.

4.3.1 Device Registrar

As previously described in section 4.2.1, the first layer of the communication protocol between the CVC and any computing device is to register the computing device for authentication purposes. Therefore, this component is responsible for registering the MAC address of every authorized computing device and store it in the SID database. In addition, this component accepts or denies any access request comes from a computing device by comparing the MAC address of the device with those stored in the SID database.

4.3.2 SO Registrar

The SOs available to the SITE CVC are registered by the user (computing device) through the SO Registrar component. During registration, the user specifies the SO name and IP address. All collected information is stored in the SID database. In the case where the CVC is deployed on cloud, this component, and for security reasons explained previously in section 4.2.1, initially tests whether the SO's IP address is already stored in the CVCs-DB (Figure 4.6) or not. If the IP address is not in the database, the SO registrar component registers the SO and forwards its IP address to the CVCs-DB.

4.3.3 SO Selector

The SO Selector allows the user to select a subset of the registered SOs whose information is stored in the SID for a particular configuration profile. A configuration profile specifies which available SOs are monitored and controlled and how they are actuated. Furthermore, the SO Selector sends a message, through the Data Transmitter, querying the selected SOs about their dynamic information. Dynamic information refers to the number and type of each SO's sensors and actuators (since transducers can be added or removed dynamically through a plug and play mechanism) and real-time sensor measurements. All received dynamic information is stored in the Dynamic Information Database (DID).

4.3.4 Configurator

SOs control definition in SITE is based on the Fuzzy logic theory. Fuzzy set theory [73, 74] is designed to mimic the human reasoning mechanism. For example, it is much easier for humans to think about room temperature in qualifying terms such as hot or cold, as opposed to specifying crisp thresholds that describe the room's thermal state. Therefore,

from a usability perspective, using Fuzzy logic will render the configuration of a smart environment more intuitive, especially to users with little to no computer programming background.

There are a plethora of Fuzzy controller languages [75], however, these controllers are suitable for engineering applications and require technical experience. However, SITE supports users with no formal technical training. Therefore, we introduce SCL (Simple Control Language), a rule-based language that allows users to define actions that are performed by actuators in response to sensor data or user commands. SCL will be described in the next section.

The user can set up SO control rules using one of three modes: Form-Based, Editor-Based, and Advanced. We estimate that users that possess a background in programming or IT, in general, can learn SCL or Fuzzy logic rule creation much faster than other users. However, the goal of SITE is to support the largest set of possible users. Nonetheless, the advanced mode provides the user with more fine-tuned control capabilities.

In the Form-Based mode, the user defines the control rules using SCL. However, instead of providing the user with a text editor for writing SCL, a graphical editor is employed. This editor allows the user to build SCL rules incrementally by filling a form. Hence, users do not have to learn the structure of SCL. In the Editor-Based mode, the user specifies SCL using a text editor. The SITE controller is based completely on Fuzzy logic. Hence, as the last step, in both the Form- and Editor-Based modes, the SCL rules are automatically translated into Fuzzy rules associated with membership functions through the Fuzzy Generator. In the Advanced mode, the user specifies Fuzzy rules and creates membership functions directly without using SCL. Hence, the Fuzzy Generator is not used in the latter

mode. In any case, at the end of the configuration, the Fuzzy rules and membership functions are saved in the Fuzzy Database (FD).

4.3.5 SCL Syntax Verifier

This component (see Figure 4.8) is responsible for verifying the syntax of the SCL rules before the Fuzzy Generator translates them into Fuzzy rules and membership functions. Hence, the SCL rules are processed through two stages by the modules:

- Lexical Analyzer: to convert the series of characters in SCL into tokens using regular expressions, and
- Syntax Analyzer: to ensure that the SCL rules comply with the SCL context-free grammar specified in Listing 4.1 and build a parse tree.

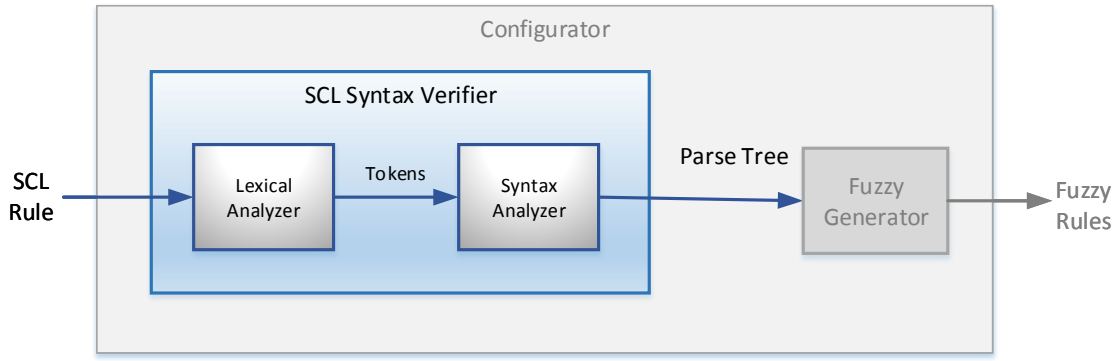


Figure 4.8 SCL syntax verifier block diagram

Listing 4.1 illustrates the SCL context-free grammar. Note that in the context-free grammar of Listing 4.1, terminals are surrounded by quotations. Visualization of the SCL grammar is provided as a syntax diagram in Figure 4.9, where dark yellow rounded rectangles are terminal symbols. This syntax diagram is generated based on the grammar using Railroad Diagram Generator [76]. The detailed syntax diagram is shown in Appendix A.

Listing 4.1 SCL Context-Free Grammar

```
scl::= 'when' condition 'then' action
condition::= sensorCondition | commandCondition
commandCondition::= 'user Command is' commandId
commandId::='alphabet numbers'
sensorCondition::= 'sensor' sensorName 'is' sensorLevel period? conditionPrime
conditionPrime::= operator condition| ε
operator::= 'and' | 'or'
period::= 'for' positiveInteger timeUnit
timeUnit::= 'seconds'|'minutes'|'hours'
action::= (msg | actuate) ('also' msg | actuate)+
actuate::= 'turn actuator' actuatorName actuatorLevel
msg ::= 'send message:' message 'to' msgDestination
msgDestination ::=  phoneNumber | email
sensorName::= SOName '. ' sensorType positiveInteger
sensorType::= 'pressure' | 'humidity' | 'carbonMonoxide' | 'light'
  | 'acceleration'
actuatorName ::= SOName '.actuator' positiveInteger
sensorLevel::= 'high' | 'medium' | 'low'
actuatorLevel::= 'on' | 'off'| 'high' | 'medium' | 'low'
digit::='0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
positiveInteger::=digit+
```

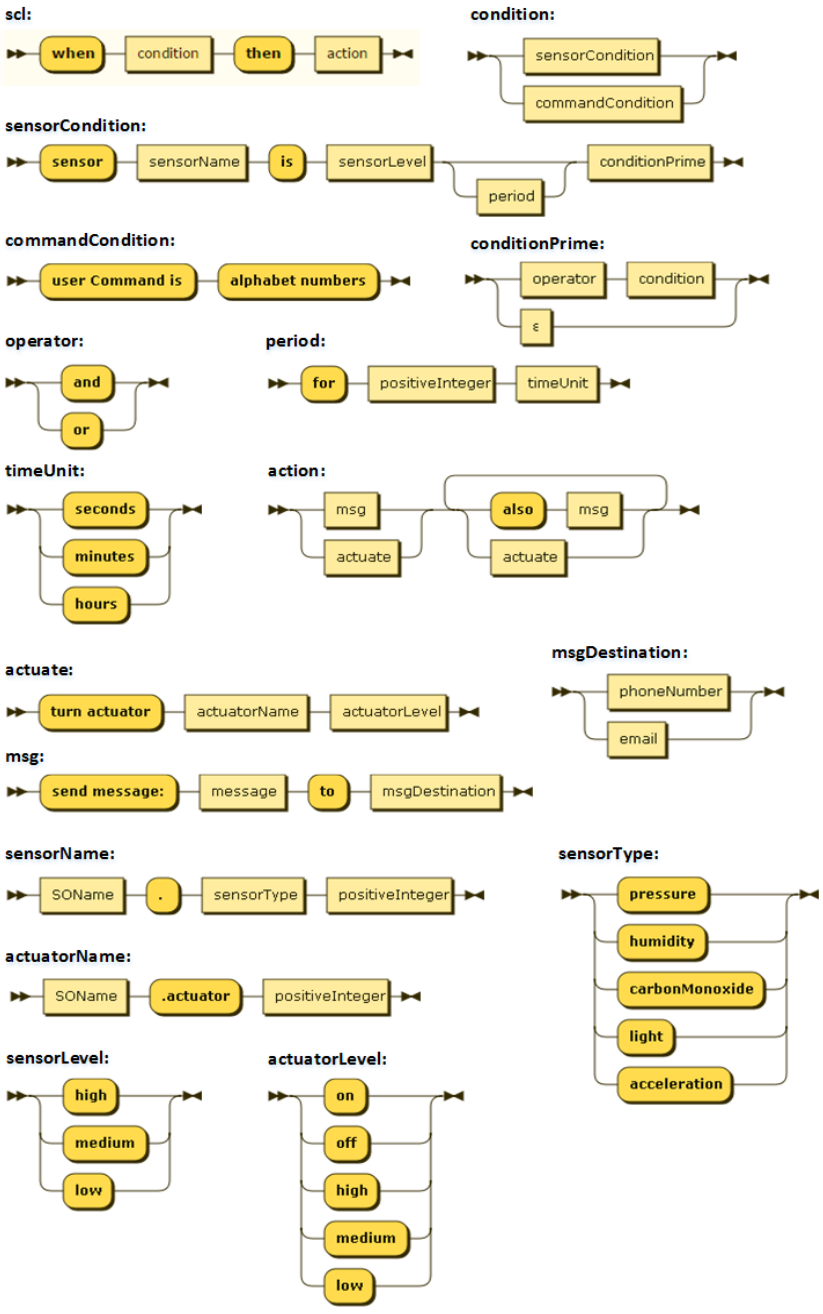


Figure 4.9 SCL syntax diagram

4.3.6 Fuzzy Generator

This component is responsible for generating the membership functions and Fuzzy rules based on the SCL rules. It receives three inputs; SCL rules organized in a parse tree from the SCL Syntax Verifier and a list of available SOs along with their sensors and actuators

from the DID. The Fuzzy Generator component produces Fuzzy membership functions for the sensors, user commands, and actuators referenced in SCL and then produces Fuzzy rules that match the logic described in the SCL.

4.3.6.1 Generating Membership Functions

The procedure of generating membership functions for the sensors is listed in Algorithm 4.1. The most common types of membership functions (triangle and trapezoid) are supported. The triangular type membership function has the following form [77, 78]:

$$\mu(x) = \max\left(\min\left(\frac{x - P1}{P2 - P1}, \frac{P3 - x}{P3 - P2} \right), 0 \right) \quad \textbf{(4.1)}$$

Where x is the current value and P1, P2, and P3 are the x coordinates for the three corners of each triangular membership function. And the trapezoidal type membership function is specified by four parameters (P1, P2, P3, and P4) as follows:

$$\mu(x) = \max\left(\min\left(\frac{x - P1}{P2 - P1}, 1, \frac{P4 - x}{P4 - P3} \right), 0 \right) \quad \textbf{(4.2)}$$

We used a union set operation (Equation 4.3) to obtain the largest membership value of the elements in either set:

$$\mu A \cup B(x) = \max\left(\mu A(x), \mu B(x) \right) \quad \textbf{(4.3)}$$

Each transducer can have up to 6 levels in its associated membership function. The range of the membership levels (Ca and Cb) for each sensor is calculated using equations 4.4 and

4.5 respectively, based on the number of membership levels per sensor and the maximum measurable value of the sensor (S_{MAX}). S_{MAX} is retrieved from the DID.

$$\text{Ca} = j*(\frac{1}{2L-1})*S_{MAX} \quad \textbf{(4.4)}$$

$$\text{Cb} = (j+3)(\frac{1}{2L-1})*S_{MAX} \quad \textbf{(4.5)}$$

Where j is the level number (starting from 0), and L is a number of levels. For example for three-level membership function (L= 3, and j ranges from 0 to 2), Ca and Cb for each level are illustrated in Table 4.1.

Applying Equation 4.1 to the triangular type membership function, the corner points (P1, P2, and P3) for each triangle are calculated as follows (the same procedure is applied to trapezoidal membership functions using Equation 4.2):

P1 = Ca
If (Ca = 0) *// The first left triangle*
P2 = 0
P3 = Cb

Table 4.1 The Range Values for a Three-Level Membership Function

Membership levels	**Ca**	**Cb**
1	0	0.6 * S_{MAX}
2	0.2 * S_{MAX}	0.8 * S_{MAX}
3	0.4 * S_{MAX}	1 * S_{MAX}

Else if (Cb = S_{MAX}) // The last right triangle

P2 = P3 = S_{MAX}

Else // Triangle is one of the middles

P2=Ca + 0.5 (Cb-Ca)

P3=Cb

Since the actuators (switches, vibrotactile actuators, warning messages, timeouts) have only two controlled values (for example "on" and "off" in the case of switches), their membership functions are of the "singleton" type and the number of these membership functions per actuator is set to 2. However, in the case of the Advanced mode, the actuators' membership functions can be any of the common types (singleton, triangle, or trapezoid). Examples of these membership functions are illustrated in Figure 4.10. As shown in the figure, the timeout has both input and output membership functions. Timeouts are used in SITE to trigger actuation events in response to the passage of time.

Algorithm 4.1 Membership Function Generation

Input: S_{MAX} [n] // maximum sensor value, n is number of sensors
μT [n][l] // membership functions type
Output: μ[n][L] // membership function, L is number of levels
Procedure:
for i = 0 to n-1 **do**
for j = 0 to L-1 **do**
Ca ← $j*(\frac{1}{2L-1})*S_{MAX}$ // the minimum value of the membership function level
Cb ← $(j+3)(\frac{1}{2L-1})*S_{MAX}$ //the maximum value of the membership function level
if (μT [i][j] type is Triangle) **then**
P1 ← Ca
P3 ← Cb
if (μT is the first left) **then** // i.e. j = 0
P2 ← P1
else if (μT is the last right) **then** // i.e. j = L-1
P2 ← P3
else // i.e. μT is one of the middles
P2 ← $C_a+\frac{1}{2}(C_b-C_a)$
end if
μ [i][j](x) ← $\max(\min(\frac{x-P1}{P2-P1},\frac{P3-x}{P3-P2}),0)$
else if (μT [i][j] type is Trapezoid) **then**
P1 ← Ca
P4 ← Cb
if μT is the first left) **then** // i.e. j = 0
P2 ← P1
P3 ← $C_a+\frac{3}{4}(C_b-C_a)$
else if (μT is the last right) **then** // i.e. j = L-1
P2 ←$C_a+\frac{1}{4}(C_b-C_a)$
P3 ← P4
else // i.e. μT is one of the middles
P2 ← $C_a+\frac{1}{4}(C_b-C_a)$
P3←$C_a+\frac{3}{4}(C_b-C_a)$
end if
μ [i][j] (x) ← $\max(\min(\frac{x-P1}{P2-P1},1,\frac{P4-x}{P4-P3}),0)$
end if
end for
end for

S_{MAX}

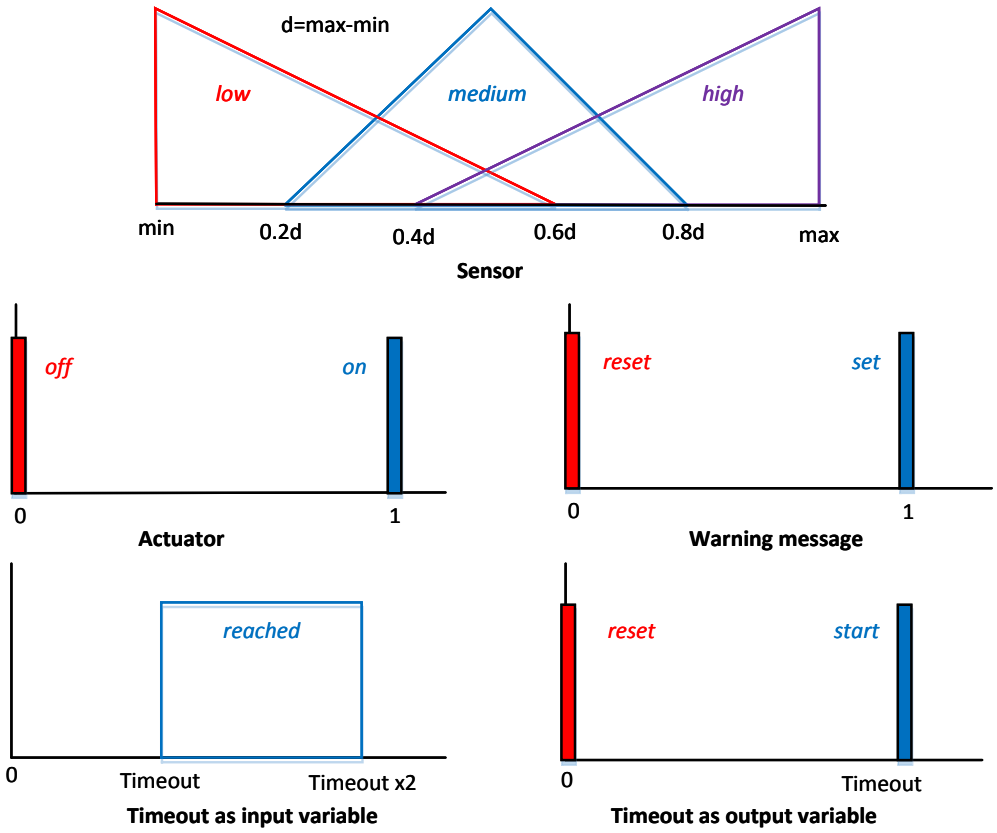


Figure 4.10 Input and output generated membership functions

4.3.6.2 Generating Fuzzy Rules

The Fuzzy Generator produces Fuzzy rules based primarily on the logical statement in the SCL rule. However, it also produces rules necessary for starting and stopping timers. Also, the Fuzzy Generator creates rules for stopping actuation when the condition that triggered its start is no longer valid. The latter details are not specified in SCL to increase its usability.

Hence, when using SCL, the user does not have to specify logic for starting, stopping, and resetting timers or actuators. The Fuzzy Generator automatically generates Fuzzy rules

Algorithm 4.2 Fuzzy Rules Generation

```
Input:  SCL[s]  // the list of SCL rules, s is the total number of SCL rules
Output: R[q]    // Fuzzy rules
Procedure:
k ← 0   // gives a number to the timeout.
for i = 0 to s-1 do
  c ← 0   // gives a Fuzzy rule's number.
  condition ← SCL[i].getCondition()   // returns the SCL condition part.
  action ← SCL[i].getAction()   // returns the SCL action part.
  reverseAction ← SCL[i].getReverseAction()   // these actions are the
      opposite of the SCL rule's actions, such as turn vibrator off, don't send
      email, and so on.
  sensor[n] ← condition.getSensors()   // returns the list of sensors in the condition
      part.
  sensorμ[n] ← condition.getSensorμs()   // returns the list of the membership
      functions associated to the sensors list.
  actuator[m] ← action.getActuators()   // returns the list of the actuators in the
      action part.
  actuatorμ[m] ← action.getActuatorsμs()   // returns the list of the actuators
      membership functions associated to the actuators list.
  FuzzyCondition ← condition.convertToFuzzyCondition()   // removes
          unnecessary keywords and attaches timer labels to timers among
          other simple adjustments in order to render the SCL condition
          compatible with the Fuzzy format.
   for j = 0 to n-1 do
          if (sensor[j] has timeout) then
            R[c++] ← IF Sensor[j] IS sensorμ[j] THEN timeout[k++]
                IS start
            R[c++] ← IF Sensor[j] IS complement(sensorμ[j]) THEN
                timeout[k++] IS reset
          end if
   end for
   R[c++] ← IF FuzzyCondition THEN action()
   R[c++] ← IF complement(FuzzyCondition) THEN reverseAction()
end for
```

to cover these tasks. Algorithm 4.2 details the steps of extracting the relevant information from SCL rule to generate corresponding Fuzzy rules. Table 4.2 depicts examples of three SCL rules and their translation to Fuzzy rules.

Table 4.2 Sample SCL Rules with Their Fuzzy Rules Set Translation

	SCL Rule	**Fuzzy Rules Set**	
1	when userCommand is Cmd1 turn actuator A on	1	**IF** Cmd1 **IS** on **THEN** A **IS** on
		2	**IF** Cmd1 **IS** off **THEN** A **IS** off
2	when sensor S1 is high and (sensor S2 is high for 10 minutes) then send this message: "This is a test message" to 613-123-4567	1	**IF** S2 **IS** high **THEN** Timeout1 **IS** start
		2	**IF** S2 **IS** low **THEN** Timeout1 **IS** reset
		3	**IF** S1 **IS** high **AND** timeout1 **IS** reached **THEN** Message1 IS set
		4	**IF** S1 **IS** low **AND** S2 **IS** high **THEN** Message1 **IS** reset
3	when (sensor S1 is high for 2 minutes or sensor S2 is high for 10 minutes) and (sensor S3 is low or sensor S4 is high for 30 seconds) then turn actuator A on and send this message: "This is a test message" to bhafi014@uottawa.ca	1	**IF** S1 **IS** high **THEN** Timeout1 **IS** start
		2	**IF** S1 **IS** low **THEN** Timeout1 **IS** reset
		3	**IF** S2 **IS** high **THEN** Timeout2 **IS** start
		4	**IF** S2 **IS** low **THEN** Timeout2 **IS** reset
		5	**IF** S4 **IS** high **THEN** Timeout3 **IS** start
		6	**IF** S4 **IS** low **THEN** Timeout3 **IS** reset
		7	**IF** (timeout1 **IS** reached **OR** timeout2 **IS** reached) **AND** S3 **IS** low **AND** timeout3 **IS** reached **THEN** A **IS** on **ALSO** Message1 **IS** set
		8	**IF** S1 **IS** low **AND** S2 **IS** low **AND** S3 **IS** high **AND** S4 **IS** low **THEN** A **IS** off **ALSO** Message1 **IS** reset

4.3.7 Controller

The controller component interacts with the SOs by sending commands to their actuators in response to sensory inputs it received, user commands, or passage of time. As mentioned previously, SOs automated control definition in SITE is based on the Fuzzy logic theory.

Figure 4.11 shows the Controller component, which is based on the Fuzzy logic. In this figure, fuzzification is the process of transforming crisp values into Fuzzy values. Since the inputs are captured in scalar forms, they need to be first converted into a format that the Fuzzy Inference Engine can comprehend. This is normally done by assigning each input a membership function that is used to associate a grade with each linguistic term. On the contrary, Defuzzification is the process of converting Fuzzy values into crisp values using membership functions analogous to the ones used in the Fuzzification process [79, 80].

The Fuzzy Inference Engine accomplishes the task of mapping the inputs to the outputs using the Fuzzy rules. Therefore, this component receives raw data from the SOs' sensors, fuzzifies it using the associated membership functions, generates the Fuzzy output by the Fussy Inference Engine using the Fuzzy rules stored in the FD, and defuzzifies the output

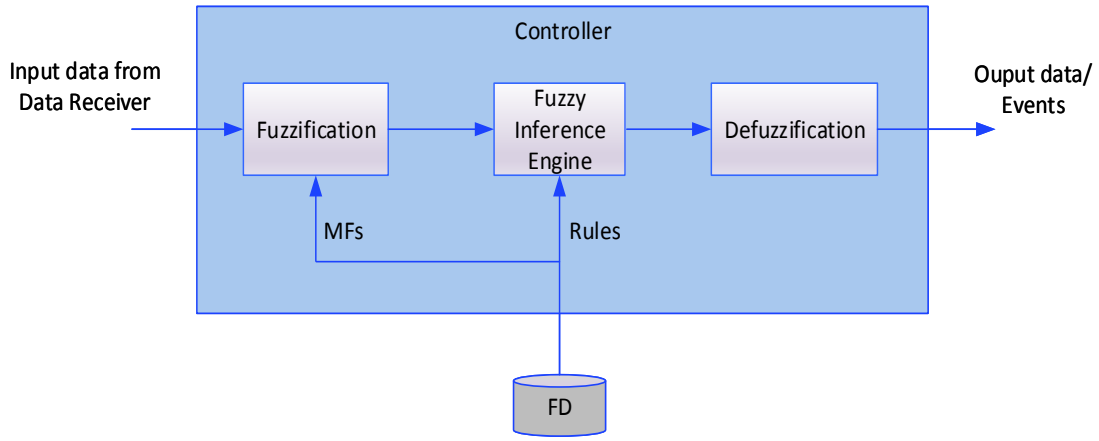


Figure 4.11 The controller module

into crisp results. The crisp results are sent to the associated actuator through the Data Transmitter. In addition to traditional actuators such as vibrotactile or electric switches, SITE supports soft actuators in the form of message generators. Hence, these soft actuators can send warnings or informative messages to the user in the form of email or SMS.

Figure 4.12 illustrates the behavior of the Controller using a Unified Modeling Language (UML) activity diagram. This component loads the Fuzzy rules from the FD and requests real-time sensor's measurement data from the SOs. Once it receives the requested data, it processes it and generates relevant actuation commands whenever necessary (per the Fuzzy rules). The Data Transmitter component transmits these commands to the assigned SOs.

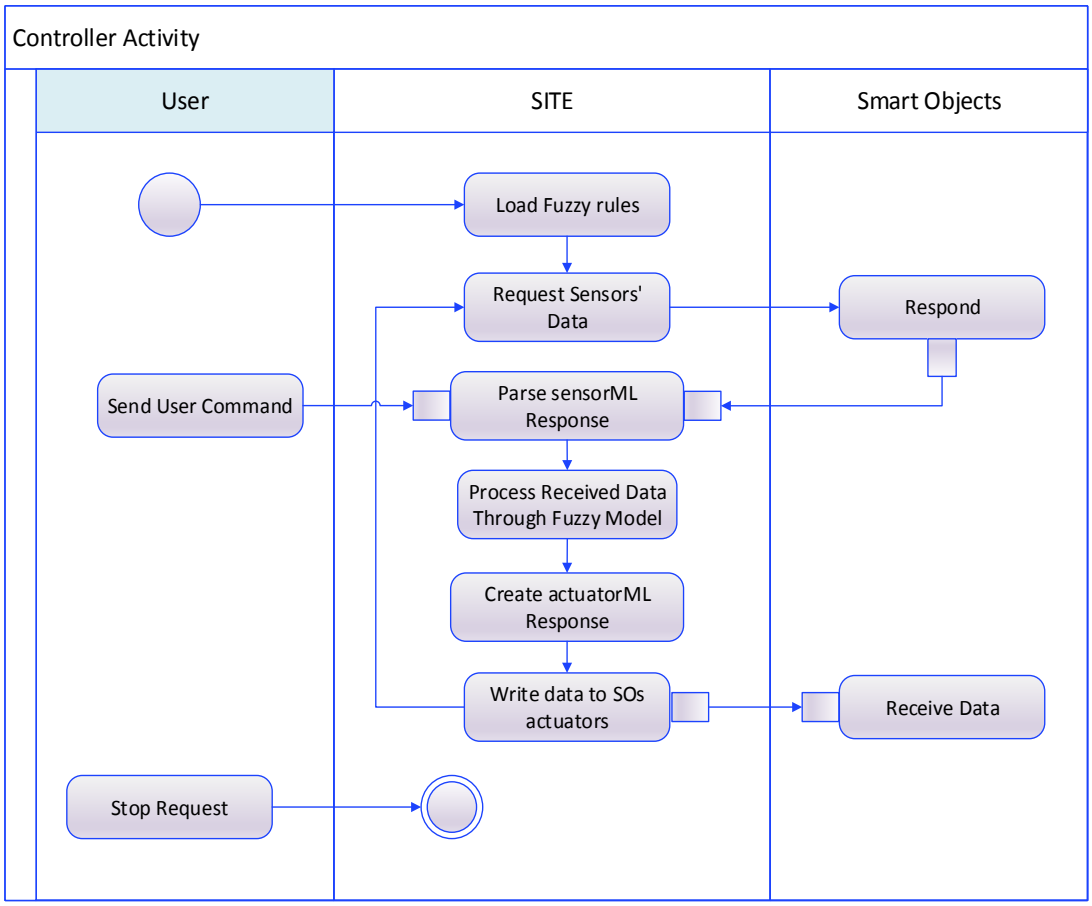


Figure 4.12 The controller activity diagram

4.3.8 Visualizer

As previously explained in the communication protocol between the CVC and computer devices (section 4.2.1), this component is responsible for sending a smart home summary message to the user, through an authorized computing device such as mobile phone, tablet, or laptop, describing the status of the smart home environment. This component sends a read request to the SOs' selected by the user, through the Data Transmitter, and displays their real-time sensor(s) information and measurements.

4.3.9 Data Transmitter and Receiver

These components are responsible for exchanging information between the SITE CVC, user, and SOs. The information exchanged is packaged into SensorML and ActuatorML messages. Note that for simplicity, we package the user command into SensorML messages where the value field is populated with the command ID. SITE CVC sends three types of requests to the SOs: read SO(s) information, read sensor(s) measurement, and send actuation signal(s). Once data is received from the SOs, it is forwarded by the Data Receiver to the DID.

4.3.10 Databases

The SITE CVC uses three databases to store relevant data about the smart objects, sensor measurements, and controller rules. These databases are detailed below:

- Static Information Database (SID): The SID stores information regarding the registration of the SOs, in particular, the SO names and IP addresses. It also stores the MAC addresses of all authorized computing devices that are allowed to communicate with the CVC.

- Dynamic Information Database (DID): This database stores the dynamic information received from the SO through the data receiver. This information includes the number and type of each SO's sensors and actuators. It also includes the real-time raw data measurements of the SO's sensors.
- Fuzzy Database (FD): This database is responsible for storing the generated Fuzzy membership functions and rules as a Fuzzy file during the configuration phase. The Controller (section 4.3.7) retrieves this information in order to fuzzify the sensors' measured raw data.

4.4 User-CVC-SOs Interaction

Figure 4.13 illustrates the overall behavior of SITE CVC and Figure 4.14 shows a UML sequence diagram that describes the interaction between the user and the SITE system. The user starts by registering the available SOs by invoking registorSOs( ) on the *SO Registrar*. The registrar stores SOs' information (names and IP address) by calling storeSOInfo( ) on the *SID* database. The *SO Registrar* displays this information to the user through the displaySOInfo( ) call. Next, the user calls selectSOs( ) on the *SO Selector* to select which SOs present in the environment will be controlled by the CVC. In turn, the *SO Selector* sends a request to every selected SO through the *Transmitter* to obtain dynamic information from each sensor. The dynamic information includes the number and type of transducers attached to each SO. The dynamic Information is received by the *Receiver* and stored in the *DID* through the storeDynamicInfo( ) call.

To visualize the sensors' measurements for each SO, the user sends the visualizeMeasurements( ) request to the *Visualizer*. The *Visualizer* passes the request to

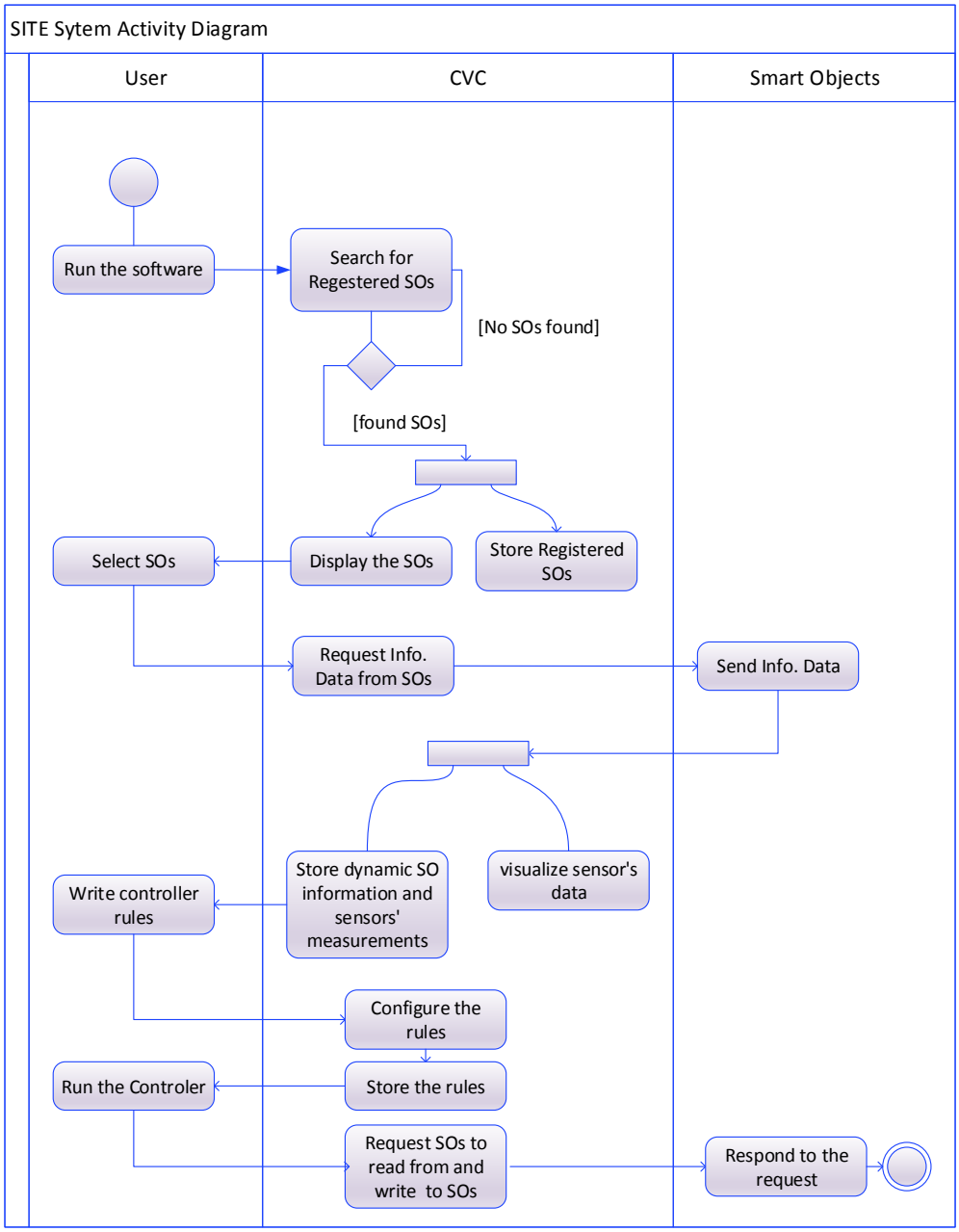


Figure 4.13 The SITE CVC behavior diagram

each SO through the *Transmitter*. Once the measurements are received by the *Receiver*, they are passed to the *Visualizer* to be displayed to the user. The visualization process continues until it is interrupted by the user.

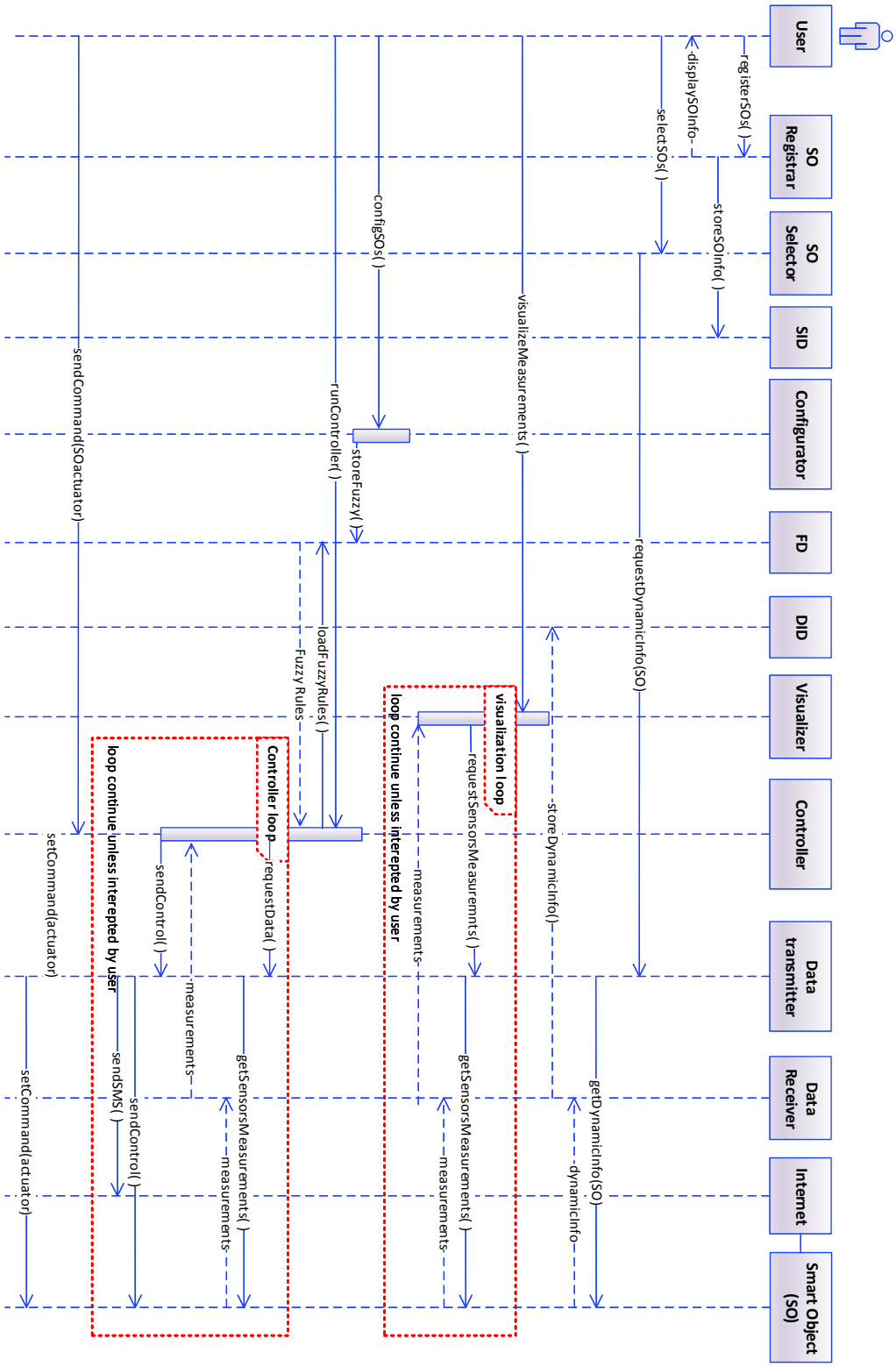


Figure 4.14 A UML sequence diagram for the interaction between the user and the SITE system

The user configures the controller by invoking ConfigSOs( ) on the *Configurator*. The *Configurator*, in turn, generates Fuzzy membership functions and rules as previously described in section 4.3.4, and stores them in the *FD* through the storeFuzzy( ) call.

To run the SITE system, the user calls runController( ) on the *Controller*. The *Controller* loads the Fuzzy rules stored in the FD through the loadFuzzy( ) request, demands sensors' measurements from the SOs through the *Transmitter* and *Receiver*, and sends control signals to SOs actuators through sendControl( ) or sends an email or SMS message to the user through the cloud by calling sendSMS( ). This processing loops continuesly until it is interrupted by the user.

4.5 CVC User Interface

In the previous section, we focused on the design and behavior of the SITE system. In this section, we illustrate the principle user interface components of the SITE CVC. The user interface is developed with the help of the LabVIEW Programming environment. Figure 4.15 depicts the flow chart of the SITE CVC user interface. When the user runs the program, the SO Registrar lists all the registered SOs in the "Registered Objects" field. The user can select the wanted SOs and add them to "Selected Objects" field. Once the SOs are selected, SITE CVC displays relevant information about them in the "Objects' Dynamic Information" field (see the main window in Figure 4.16).

To monitor real-time sensors' measurements for the selected SOs and instantly control actuators, the user can run the visualizer by clicking on the "Visualizer" button. By this button, the SITE CVC automatically generates a GUI based on the number and type of

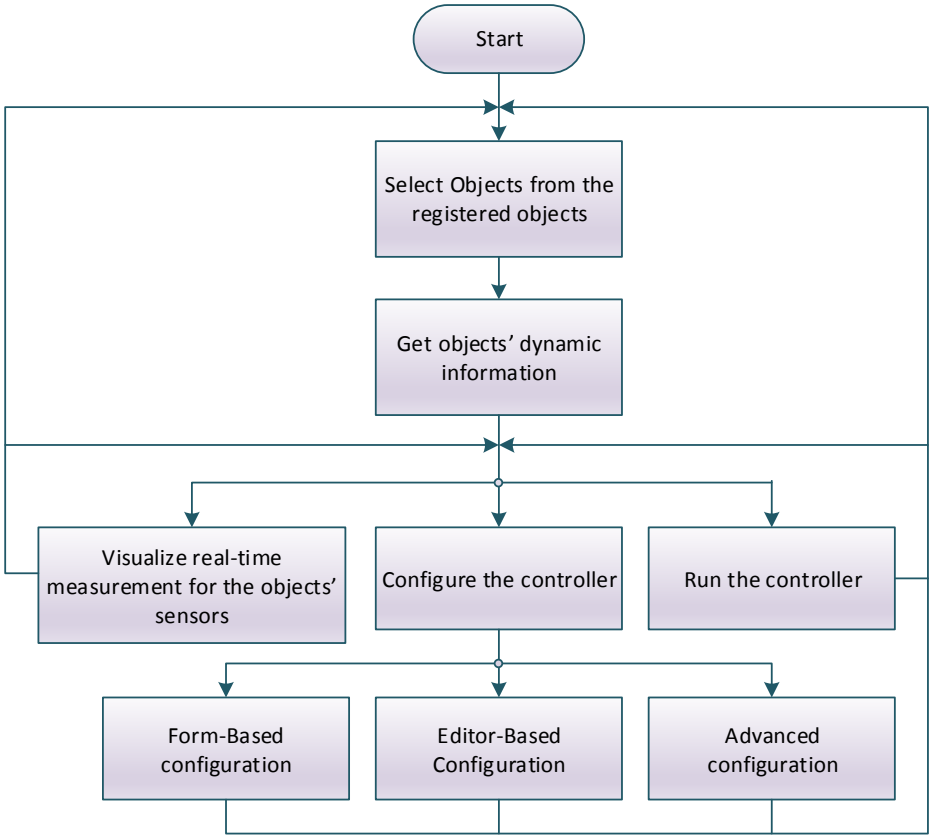


Figure 4.15 Flowchart of the CVC user interface

sensors and actuators of the selected SO. For instants, the automatically generated GUI depicted in Figure 4.17 is for a SO that has two sensors and two actuators. The upper two windows show the sensors' real-time measurements. The type of the displayed measurement meter is specified in the SensorML messages received from the SO. The lower two windows are the actuator windows. By these windows, the user can manually control these actuators. Similar to the sensors, the way of controlling the actuators is based on the actuator type. For example, two types of user commands can be used to control the light actuator shown in the left window: ON/OFF and Dimmer. In the right down window, the user can directly control the vibrotactile actuator of the SO by setting the vibration

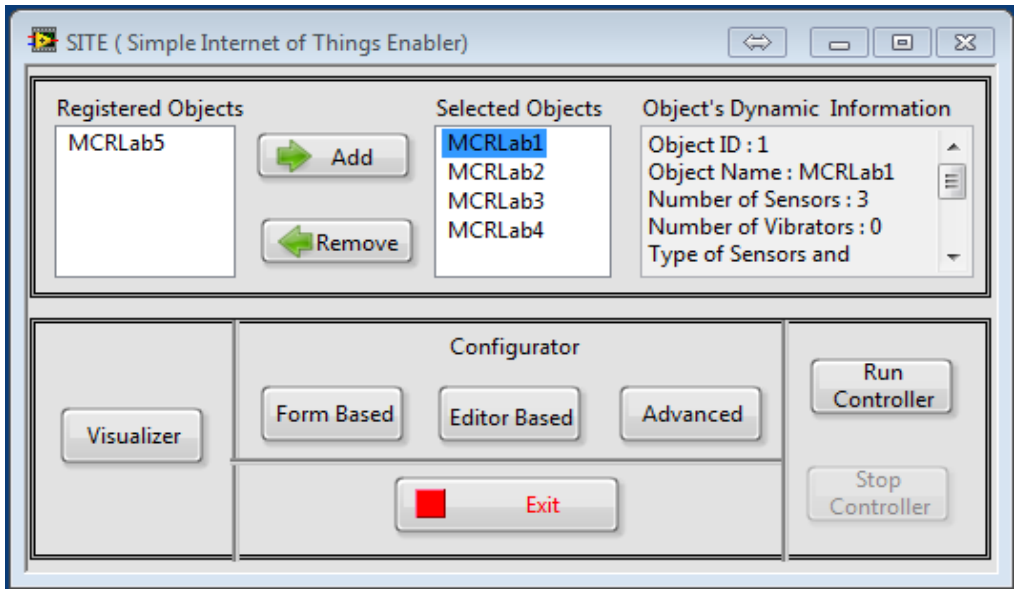


Figure 4.16 CVC main window

intensity, period of vibration and how often (frequency) it vibrates. The actuators can also be controlled automatically by setting up rules using the configurator shown in Figure 4.18.

The main window in Figure 4.16 contains buttons that correspond to three "Configurator" modes: Form-Based, Editor-Based and Advanced. The Form-Based and Editor-Based configurators (shown in Figures 4.18(a) and (b)) are SCL based. The Form-Based mode is designed for novice users and allows the specification of rules using a simplified Graphical User Interface (GUI). The user simply builds SCL rules by completing a form. In the Editor-Based mode, the user enters SCL textually.

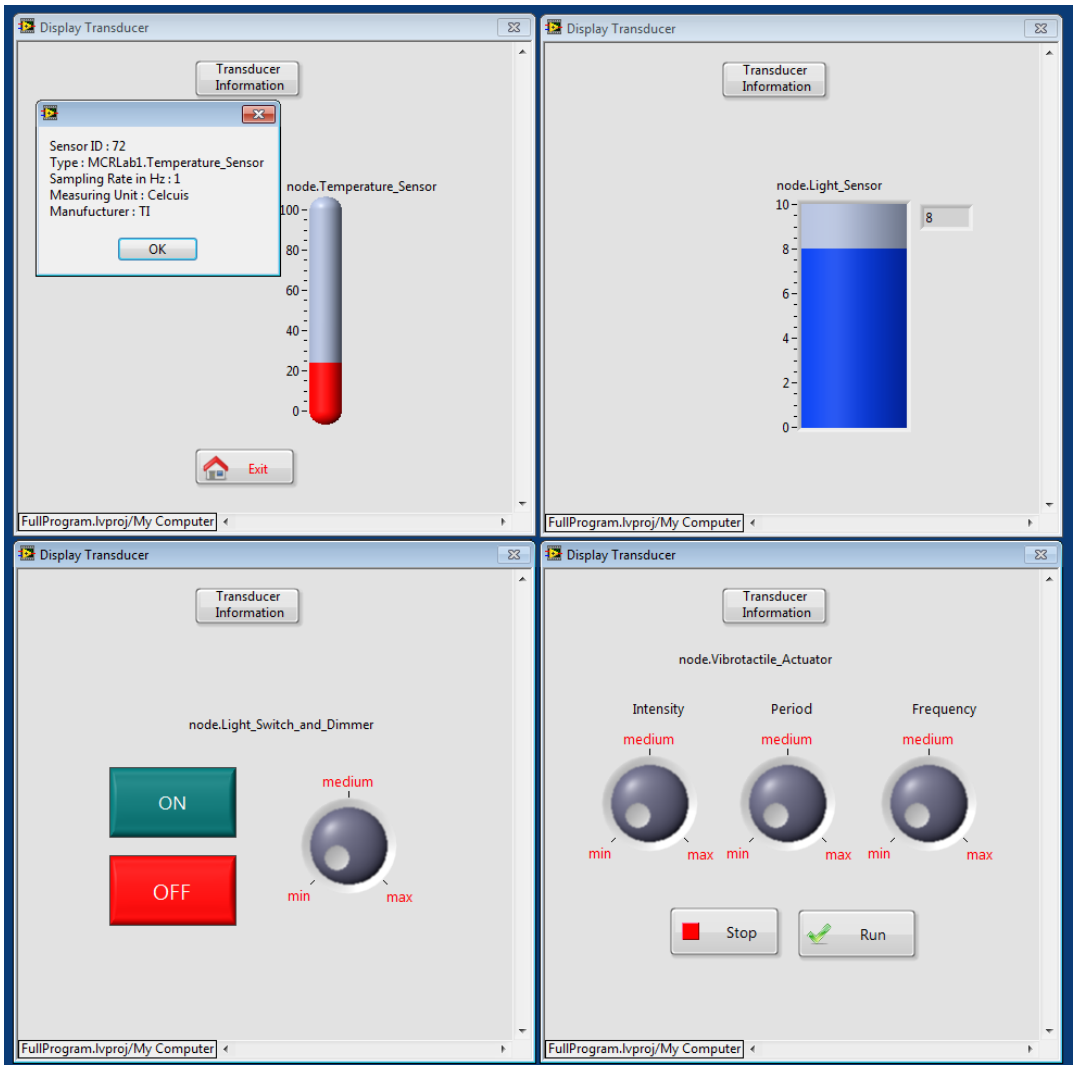
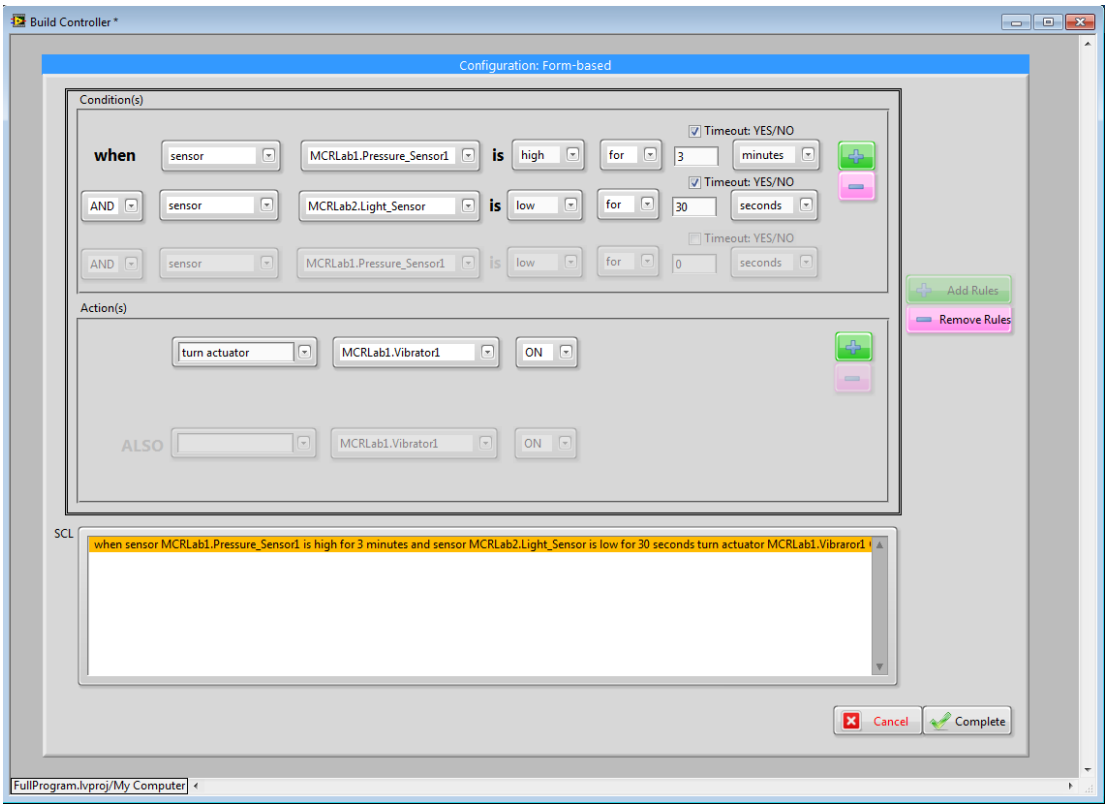


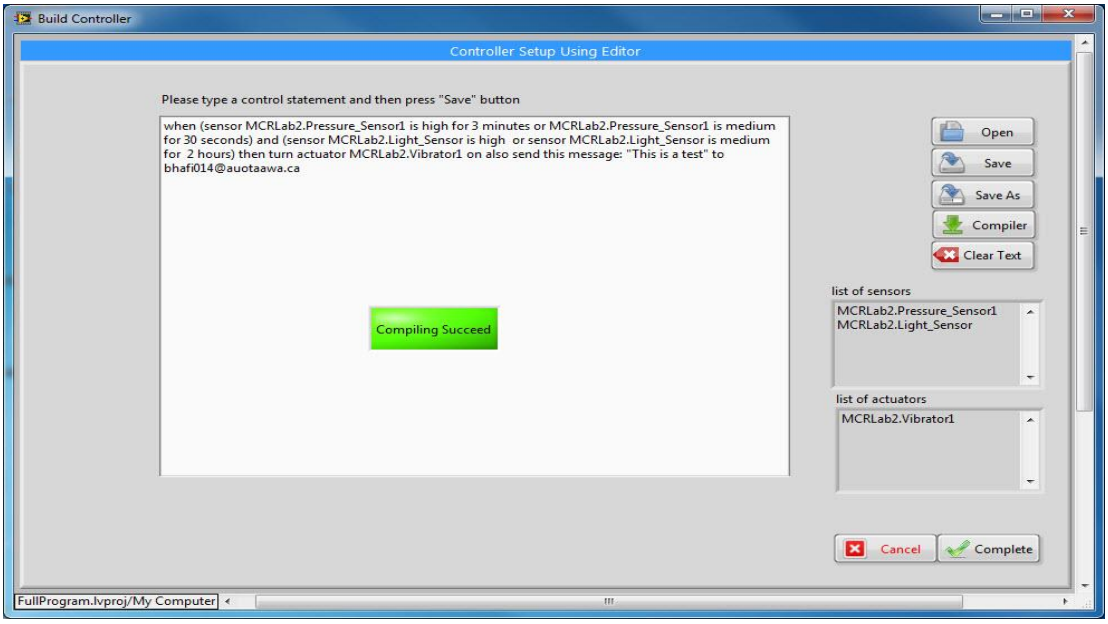
Figure 4.17 The visualizer window showing the real-time measurements for two sensors in addition to two actuators, which can be controlled manually by a user command

The Advanced configuration mode is the most expressive as it allows the user to directly specify Fuzzy rules and membership functions. Hence, the user is not restricted to the level,

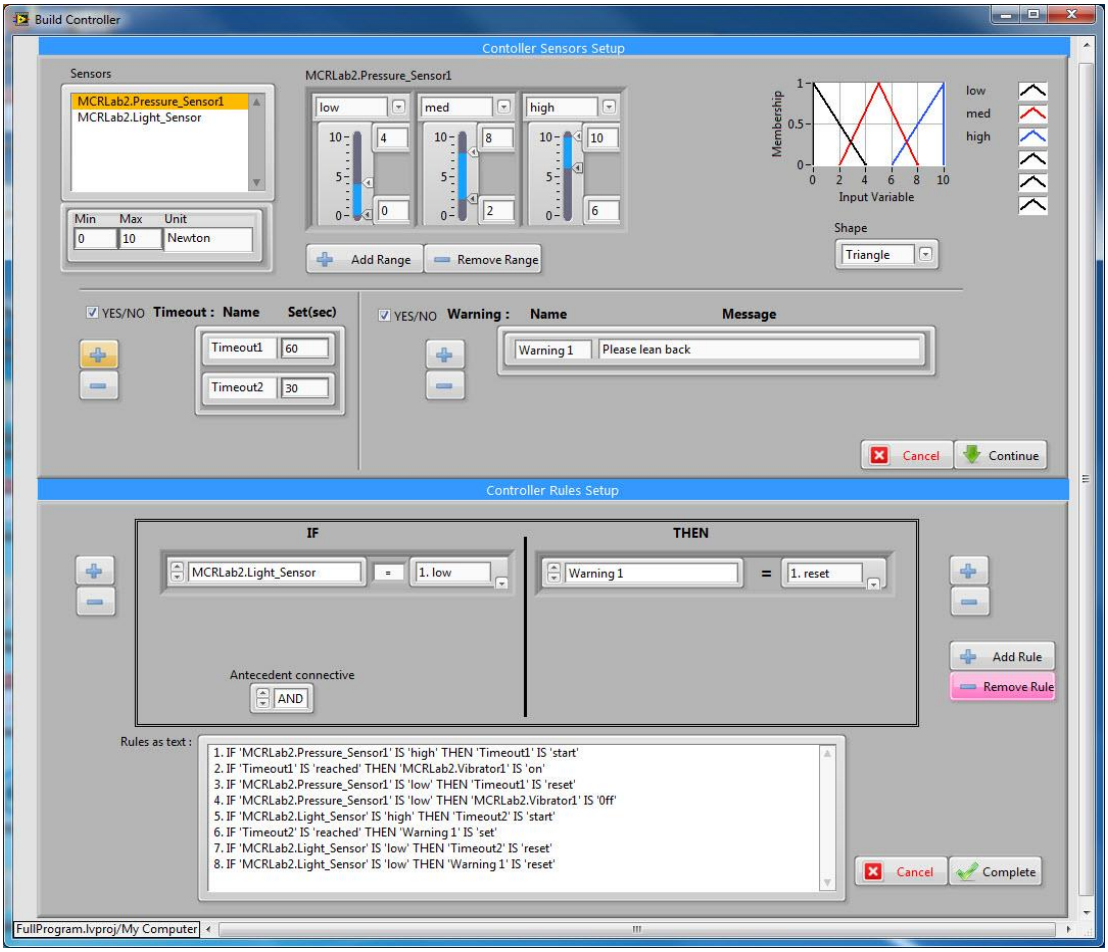
range, and shape of the membership functions. Figure 18(c) illustrates a screenshot of this configuration mode.



(a)



(b)



(c)

Figure 4.18 Configuration modes: (a) Form-Based, (b) Editor-Based, and (c) Advanced

4.6 Summary

In this chapter, we have detailed the design and user interface for the proposed SITE CVC system. In terms of the design, we described various components that compose the system. We elaborated on how the SITE CVC interacts with SOs and the user by modeling the entire workflow through UML activity and sequence diagrams. We introduced a rule-based language (SCL) that allows end-users to define the interaction behavior between the CVC

and SOs. We detailed two algorithms that we developed to translate the SCL language into Fuzzy logic rules and membership functions. A graphical user interface for the SITE CVC was implemented to allow users to monitor, configure and control SOs. In terms of configuration, three configuration modes were designed, two for novice and advanced users (Non-IT and IT) and one exclusively for advanced (IT) users.

Chapter 5 **Evaluation**

As previously discussed in Section 2.5, energy consumption is considered as one of the major challenges in wireless transducer network design [42-44]. Therefore, in this chapter, we compare the energy consumption of the GPTN to a representative conventional transducer network. Additionally, we assess the usability of the SITE system to investigate its suitability for end-user development.

5.1 GPTN Energy Consumption

The proposed GPTN power consumption is evaluated in this section by measuring the power consumption of each node of the 6 nodes previously explained in the house example in section 3.3.2 and Figure 3.5. Figure 5.1 shows a visualization of the data measured by the 6 wireless nodes in the home setup previously described during a period of 24 hours. The white color represents the minimum measured value and the dark represents the maximum. For example, the first bar represents the data for the CO level in the kitchen (SO 1). This level was high from 10 am to 12 pm, from 3 pm to 5 pm, and from 8 pm to 9 pm (during which the kitchen was used for cooking).

In order to prove that our proposed framework consumes less energy compared to "conventional" wireless transducer networks. A conventional wireless network is a set of nodes connected through a wireless channel. Each node is composed of a single sensor,

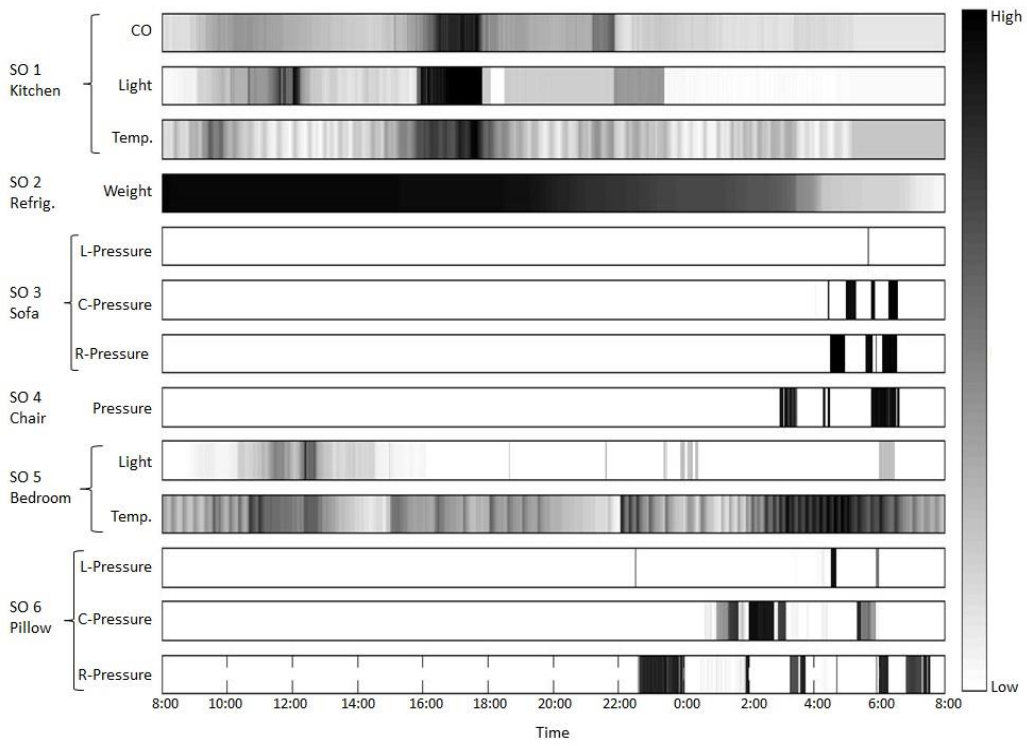


Figure 5.1 Smart Objects' sensors measurement

microcontroller, and wireless transceiver. Such networks have been described in numerous previous works including [81-85]; we have set up a dedicated experiment over a period of 24 hours. During the experiment, we ran in parallel two transducer networks: the one described above using our proposed GPTN and another conventional network composed of the same transducers. To measure the energy consumption, the node's supply voltage and current were measured. The current sensor (TI-INA169) [86] was used to measure the node's drawn current. Figure 5.2 shows the 24-hour energy consumption by each proposed node as compared to conventional transducer nodes that are typically used in wireless sensor networks. For example, SO1 is compared with the combination of three conventional transducer nodes. For SO4 (the smart chair), however, there is no energy saving compared to the conventional network of nodes, since SO4 is composed of only one pressure sensor and one actuator, and the effect of the actuator was neglected as it is run

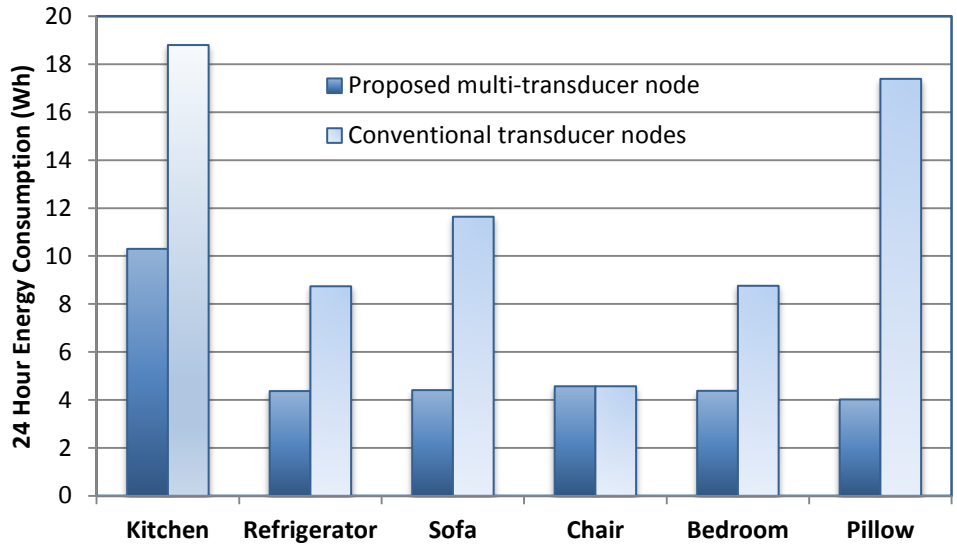


Figure 5.2 Energy consumption of each proposed transducer node compared with conventional transducer nodes

for 30 seconds every 30 minutes to buzz the sitter when the pressure sensor continuously reads values greater than zero for a period of 30 minutes. We can see from the figure, the more transducers are included in the node, the more energy we save and besides the fewer nodes can be involved in the network.

5.2 SITE Usability Study

In this section, we present an empirical study that assesses the usability of the SITE system. We propose the following hypothesis:

"Given that users receive individually a 15 minutes video tutorial about using the SITE system, they will be able to successfully complete the tasks of:

- *Building SOs using the GPTN framework;*
- *Visualizing the data generated by the SOs; and*

- *Defining rules to control SOs through the SITE CVC."*

The experimental procedure is composed of five main phases as shown in Figure 5.3 (see the details of the experimental protocol in Appendix B). Each of these phases is explained in section 5.2.2.

5.2.1 Participants

We conducted a user trial involving 20 adults (10 males and 10 females) with a mean age of (29.45 ± 5.3) years. All participants signed an informed consent form.

5.2.2 Procedure

5.2.2.1 Background Questions

The subjects were first asked four questions centered on their experience in information technology. Table 5.1 presents the distribution of their answers. Based on the answers, the

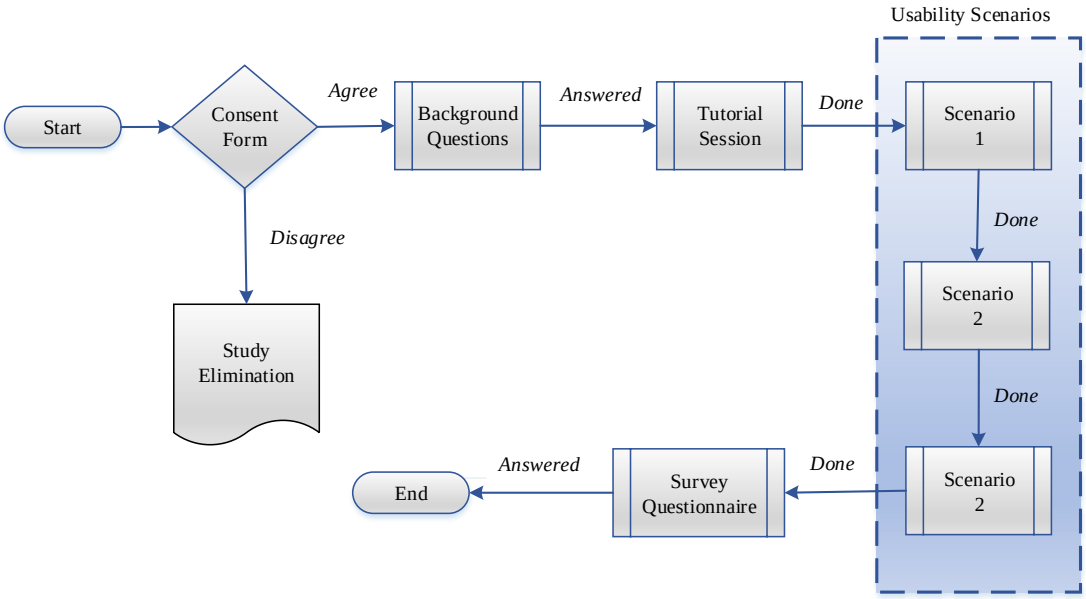


Figure 5.3 Empirical study phases

participants were divided into two broad user groups: IT (5 males and 5 females), and non-IT (5 males and 5 females). A description of the user groups is shown in Table 5.2.

Table 5.1 Background Question Answers Distribution

Background Question	Answer option	Response Count	% Response
Q1: Education	High School	6	30%
	Bachelor	6	30%
	Masters	8	40%
	Doctorate	0	0%
Q2: General expertise with computers	None	0	0%
	Basic	6	30%
	Intermediate	4	20%
	Excellent	10	50%
Q3: Formal software/ Hardware design training	None	10	50%
	Basic	7	35%
	Intermediate	3	15%
	Excellent	0	0%
Q4: Fuzzy logic knowledge	None	18	90%
	Basic	1	5%
	Intermediate	1	5%
	Excellent	0	0%

Table 5.2 Groups of Participants

Participants group	Common characteristics	Sample size
IT Users	Users that possess an undergraduate degree in a discipline that includes intermediate or advanced courses in software development, hardware development, or both. or have undergone equivalent training	10
Non-IT Users	Users that have not undergone any training in software and hardware development.	10

5.2.2.2 Tutorial

Participants received individually a 15 minutes video tutorial describing how to build a SO using the GPTN and interact with the SITE CVC. The tutorial is designed to teach the subjects by example. Hence, it shows them how to build a smart chair. Also, the subjects were given an instruction manual (See Appendix C) that covers the same material as the video tutorial. They were told that they could reference the manual whenever needed during the evaluation.

5.2.2.3 Usability Scenarios

Participants were directed to build SOs in three selected scenarios. For the first scenario (Figure 5.4), the users were asked to realize a smart chair that vibrates when the user continuously sits for a predefined period. The chair is also equipped with a basic posture monitoring mechanism to ensure that the seated person does not lean forward for a prolonged period. This scenario was covered by the tutorial. Hence, the users were required

Figure 5.4 The smart office chair scenario, the hardware prototype composed of the mainboard, pressure and light sensors, and vibrotactile actuator

to reproduce the steps shown in the tutorial. To build the smart chair, the subjects had to append:

- A pressure sensor on the seat to sense the presence of a seated person
- A light sensor on the back to detect whether the seated person is leaning forward
- A vibrotactile actuator on the back to convey a haptic warning regarding sedentary behavior

For the second scenario (Figure 5.5), the users were asked to realize a smart fridge that sends a notification message in the form of email or SMS when the eggs container is empty. It also activates a buzzer if the fridge door is open for a predefined period. To build the latter SO, the subjects had to append:

- A pressure sensor underneath the eggs container to detect the presence of eggs
- A light sensor inside the fridge to detect when the door opens
- A buzzer inside the fridge to remind users to close the fridge's door

Figure 5.5 The smart fridge scenario, (Left) a pressure sensor located underneath the eggs container. (Right) the mainboard, light sensor, and buzzer are appended on the fridge door

For the third scenario (Figure 5.6), the users were asked to realize a smart living room where a TV set is turned on when the user sits on the couch, a lamp light intensity changes depending on the room's lighting level, and a fan speed changes based on the room's temperature. To build the necessary SOs for this scenario, the subjects had to append:

- A pressure sensor to the couch seat cushion to sense the presence of a seated person

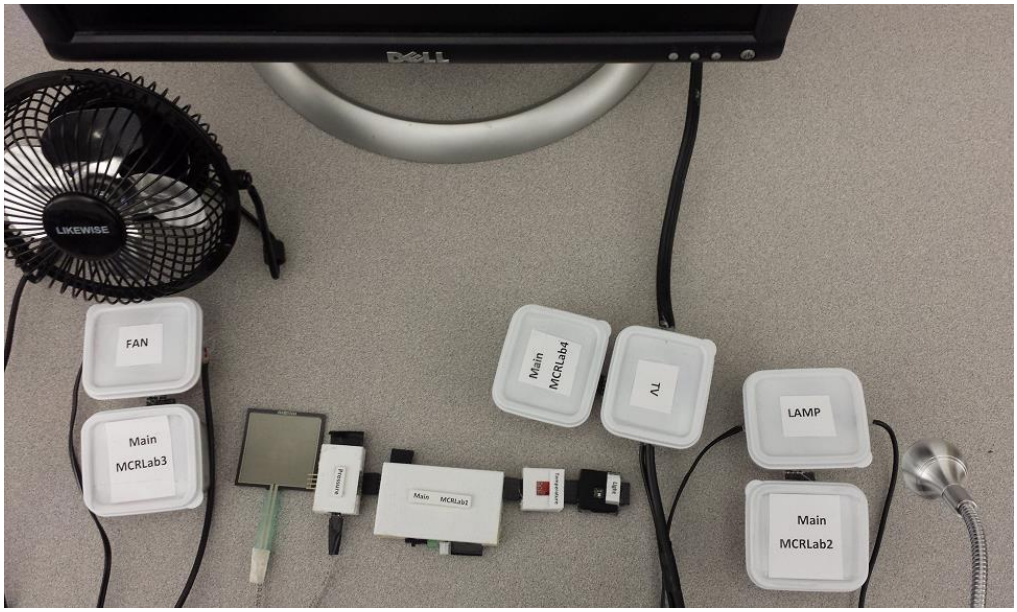


Figure 5.6 The smart living room scenario, composed of four GPTN nodes, one node (MCRLab1) for the sensors (pressure, temperature, and light) and three nodes (MCRLab2, 3, and 4) for the lamp, fan and TV actuators

- A temperature and light sensors inside the room to measure the room's temperature and light intensity
- An On-Off actuator to control the TV power

Table 5.3 The Three Scenarios for the Experimental Study and Their Tasks

Scenario	Task	Description
1 Smart Chair	1	Building the chair SO
	2	Writing SCL rules for prolonged seating detection and user warning
	3	Writing rules for basic posture monitoring and user warning
	4	Running controller and testing the system
2 Smart Fridge	1	Building the fridge SO
	2	Writing SCL rules for eggs absence and user warning
	3	Writing SCL rules for open door detection and user warning
	4	Running controller and testing the system
3 Smart Living Room	1	Building the lamp SO
	2	Building the fan SO
	3	Building the TV SO
	4	Writing SCL rules for controlling lamp's intensity
	5	Writing SCL rules for controlling fan's speed
	6	Writing SCL rules for controlling TV on/off switch
	7	Running controller and testing the system

- Two dimmer actuators for the lamp and fan

Each scenario was divided into multiple tasks as shown in Table 5.3. Subjects were asked to select any configuration mode described previously in section 4.5. They all selected the Form-Based mode. Figures 5.7 (a), (b), and (c) showing subjects working on scenarios 1, 2, and 3 respectively.

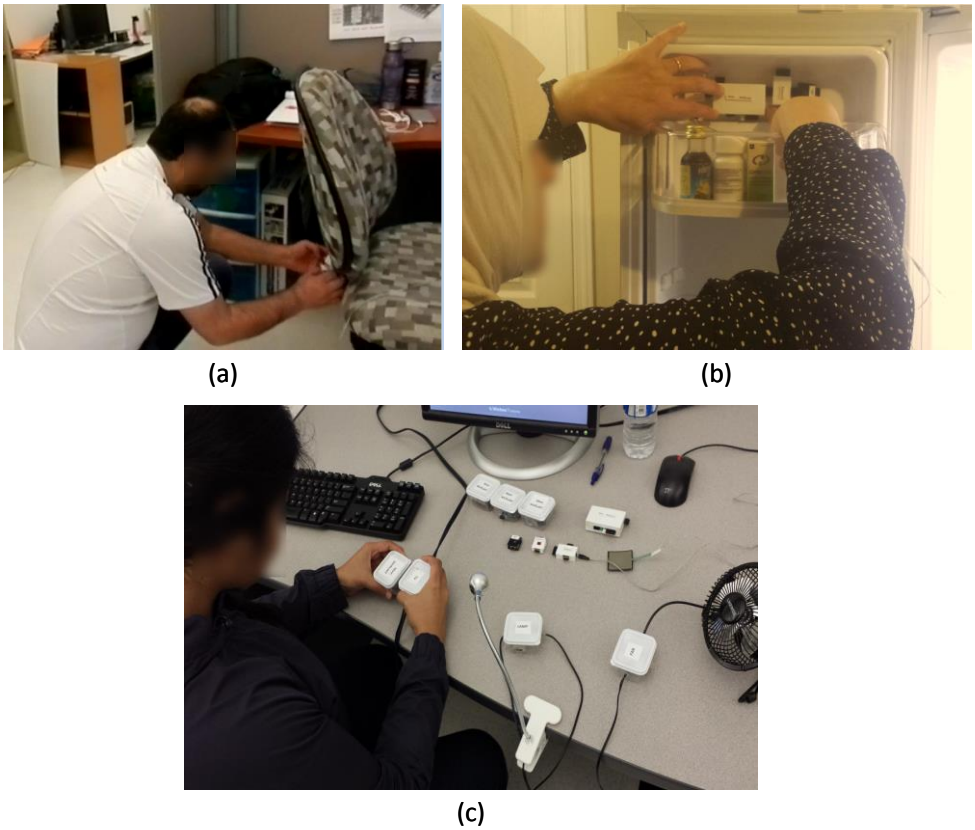


Figure 5.7 Subjects building smart objects for (a) Scenario 1, (b) Scenario 2 and (c) Scenario 3

5.2.3 Evaluation Metrics

We assessed the usability of the proposed system using the metrics presented in the following sections.

5.2.3.1 Error Score

The error score corresponds to the severity of mistakes committed by a subject during task performance [87, 88]. Each task is given an error score between 0 and 1 as follows:

- Very High Severity (1): Errors preempted task completion.
- High Severity (0.75): Errors led to significant difficulties in task completion.
- Medium Severity (0.5): Errors required substantive remedial actions.
- Low Severity (0.25): Errors required minor remedial actions.
- No Error (0): Task was completed without error.

Therefore, if all the participants completed the tasks without error, the sum of error score would be 0. In contrast, if all of them fail to complete all the tasks, the sum of error score would be 300 (20 participants x 15 tasks).

5.2.3.2 Efficiency

The ISO-9241 standard defines the efficiency as: *"resources spent by the user in order to ensure accurate and complete achievement of the goal"* [89]. Hence, in software and information system, the resource is considered as the time spent by the user to accurately achieve the goal. This time-based efficiency can be calculated using the following equation [90]:

$$TimeBasedEffeciency = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N}\frac{n_{ij}}{t_{ij}}}{NR} \quad \textbf{(5.1)}$$

Where:

N = Number of tasks

R = 10 (Number of participants per group)

n_{ij} = The result of task i by participant j (1 if the task is completed successfully and 0 if not)

t_{ij} = The time elapsed for completing task i by participant j.

5.2.3.3 Survey

In addition to the quantifiable metrics, a survey was used to measure the subjects' satisfaction level. A total of 11 questions, 7 Likert scale and 4 open-ended, were answered by the subjects at the end of the evaluation. These questions, along with the subjects' answers are provided in the next section.

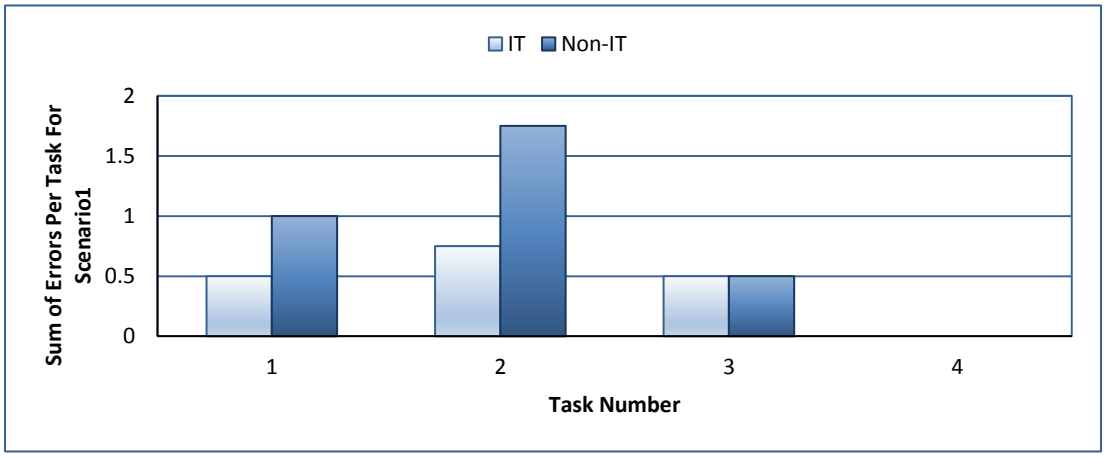
5.2.4 Results

Table 5.4 presents the sum of error score and the % error score for the IT users, Non-IT users, and all participants. For scenario 1, the sum of error score and % error score metrics

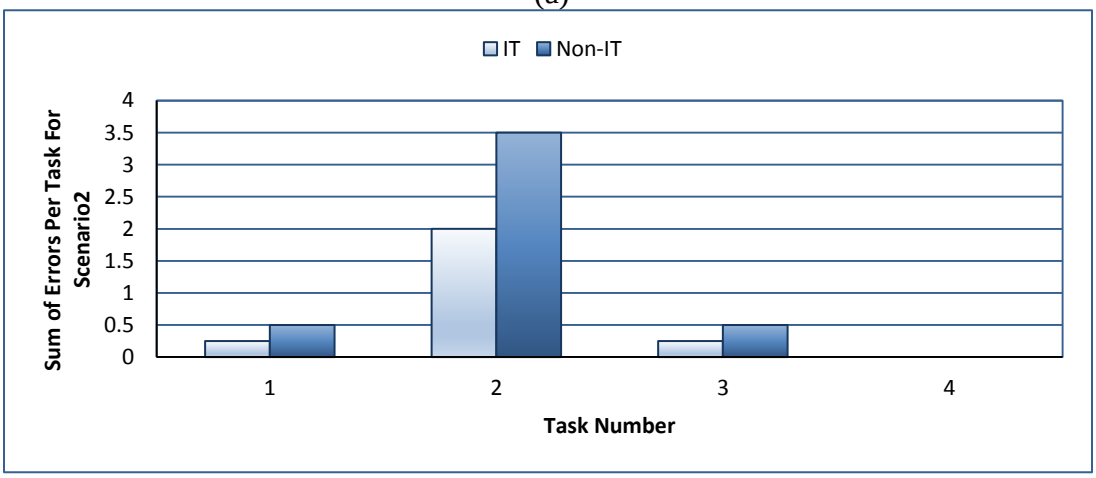
Table 5.4 The Error Scores for the Three Scenarios

Scenario #	Sum of Error Score			% Error Score (sum of errors / (#participants x #tasks))		
	IT	Non-IT	Total	IT	Non-IT	Total
Scenario 1	1.75	3.25	5	2.18%	4.06%	3.125%
Scenario 2	2.5	4.5	7	6.25%	11.25%	8.75%
Scenario 3	8.25	10.75	19	11.78%	15.35%	13.57%

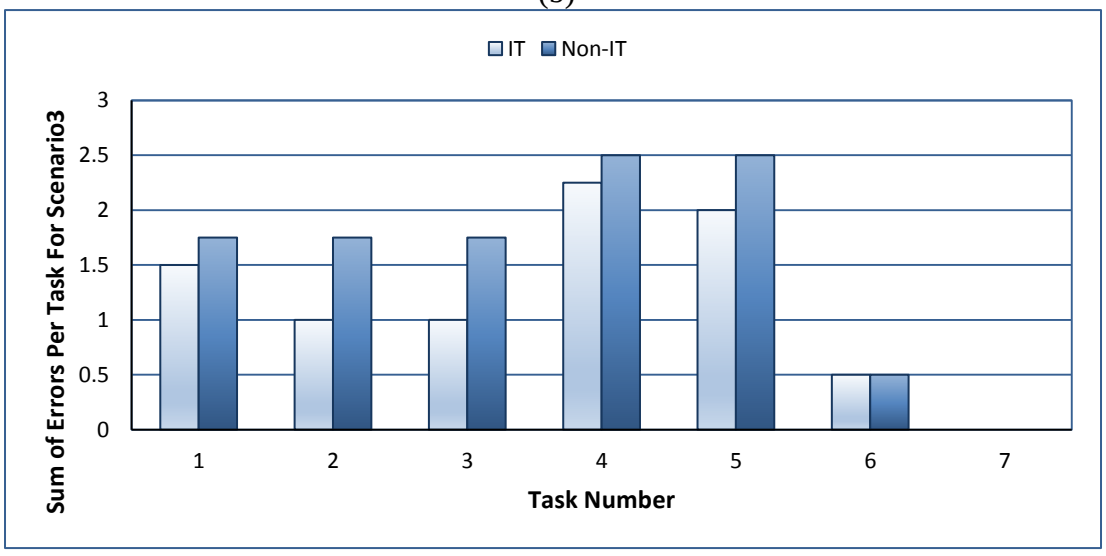
are very comparable across both user groups. A 1-tailed Mann-Whitney test indicates that the sum of error score for scenario 1 is greater for Non-IT users (3.25) compared to IT users (1.75), $U = 40.5$, $p = 0.481$. The difference between both groups is not statistically significant. There is a slight increase of less than 2% in the % error score for Non-IT users. For scenario 2, the error score of the Non-IT users (4.5) is almost twice as high as that of the IT users (2.5). However, it is still a somewhat inconsequential score, considering that the maximum possible sum of error score is 40 (10 participants × 4 tasks). A 1-tailed Mann-Whitney test reveals that the error score difference between both groups for scenario 2 is statistically insignificant, $U = 33.0$, $p = 0.218$. The error score for Non-IT users in scenario 3 (10.75) is also slightly higher than that of IT users (8.25) with a 4% increase in % error score. The maximum possible error score is 70 (10 participants × 7 tasks). The 1-tailed Mann-Whitney test reveals that the error score difference between both groups is statistically insignificant, $U = 34.0$, $p = 0.247$. Note that the Mann-Whitney non-parametric statistical test of difference was employed for the three scenarios since the normality assumption could not be satisfied. Figure 5.8 shows the sum of error score per task for each scenario.



(a)



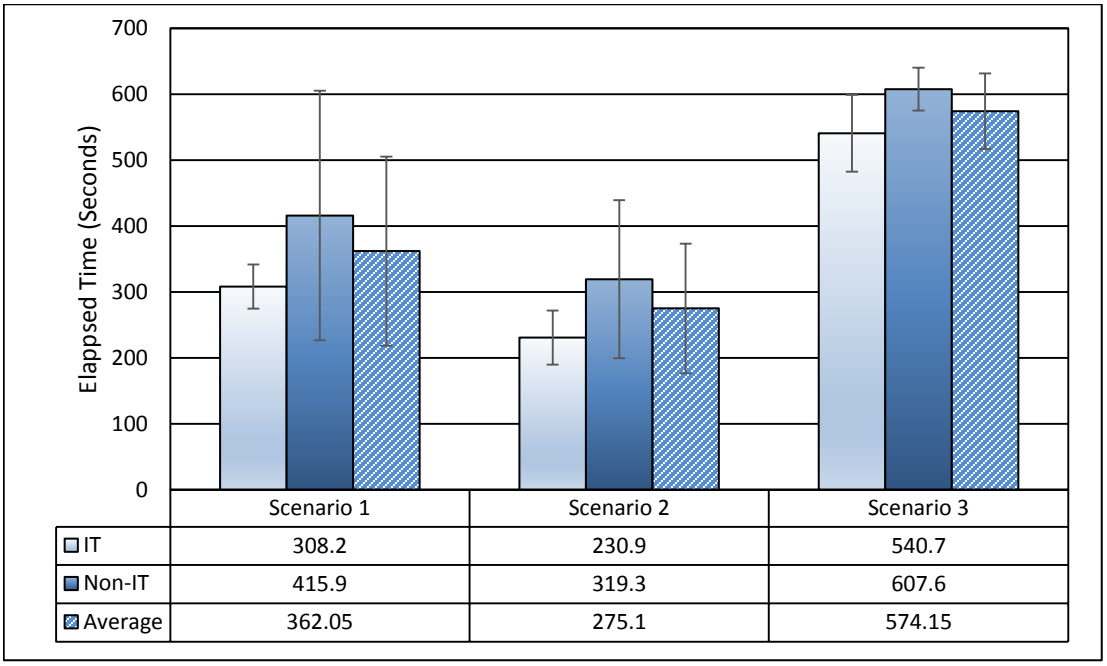
(b)



(c)

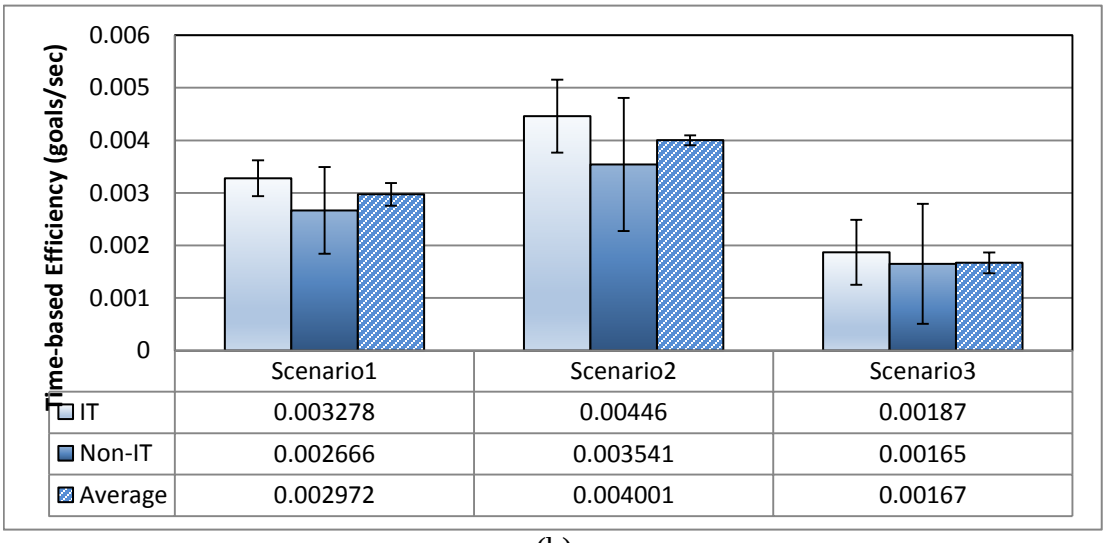
Figure 5.8 Sum of error scores per task for (a) scenario1, (b) scenario2, and (c) sensrio3

Figure 5.9 (a) compares the mean elapsed time for each scenario. The Non-IT users required more time to complete the tasks for all three scenarios compared to the IT users. A 1-tailed Mann-Whitney test indicates that the elapsed time is significantly greater for Non-IT users compared to IT users for scenarios 1 and 3, with U = 16.5, p = 0.009 for



	Scenario 1	Scenario 2	Scenario 3
IT	308.2	230.9	540.7
Non-IT	415.9	319.3	607.6
Average	362.05	275.1	574.15

(a)



	Scenario1	Scenario2	Scenario3
IT	0.003278	0.00446	0.00187
Non-IT	0.002666	0.003541	0.00165
Average	0.002972	0.004001	0.00167

(b)

Figure 5.9 (a) The main elapsed time for the two groups for each scenario. (b) Time-based efficiency for the two groups for each scenario

scenario 1, U = 28.0, p = 0.105 for scenario 2, and U = 11.0, p = 0.002 for scenario 3. This is also reflected in Figure 5.9(b) that shows that the calculated time-based efficiency for Non-IT users is below that of IT users by 18.6% for scenario 1, 25.9% for scenario 2, and 13.3% for scenario 3. Note that the Mann-Whitney non-parametric statistical test of difference was employed since the normality assumption could not be satisfied. Table 5.5 summarizes the results found from Mann-Whitney test for both error score and elapsed time.

Table 5.5 Results From 1-Tailed Mann-Whitney Test

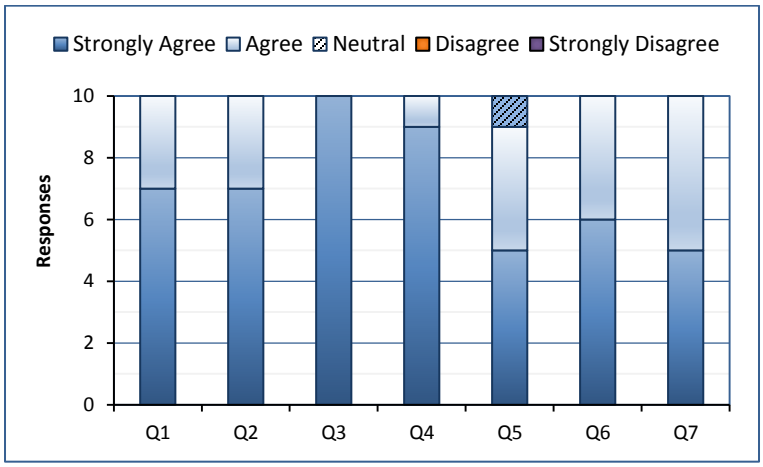
Metric	Scenario #	*U* value	*p* value	Statistically significant?
Error score	Scenario 1	40.5	0.481	no
	Scenario 2	33.0	0.218	no
	Scenario 3	34.0	0.247	no
Elapsed time	Scenario 1	16.5	0.009	yes
	Scenario 2	28.0	0.105	no
	Scenario 3	11.0	0.002	yes

In terms of participants' satisfaction, we present the results of the survey conducted after test completion. The first 7 questions were Likert-scale based and prompted participants to specify their opinion on various interaction aspects by selecting one of 5 options. Table 5.6 lists these questions. Figure 5.10 graphically summarizes the percentage of the IT and Non-IT participants' satisfaction associated with them. Table 5.6 also lists the level of agreement

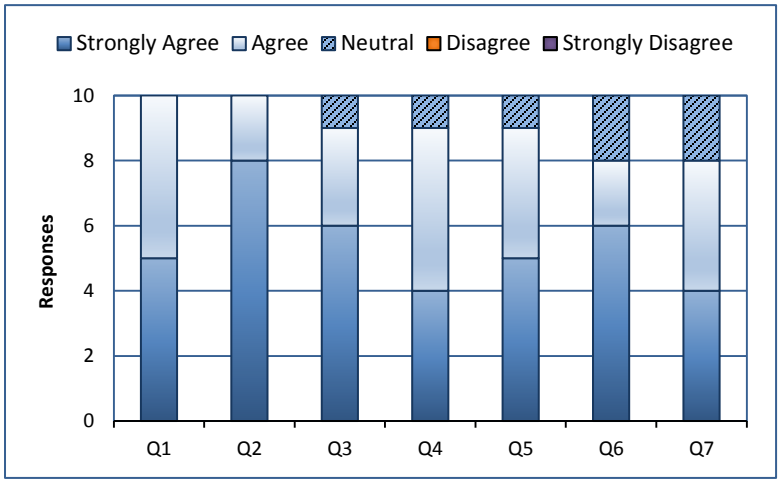
for a question corresponds to the median value of its responses. The results show that there is an overall very high level of agreement with the positive statement of each question.

Table 5.6 Participants' Level of Agreement Results for the Likert-Scale Questions

Number	Likert-scale question	Level of agreement	
		IT	Non-IT
Q1	The blocks were easy to connect together	4.7	4.5
Q2	The setup of the controller on the computer program was easy to understand	4.7	4.8
Q3	The rules of the controller on the computer program were easy to write	5.0	4.5
Q4	The computer program was easy to use	4.9	4.3
Q5	The blocks that you had to connect together for the second and third scenarios were easy to build	4.4	4.4
Q6	The rules of the controller on the computer program for the second and third scenarios were easy to write	4.5	4.4
Q7	After these tutorials, I can build several objects without needing technical help	4.5	4.2



(a)



(b)

Figure 5.10 Answers distribution over satisfaction questionnaire for (a) IT and (b) Non-IT Participants

Participants were also asked four open-ended questions. These questions are listed in Table 5.7. When answering these questions, most of the participants gave positive responses. Table 5.8 depicts a sample of the answers to these questions. Most of the participants were repeating the word "simple" in answering question 8. The table also lists selected responses for what participants disliked about the SITE system in answering question 9. The third open-ended question was if they have any suggestions in order to improve the SITE software system. The table shows a sample of their suggestions. The last

Table 5.7 Open Ended Questions

#	Open ended question
8	Please indicate what you like about the computer program
9	Please indicate what you dislike about the setup of the controller in the computer program
10	Please provide any suggestions to improve the SITE system
11	Please provide any feedback about this study itself, so I may improve it

open-ended question asked participants if they have any suggestions to improve the study. Most of the suggestions focused on improvements to the tutorial. There were few suggestions for various services.

Table 5.8 Sample of Participants' Responses to the Open-End Questions

Question #	Feedback #	Feedback
8	1	*"I like the organization of the interface components and the display of sensors information. The way I can set the rules is very simple and organized. I can say, it is clear and very easy to use"*
	2	*"…very helpful in dealing with different situations that require interaction with sensors"*
	3	*"User-friendly GUI. Simple and Straightforward steps"*

	4	*"It is easy to use and friendly, no need for any computer or programming skills"*
	5	*"I like its simplicity and user friendliness"*
	6	*"The program was easy to use; the labels on the program matched the ones in the video, making it easy to follow"*
	7	*"Simple, no extra buttons that could confuse me"*
	8	*"The rules were clear and easy to use"*
9	1	*"Although it works perfectly, it's slow when I click to run the controller"*
	2	*"The buttons can be enhanced by making them larger in size with more interesting icons, like the iPhone buttons"*
10	1	*"Instead of using the words "low" and "high" for the values of the sensors, for example, for the light sensor, use the words "dark" and "bright" or add a description of what "high" and "low" mean with a small example embedded in the software"*
	2	*"I think the Advanced configurator button in the main window should be located somewhere else so that the user doesn't click on it mistakenly and confuse her/himself"*

	3	*"The screen should be adjustable in size, font selection can be added, other languages should be supported, and pictures of sensors instead of labels should be used"*
11	1	*"... this system can be linked directly to the grocery store in the case of Scenario 2"*

5.2.5 Discussion

We elected to permit users to choose the configuration mode of their choice. They all chose the Form-Based mode for rule specification. This is unsurprising given that this mode does not require users to remember SCL syntax or develop Fuzzy logic expertise. All of our subjects were first time users of SITE. We suppose that increased familiarity with the system after prolonged interaction might encourage users to use the Editor-Based mode. This mode might offer a faster method of entering rules into the system given that the user has memorized the syntax after repeated use. The Advanced mode will probably only benefit expert users familiar with Fuzzy logic knowledge. Although this might be a small user group, we developed this mode to allow an interested user to fine-tune control rules more effectively.

Although there were small differences in the performance of IT and Non-IT users in terms of error score, elapsed time, and time-efficiency, all subjects successfully completed the three scenarios. We obtained a statistically significant difference across groups in the elapsed time for scenarios 1 and 3. However, we did not find a statistically significant difference in the error score across groups for the scenarios. Hence, we can deduce that although Non-IT users were spending more time performing the tasks, they were not

committing significantly more errors. IT users had a particular advantage in both scenarios 2 and 3 since subjects were not assisted in the development and configuration of both the smart fridge and smart living room. Although the completion of these scenarios did not require technical skills beyond those that were covered in the tutorial, they involved some design tasks pertaining to the choice of sensors, their placement (subjects were given hints on sensor placement), and the development of SCL rules.

Both types of users were highly satisfied with the interaction with SITE, as evidenced by the results of the survey. These results are encouraging as they reflect the potential desire of Non-IT users to not only benefit from smart home systems but also participate in their development.

The hypothesis stated at the beginning of section 5.2 is validated given that all participants were able to successfully complete the experiment's tasks.

5.2.6 Limitations

We identify limitations related to the size and scope of the study. We list them as follows:

- *A number of participants*: There is an on-going debate about the appropriate sample size in usability studies. Nielsen [91] recommends the recruitment of 20 subjects for quantitative usability studies. We have followed this recommendation.
- *The age range of the participants*: None of the subjects was above 42 years of age in both user groups. Older individuals are potentially less comfortable with technology [92] and hence might attain a lesser level of success, especially for Non-IT users. Further studies are needed to assess the usability of the system with other age groups.

- *The scope of features investigated*: All subjects chose the Form-Based mode for rule specification. This left the other configuration modes untested. Further investigations are required to assess the usability of the other supported modes.

5.3 Summary

Chapter 5 discussed the experimental evaluation of the GPTN framework using the home example of chapter 3. In this test, we compared the energy consumption of the GPTN to a representative conventional transducer network over a period of 24 hours. The results obtained showed that the GPTN achieves considerable energy saving compared to the conventional transducer network. We also assessed the usability of the SITE system, which was conducted with 20 participants (10 IT and 10 Non-IT). Participants were directed to build SOs in three selected scenarios. Different usability metrics were applied along with survey questionnaire, which revealed that all the participants successfully completed the three scenarios and expressed their satisfaction with the system.

Chapter 6 **Conclusions and Future Work**

6.1 Thesis Summary

In this thesis, we proposed the SITE system that interacts with two types of entities: users and SOs. SITE allows the development of a smart environment using the GPTN to create and communicate SOs and SITE CVC to specify the SO control rules.

The proposed GPTN is aimed at reducing the energy consumption of transducer networks through wired clustering. A plug and play mechanism was conceived to cluster any number of transducers together in several possible configurations. The node's master dynamically recognizes newly added/removed transducers and informs the CVC. A prototype smart home was developed by placing GPTN nodes on various home objects. Each GPTN node was composed of one or more transducers depending on the required measurements. We evaluated the energy consumption of the proposed transducer network by comparing it to a conventional wireless transducer network where each node is dedicated for a single transducer. We showed that our proposed transducer network requires significantly less energy to run. For the prototype home setup, our proposed method consumes 46% less energy compared to conventional approach.

The proposed SITE system supports users with varying levels of technology expertise. Hence, three modes of SO control rules specification are provided (Form-Based, Editor-Based and Advanced). The Form- and Editor-Based modes are SCL based and designed for novice and expert users. The Advanced mode allows more refined rule specification and is designed for technology savvy users with basic knowledge of Fuzzy logic.

An Empirical study was conducted to evaluate the usability of the SITE system. Twenty (IT and Non-IT) participants were directed to build SOs and interact with the SITE CVC. Although some differences in performance were observed between the groups, with the IT user group achieving a slightly smaller error score and higher time-efficiency, all users were able to complete the experimental tasks. Furthermore, both user groups expressed their satisfaction with the system. These encouraging results signify that Non-IT users might be interested in taking on the role of smart home system designer in addition to end-user.

6.2 Future Work

As mentioned in section 2.3, some researchers [58, 59] have developed frameworks to describe home appliances using the User Interface Description Language (UIDL). The latter description is entered into a tool that automatically generates a user interface (UI) that allows users to interact with these appliances. As an immediate future project, we will integrate the UIDL with SITE to achieve a complete solution that not only allows users to create their custom SOs but also supports user interaction with inherently smart objects, such as smart appliances and smart electronics (e.g. TV, personal assistant…).

Furthermore, as discussed in the limitations section of the SITE CVC usability study (section 5.2.6), none of the subjects in our empirical study were above 42 years of age in both user groups. Since senior individuals are potentially less comfortable with technology, further studies are needed to assess the usability of the system with older age groups.

To further increase the usability of the SITE system, we can add a fourth configuration mode that supports speech based interaction. Hence, the user can enter or edit SCL rules through speech based commands as opposed to using the GUI based approach.

Bibliography

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks,* vol. 54, no. 15, pp. 2787-2805, 2010.

[2] D. Bandyopadhyay and J. Sen, "Internet of Things: Applications and Challenges in Technology and Standardization," (in en), *Wireless Personal Communications,* vol. 58, no. 1, pp. 49-69, 2011.

[3] S. Li, L. Da Xu, and S. Zhao, "The internet of things: a survey," *Information Systems Frontiers,* vol. 17, no. 2, pp. 243-259, 2015.

[4] A. Whitmore, A. Agarwal, and L. Da Xu, "The Internet of Things—A survey of topics and trends," *Information Systems Frontiers,* vol. 17, no. 2, pp. 261-274, 2015.

[5] I. Bose and R. Pal, "Auto-ID: managing anything, anywhere, anytime in the supply chain," (in en), *Communications of the ACM,* vol. 48, no. 8, pp. 100-106, 2005.

[6] I. Peña-López, "ITU Internet report 2005: the internet of things," 2005.

[7] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal,* vol. 1, no. 1, pp. 22-32, 2014.

[8] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems,* vol. 29, no. 7, pp. 1645-1660, 2013.

[9] G. Fortino, A. Guerrieri, and W. Russo, "Agent-oriented smart objects development," in *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on*, 2012, pp. 907-912.

[10] M. R. Alam, M. B. I. Reaz, and M. A. M. Ali, "A review of smart homes—past, present, and future," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42, no. 6, pp. 1190-1203, 2012.

[11] A. Souza and J. R. A. Amazonas, "A Novel Smart Home Application Using an Internet of Things Middleware," in *Smart Objects, Systems and Technologies (SmartSysTech), Proceedings of 2013 European Conference on*, 2013, pp. 1-7.

[12] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, "Smart objects as building blocks for the internet of things," *Internet Computing, IEEE,* vol. 14, no. 1, pp. 44-51, 2010.

[13] J. Voas, "Building Blocks of the Internet of Things," in *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, 2016, pp. 1-12.

[14] V. D. P. Philippopoulos and C. Georgopoulos, "Smart Homes: a user perspective," *hft.org*, 2004.

[15] J. A. Macías, "Development of end-user-centered EUD software," in *Proceedings of the 13th International Conference on Interacción Persona-Ordenador*, 2012, p. 24.

[16] H. Lieberman, F. Paternò, M. Klann, and V. Wulf, "End-user development: An emerging paradigm," in *End user development*: Springer, 2006, pp. 1-8.

[17] R. Dautriche, C. Lenoir, A. Demeure, C. Gérard, J. Coutaz, and P. Reignier, "End-user-development for smart homes: relevance and challenges," in *Proceedings of the Workshop" EUD for Supporting Sustainability in Maker Communities", 4th International Symposium on End-user Development (IS-EUD)*, 2013, p. 6.

[18] R. Blasco, Á. Marco, R. Casas, D. Cirujano, and R. Picking, "A smart kitchen for ambient assisted living," *Sensors,* vol. 14, no. 1, pp. 1629-1653, 2014.

[19] R. Piyare and M. Tazil, "Bluetooth based home automation system using cell phone," 2011, pp. 192-195.

[20] R. Piyare and S. R. Lee, "Smart Home-Control and Monitoring System Using Smart Phone," in *1st International Conference on Convergence and its Application (ICCA)*, 2013.

[21] M. Yan and H. Shi, "Smart living using Bluetooth-based Android smartphone," *International Journal of Wireless & Mobile Networks*, vol. 5, no. 1, p. 65, 2013.

[22] J. Byun, B. Jeon, J. Noh, Y. Kim, and S. Park, "An intelligent self-adjusting sensor for smart home services based on ZigBee communications," *Consumer Electronics, IEEE Transactions on*, vol. 58, no. 3, pp. 794-802, 2012.

[23] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, "Information Centric Networking in IoT scenarios: The case of a smart home," in *2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 648-653.

[24] S. H. Ahmed and D. Kim, "Named data networking-based smart home," *ICT Express*, vol. 2, no. 3, pp. 130-134, 2016.

[25] S. Folea, D. Bordencea, C. Hotea, and H. Valean, "Smart home automation system using Wi-Fi low power devices," in *Automation Quality and Testing Robotics (AQTR), 2012 IEEE International Conference on*, 2012, pp. 569-574.

[26] M. Sarnovský, P. Kostelník, P. Butka, J. Hreňo, and D. Lacková, "First demonstrator of hydra middleware architecture for building automation," in *Proceedings of the Scientific Conference Znalosti*, 2008.

[27] T. D. Dang, "A Framework for Cloud-Based Smart Home," University of Technology, Sydney, 2017.

[28] Z. Wei, S. Qin, D. Jia, and Y. Yang, "Research and design of cloud architecture for smart home," in *Software Engineering and Service Sciences (ICSESS), 2010 IEEE International Conference on*, 2010, pp. 86-89.

[29] N. D. Lap. (2012). *Cloud Computing Introduction*. Available: http://www.slideshare.net/lapnd/cloud-computing-introduction-15600996

[30] S. Attitalla, V. Choksi, and M. Potdar, "Web and Cloud Based Home Automation Systems: An Overview," 2016.

[31] H. Gu, Y. Diao, W. Liu, and X. Zhang, "The design of smart home platform based on cloud computing," in *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*, 2011, vol. 8, pp. 3919-3922.

[32] A. Meola. (2016, Dec.28,2016). *The roles of cloud computing and fog computing in the Internet of Things revolution*. Available: http://www.businessinsider.com/internet-of-things-cloud-computing-2016-10

[33] N. Dickey, D. Banks, and S. Sukittanon, "Home automation using Cloud Network and mobile devices," in *Southeastcon, 2012 Proceedings of IEEE*, 2012, pp. 1-4.

[34] M. Soliman, T. Abiodun, T. Hamouda, J. Zhou, and C.-H. Lung, "Smart home: Integrating internet of things with web services and cloud computing," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, 2013, vol. 2, pp. 317-320.

[35] D. Bin, R.-C. Chen, and K.-L. Chen, "Smart Home Energy Conservation Based on Context Awareness Technology," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, 2015, pp. 722-731.

[36] L. E. Holmquist *et al.*, "Building intelligent environments with smart-its," *Computer Graphics and Applications, IEEE,* vol. 24, no. 1, pp. 56-64, 2004.

[37] E. M. Tapia, S. S. Intille, and K. Larson, "Activity recognition in the home using simple and ubiquitous sensors," in *Proceedings of Pervasive,* 2004, pp. 158-175.

[38] A. Kameas, I. Mavrommati, and P. Markopoulos, "Computing in tangible: using artifacts as components of ambient intelligence environments," *Ambient Intelligence*, pp. 121-142, 2005.

[39] S. Antifakos, B. Schiele, and L. E. Holmquist, "Grouping mechanisms for smart objects based on implicit interaction and context proximity," in *Adjunct Proceedings of International Conference on Ubiquitous Computing (Ubicomp), Seattle, USA*, 2003.

[40] M. Beigl, H.-W. Gellersen, and A. Schmidt, "Mediacups: experience with design and use of computer-augmented everyday artefacts," *Computer Networks,* vol. 35, no. 4, pp. 401-409, 2001.

[41] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM,* vol. 43, no. 5, pp. 51-58, 2000.

[42] R. Muraleedharan and L. A. Osadciw, "Balancing the performance of a sensor network using an ant system," 2003.

[43] L. M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2001, vol. 3, pp. 1548-1557.

[44] M. S. Mahmoud and A. A. Mohamad, "A Study of Efficient Power Consumption Wireless Communication Techniques/Modules for Internet of Things (IoT) Applications," *Advances in Internet of Things*, vol. 6, no. 02, p. 19, 2016.

[45] J.-H. Chou, D. Petrovic, and K. Ramachandran, "A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, 2003, vol. 2, pp. 1054-1062.

[46] C.-K. Toh, "Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks," *Communications Magazine, IEEE*, vol. 39, no. 6, pp. 138-147, 2001.

[47] R. C. Shah and J. M. Rabaey, "Energy aware routing for low energy ad hoc sensor networks," in *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, 2002, vol. 1, pp. 350-355.

[48] S.-K. Noh, K.-S. Kim, and Y.-K. Ji, "Design of a room monitoring system for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2013.

[49] C. Suh and Y.-B. Ko, "Design and implementation of intelligent home control systems based on active sensor networks," *Consumer Electronics, IEEE Transactions on*, vol. 54, no. 3, pp. 1177-1184, 2008.

[50] M. W. Newman, A. Elliott, and T. F. Smith, "Providing an integrated user experience of networked media, devices, and services through end-user composition," in *International Conference on Pervasive Computing*, 2008, pp. 213-227.

[51] A. K. Dey, T. Sohn, S. Streng, and J. Kodama, "iCAP: Interactive prototyping of context-aware applications," in *International Conference on Pervasive Computing*, 2006, pp. 254-271.

[52] K. N. Truong, E. M. Huang, and G. D. Abowd, "CAMP: A magnetic poetry interface for end-user programming of capture applications for the home," in *International Conference on Ubiquitous Computing*, 2004, pp. 143-160.

[53] J. Coutaz, A. Demeure, S. Caffiau, and J. L. Crowley, "Early lessons from the development of SPOK, an end-user development environment for smart homes," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, 2014, pp. 895-902.

[54] D. Fogli, R. Lanzilotti, and A. Piccinno, "End-User Development Tools for the Smart Home: A Systematic Literature Review," in *International Conference on Distributed, Ambient, and Pervasive Interactions*, 2016, pp. 69-79.

[55] (2016) *IFTTT*. Available: *https://ifttt.com/*.

[56] F. Cabitza, D. Fogli, R. Lanzilotti, and A. Piccinno, "End-user development in ambient intelligence: a user study," in *Proceedings of the 11th Biannual Conference on Italian SIGCHI Chapter*, 2015, pp. 146-153.

[57] M. Resnick *et al.*, "Scratch: programming for all," *Communications of the ACM*, vol. 52, no. 11, pp. 60-67, 2009.

[58] J. Nichols and B. A. Myers, "Creating a lightweight user interface description language: An overview and analysis of the personal universal controller project," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 16, no. 4, p. 17, 2009.

[59] S. Mayer, A. Tschofen, A. K. Dey, and F. Mattern, "User interfaces for smart things--A generative approach with semantic interaction descriptions," *ACM Transactions on Computer-Human Interaction (TOCHI),* vol. 21, no. 2, p. 12, 2014.

[60] M. Botts and A. Robin, "OpenGIS sensor model language (SensorML) implementation specification," *OpenGIS Implementation Specification OGC,* vol. 7, no. 000, 2007.

[61] F. Leens, "An introduction to I 2 C and SPI protocols," *IEEE Instrumentation & Measurement Magazine,* vol. 12, no. 1, pp. 8-13, 2009.

[62] (2015). *Arduino Pro-Mini*. Avalable: https://www.arduino.cc/en/Main/ArduinoBoardProMini

[63] E. S. I. Team. (2015). *ESP8266EX Datasheet Version 4.3*. Available: http://download.arduino.org/products/UNOWIFI/0A-ESP8266-Datasheet-EN-v4.3.pdf

[64] (2015). *LiPolymer Battery Datasheet*. Available: http://www.powerstream.com/lip/GMB042035.pdf

[65] A. Hollinger and M. M. Wanderley, "Evaluation of commercial force-sensing resistors," in *Proceedings of International Conference on New Interfaces for Musical Expression*, 2006.

[66] V. Semiconductors, "TEMT6000: Ambient Light Sensor," *Heilbronn, Alemanha,* 2004.

[67] (2015). *Accelerometer ADXL345 Datasheet.* Available: www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf

[68] (2015). *Vibration Motor Datasheet*. Available: https://cdn.sparkfun.com/datasheets/Robotics/B1034.FL45-00-015.pdf

[69] (2015). *Flex Sensor Datasheet*. Available: https://www.sparkfun.com/datasheets/Sensors/Flex/FlexSensor.pdf

[70] A. E. C. Cloud, "Amazon web services," *Retrieved November,* vol. 9, p. 2011.

[71] *Google cloud platform*. Available: https://cloud.google.com/

[72] B. Von Solms and J. Roussel, "A Solution to improve the cyber security of home users," in *AFRICAN CYBER CITIZENSHIP CONFERENCE 2015 (ACCC2015)*, 2015, p. 157.

[73] L. A. Zadeh, "Fuzzy sets," (in en), *Information and Control,* vol. 8, no. 3, pp. 338-353, 1965.

[74] Y. Bai and D. Wang, "Fundamentals of fuzzy logic control—fuzzy sets, fuzzy rules and defuzzifications," in *Advanced Fuzzy Logic Technologies in Industrial Applications*: Springer, 2006, pp. 17-36.

[75] P. Cingolani and J. Alcala-Fdez, "jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation," in *FUZZ-IEEE*, 2012, pp. 1-8: Citeseer.

[76] G. Rademacher. *Railroad Diagram Generator*. Available: http://www.bottlecaps.de/rr/ui

[77] N. H. Phuong and V. Kreinovich, "Fuzzy logic and its applications in medicine," *International journal of medical informatics,* vol. 62, no. 2, pp. 165-173, 2001.

[78] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, "Neuro-fuzzy and soft computing, a computational approach to learning and machine intelligence," 1997.

[79] A. Karime, "CAHR: A Contextually Adaptive Rehabilitation Framework for In-Home Training," Université d'Ottawa/University of Ottawa, 2014.

[80] F. O. Karray and C. W. De Silva, *Soft computing and intelligent systems design: theory, tools, and applications*. Pearson Education, 2004.

[81] J. Zhang, G. Song, H. Wang, and T. Meng, "Design of a wireless sensor network based monitoring system for home automation," in *Future Computer Sciences and Application (ICFCSA), 2011 International Conference on*, 2011, pp. 57-60.

[82] D. Surie, O. Laguionie, and T. Pederson, "Wireless sensor networking of everyday objects in a smart home environment," in *Intelligent Sensors, Sensor Networks and Information Processing, 2008. ISSNIP 2008. International Conference on*, 2008, pp. 189-194.

[83] M. Kusserow, O. Amft, and G. Tröster, "Bodyant: Miniature wireless sensors for naturalistic monitoring of daily activity," in *Proceedings of the Fourth International Conference on Body Area Networks*, 2009, p. 9.

[84] C. Park, J. Liu, and P. H. Chou, "Eco: an ultra-compact low-power wireless sensor node for real-time motion monitoring," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, 2005, pp. 398-403.

[85] A. Flammini, P. Ferrari, D. Marioli, E. Sisinni, and A. Taroni, "Wired and wireless sensor networks for industrial applications," *Microelectronics Journal*, vol. 40, no. 9, pp. 1322-1336, 2009.

[86] (2015). *TI-INA169 Analog DC Current Sensor*. Available: www.adafruit.com/product/1164

[87] H. I. H. Aljamaan, "Model-Oriented Tracing Language: Producing Execution Traces from Tracepoints Injected into," University of Ottawa, 2015.

[88] W. Albert and T. Tullis, *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. Newnes, 2013.

[89] W. ISO, "9241-11. Ergonomic requirements for office work with visual display terminals (VDTs)," *The international organization for standardization*, vol. 45, 1998.

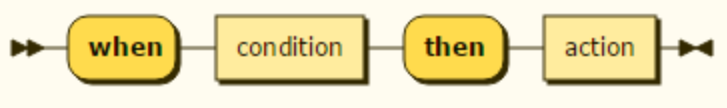
[90] A. Sergeev. (2010, March 3). "User Interfaces design and UX/usability evaluation," available: www.ui-designer.net/usability/efficiency.html

[91] J. Nielsen, "Quantitative studies: How many users to test," *Alertbox, June*, vol. 26, p. 2006.

[92] A. Smith, "Older adults and technology use," Pew Research Sener [Interner & American Life Project], 2014.

Appendix A: SCL Grammar Syntax Diagram

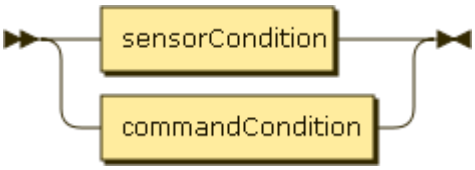
scl:




```
scl::= when condition 'then' action
```

no references

condition:

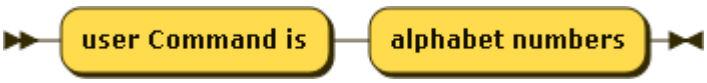



```
condition
         ::= sensorCondition | commandCondition
```

referenced by:

- conditionPrime
- scl

commandCondition:




```
commandCondition
         ::= 'user Command is' 'alphabet numbers'
```

referenced by:

- condition

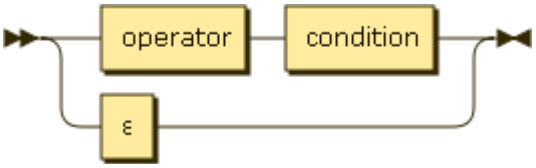
sensorCondition:


```
sensorCondition
         ::= 'sensor' sensorName 'is' sensorLevel period?
conditionPrime
```

referenced by:

- condition

conditionPrime:

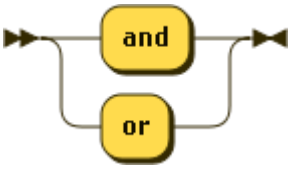



```
conditionPrime
         ::= operator condition | ε
```

referenced by:

- sensorCondition

operator:




```
operator ::= 'and' | 'or'
```

referenced by:

- conditionPrime

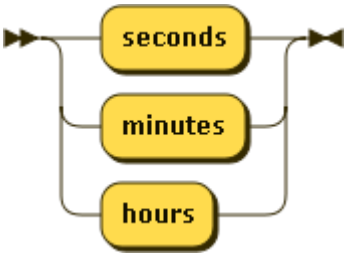
period:


```
period   ::= 'for' positiveInteger timeUnit
```

referenced by:

- sensorCondition

timeUnit:

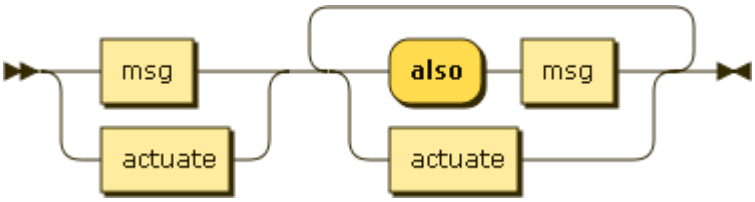



```
timeUnit ::= 'seconds' | 'minutes' | 'hours'
```

referenced by:

- period

action:




```
action   ::= ( msg | actuate ) ( 'also' msg | actuate )+
```

referenced by:

- scl

actuate:

```
actuate  ::= 'turn actuator' actuatorName actuatorLevel
```

referenced by:

- action

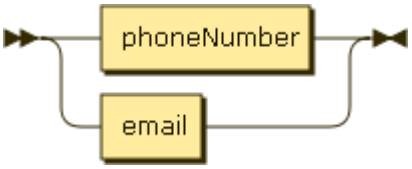
msg:


```
msg      ::= 'send message:' message 'to' msgDestination
```

referenced by:

- action

msgDestination:




```
msgDestination
         ::= phoneNumber | email
```

referenced by:

- msg

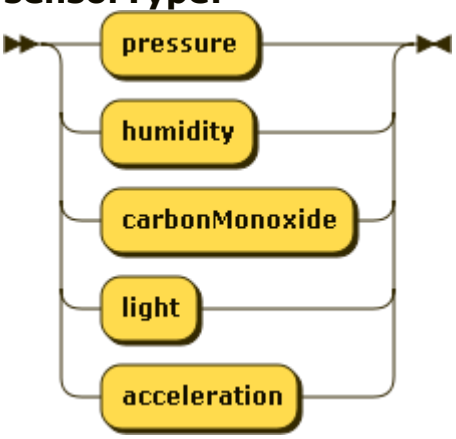
sensorName:


```
sensorName
         ::= SOName '. ' sensorType positiveInteger
```

referenced by:

- sensorCondition

sensorType:




```
sensorType
         ::= 'pressure' | 'humidity' | 'carbonMonoxide'
           | 'light' | 'acceleration'
```

referenced by:

- sensorName

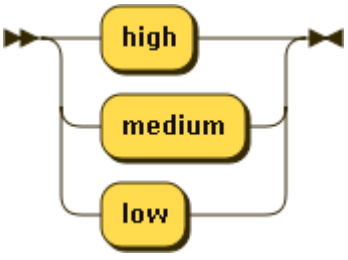
actuatorName:


```
actuatorName
         ::= SOName '.actuator' positiveInteger
```

referenced by:

- actuate

sensorLevel:

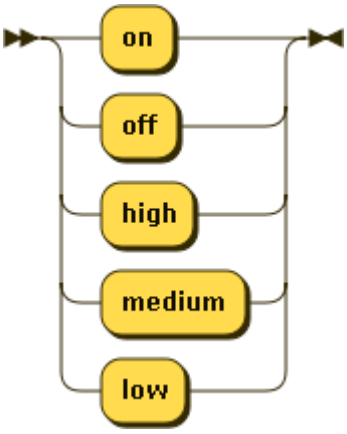


```
sensorLevel
          ::= 'high' | 'medium' | 'low'
```

referenced by:

- sensorCondition

actuatorLevel:

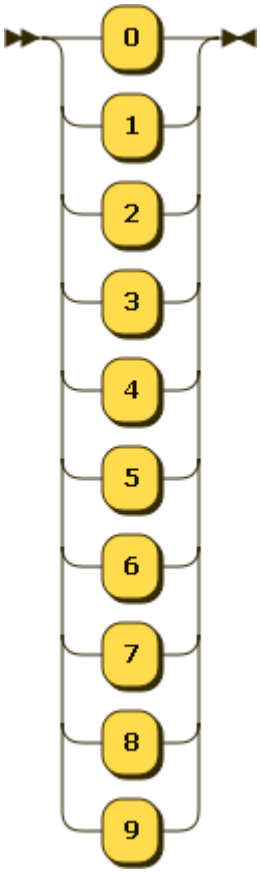



```
actuatorLevel
          ::= 'on'
            | 'off'
            | 'high'
            | 'medium'
            | 'low'
```

referenced by:

- actuate

digit:

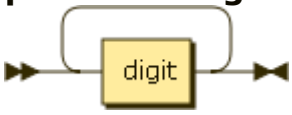



```
digit     ::= '0'
            | '1'
            | '2'
            | '3'
            | '4'
            | '5'
            | '6'
            | '7'
            | '8'
            | '9'
```

referenced by:

- positiveInteger

positiveInteger:




```
positiveInteger
          ::= digit+
```

referenced by:

- actuatorName
- period
- sensorName

Appendix B: Experimental Protocol

The following is the experimental protocol that is followed to guide each subject to conduct the experiments discussed in chapter 5 :

1. Meet the subject
2. Tell the subject to sit on the chair
3. Explain why we ask him/her to come
4. Ask the subject to sign the contest
5. Let the subject know that the experiment will be videotaped for time measurement purposes
6. Explain the purpose of the experiment
7. Let the subject know that the experiment will be divided into three scenarios, the first one will watch it in the video and will do it later, and the second and third scenarios, the subject will do them without help.
8. Open the computer
9. Play the video
10. Give the subject the instructions sheet (See Appendix C)
11. When the subject will be ready to do the experiment, start recording.
12. Make sure to run “Snagit” software before the subject runs the user interface and press Shift+F9 to record.
13. Press Shift+F10 to save the recorded video
14. Let the subject sit on the chair for 5 minutes to feel the actuator when it running.
15. Tell the subject to do the second and third scenarios.

16. Repeat steps (11-13)

17. Stop the camera

18. Ask the subject to fill in the Questionnaire

19. Thank the subject for spending his valuable time for doing the experiment.

Appendix C: Experimental Instructions

The following Instructions guide you to build any smart object in any smart environment.

The purpose is to test how easy to make any object smart using the following designed blocks and computer program.

The blocks:

The blocks are composed of a main box, which is used to communicate with the computer or mobile phone wirelessly, and small blocks which contain different types of sensors and vibrators, as shown in Figure 1. Depending on the kind of the physical measurement, one or more sensors can be attached to the main block.

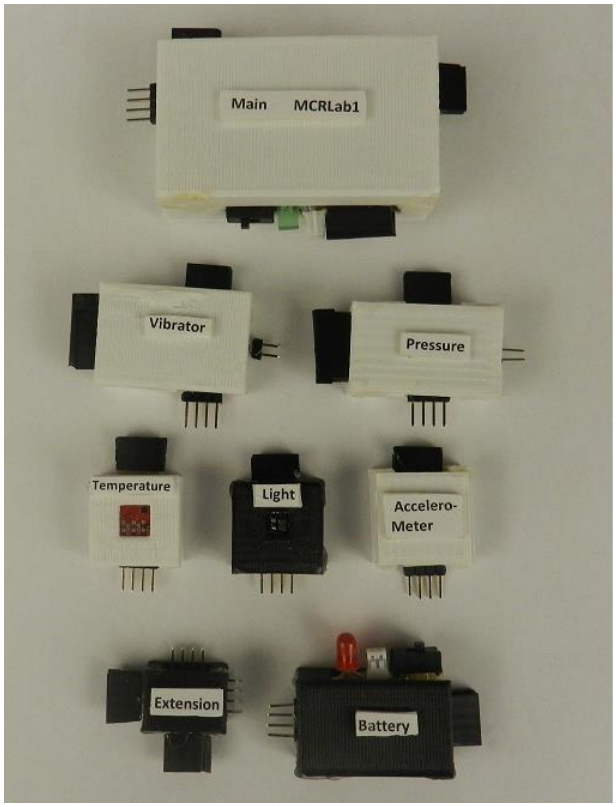


Figure 1 The blocks

The computer program:

The computer program is designed to monitor the sensors' measurements received from the blocks and also to control the attached vibrators.

In order to understand how the blocks and computer program are interacting with each other, we will consider the following two scenarios:

Scenario #1: Smart Chair

In this scenario your task is to make your chair a smart chair by letting the chair running a vibrator mounted on the back seat when you continuously sitting for more than 30 minutes. (In the experiment, you can reduce the time to 60 seconds).

In order to easy understand how to build the blocks and setup the computer program, we will divide your task into 3 tutorials:

Tutorial 1: Building the blocks

To build the blocks, you need to attach the desired small boxes to the main box. Make sure to face up the labeled side of each box.

1. Build the hardware part by connecting pressure sensor and vibrator boxes to the main box as shown in Figure 2.

Figure 2 The smart chair example

2. Mount the combination in a suitable place and mount the sensing part of the pressure sensor and the vibrator on the chair as illustrated in Figure 2.
3. Switch on the main box.

Tutorial 2 : Get familiar with the computer program

In this tutorial we will let the chair runs the vibrator after the user being sitting for a period of time (timeout). In this test, let the timeout be 60 seconds. The vibrator stops when the user leaves the chair. In this case we will use the predesigned computer program shown in Figure 3. This computer program interacts with the blocks by displaying the sensors' measurements and also controlling the vibrators. Follow the following steps to let the computer program interacts with your chair:

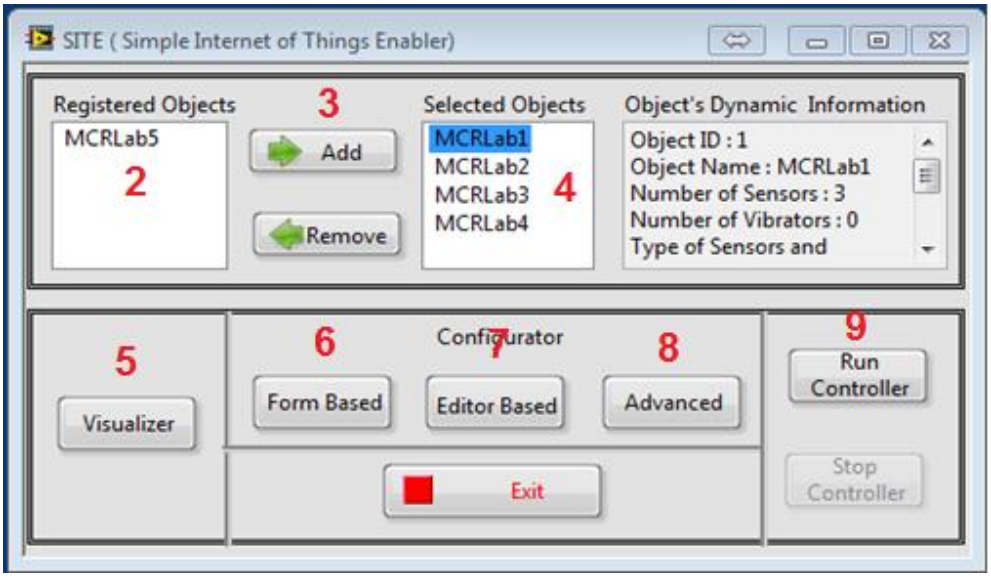


Figure 3 The computer program's main window

1. Run the program
2. Select your main block "**MCRLAB1**" from the **Registered Objects** box. We call the object "chair in this scenario" as "**Smart Object**"
3. Add it to **Selected Objects** box.
4. Click The MCRLab1 inside **Selected Objects** to see your node's general in the **Selected Node information** field.

   This main window has five main buttons: the **Visualizer, Form Based**, **Editor Based, Advanced** and **Run Controller.**
5. Press the **Visualizer** to open a new window. This window displays the sensors' measurements. In this window you will see the pressure sensor measurements.

   Click the **Exit** button to close the Display window and go back to the main window.

6. Press the **Form Based** button to open a Build Controller window. By this window (Figure 4), you will setup a controller by writing some rules in order to control your chair as follows:

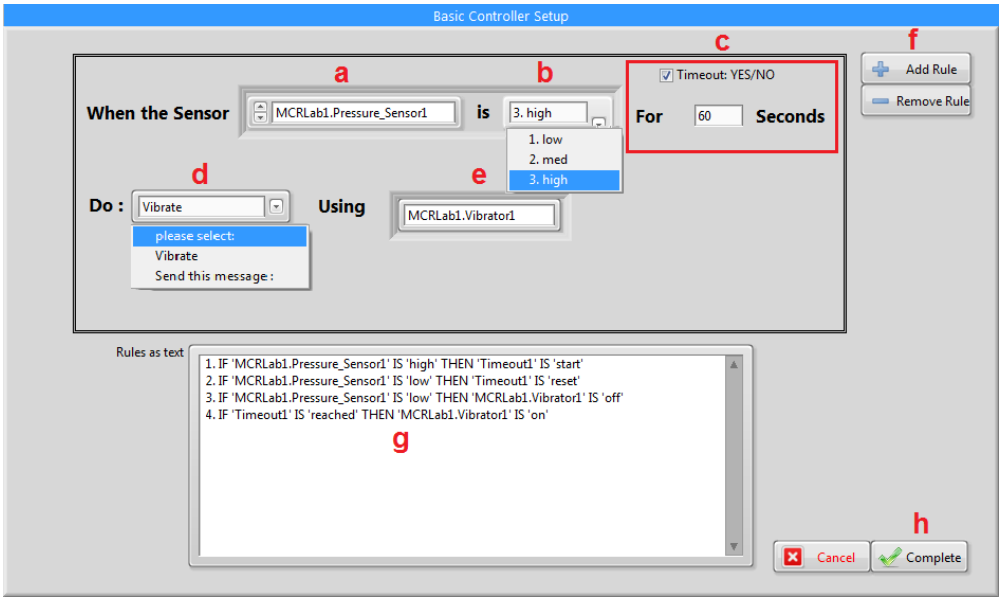


Figure 4 The Controller Setup window

a. Select the pressure sensor (**Pressure_Sensor1**) from the sensor's field.

b. The pressure sensor measurement values is divided into three ranges by default instead of numbers; low, med, and high, select the range **high**.

c. Check the **Timeout YES/NO** and select a period of 60 seconds for example.

d. Select **vibrate** from the **Do** field.

e. Select **Vibrator1** from the Using field

f. Click on **Add Rules.** After you adding rules and in case you found that you made a mistake, you can click **Remove Rules** to delete the last rules that you have added.

g. You will see the rules will be added as a text inside the white field on the bottom. These rules describe logically how the computer program will control the chair.
h. Click on **Complete** button to leave this window and go back to the main window.

7. The **Editor Based** button can be used if you want to write the rules using text rather than using a **Form Based**.
8. The **Advanced** button can be used by users which have knowledge of information technology and also have knowledge in fuzzy logic.
9. In the main window press **Run Controller** button. Now the vibrator will be on after you have being sitting on the chair for 60 seconds and stops vibrating when you leave the chair. Congratulations your chair is smart.
10. Press **Stop Controller** button and go ahead to the third tutorial.

Tutorial 3 : Detecting the user's wrong posture.

In this tutorial, the chair will tell the computer to display a warning message when you have not leaned your back on the seat for a predefined timeout (let's say 30 seconds as an example). In this tutorial you need to add another sensor block to detect whether your back is leaned on the seat or not (for example; light sensor as shown in Figure 5) and locate it in the back of the chair, then power the main box OFF and ON. Repeat tutorial 2 by adding more rules follows:

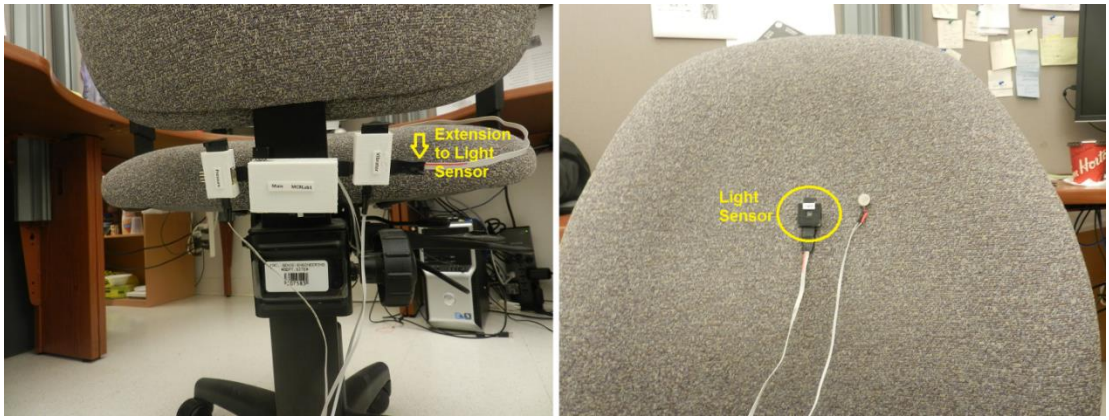


Figure 5 Mounting the light sensor block.

1. Run the program.
2. **Add** the node to **Your Nodes** field.
3. Press **Display Sensors.** Now you can see the measurements of two sensors, the pressure and light.
4. Click on **Exit** to go back to the main window.
5. Press **Basic Controller Setup.**
6. You will see two sensors in the sensors field as shown in Figure 6; the pressure and light sensors.
7. Write the rules for the pressure sensor as in tutorial 2 and Figure 4.

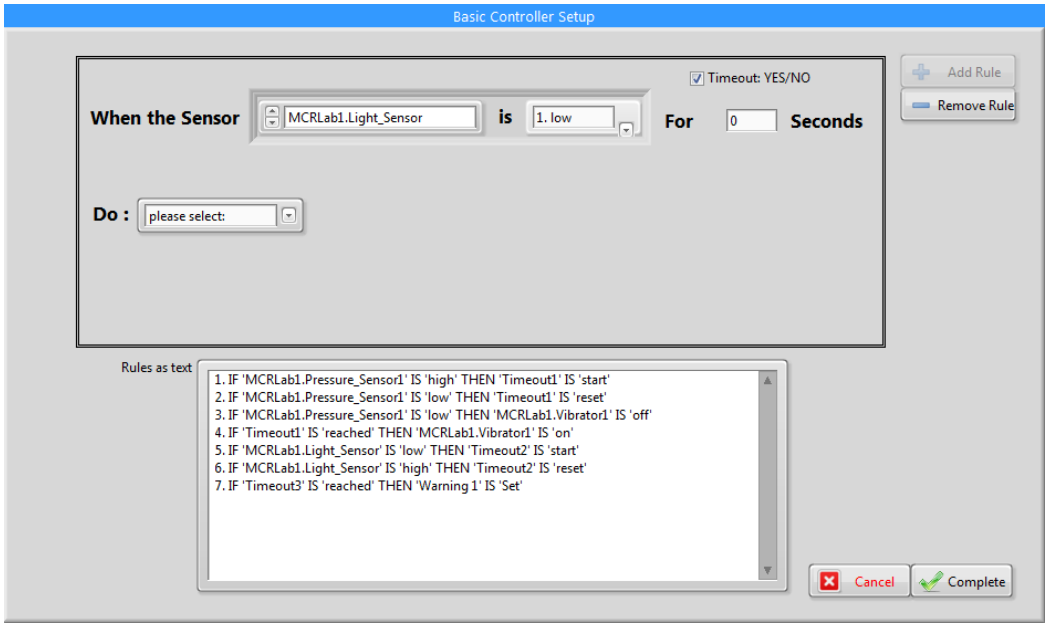


Figure 6 Adding light sensor

8. Add rules for the light sensor as follows (see Figure 6):

    a- Select **Light_ Sensor** from the sensor filed.

    b- Select **low** for the "is" field.

    c- Print **30** in the "seconds" field

    d- Select **send this message** from the "Do" field

    e- Write a warning message like, for example, " Please lean back to your seat" in the message field.

    f- In case you need to receive this message in your email address and/or your mobile phone, you can write your email address and phone number respectively.

    g- Click **Add Rules** button. Make sure that you have 7 rules in the **Rules as text** field; 4 rules for the pressure sensor as in tutorial 2, and 3 rules for the light sensor.

9. Click **Complete** to save the above settings and return back to the main window.
10. Press **Run Controller** and test your smart chair.

Now that you understand how to make objects smart, try to build, setup, and test the following scenario:

Scenario #2: Smart Fridge

1. Required sensors and actuators
    a. Pressure sensor
    b. Light sensor anywhere inside the fridge to measure if the door is open or not.
    c. Vibrator mounted on the door handle runs when the fridge door has been open for a long time.
2. The scenario

    In this scenario you need to make your refrigerator smart. It should send you a warning message when the eggs container is empty, and it also runs a vibrator when the fridge door has been opened for more than a specific time (assume 60 seconds).

Scenario #3: Smart Living Room

In this scenario your task is to make your living room smart.

3. Required sensors and actuators
    a. Light and Temperature sensors inside the room
    b. Pressure sensor
    c. On-off actuator for the TV
    d. 2 dimmers for both the lamp and fan

4. The scenario
    a. When the person opens the door to enter the room, the light will be on
    b. When a user sits on the couch the TV will be on
    c. The light intensity of the lamp changed depending on the room light
    d. The fan speed changed according to the room temperature

Thank you for testing my system.

Please fill in the questioners' sheet before you leave.