A framework for generating data stream & controller using pluggable IoT sensors and actuators

Abu Sayeed Bin Mozahid

(151-35-843)

Md.Ariful Islam

(151-35-937)

A thesis submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Software Engineering

Department of Software Engineering

Daffodil International University

Spring-2019

APPROVAL

This **Project/Thesis** titled "**Autonomous service creation framework for pluggable IoT sensors & actuators**", submitted by **Abu Sayeed Bin Mozahid (151-35-843) & Md. Ariful Islam (151-35-937)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc in Software Engineering and approved as to its style and contents.

BOARD OF EXAMINERS

Dr. Touhid Bhuiyan
Professor and Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Chairman

Md. Maruf Hassan
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 1

Asif Khan Shakir
Lecturer
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Dr. Md. Nasim Akhtar
Professor
Department of Computer Science and Engineering
Faculty of Electrical and Electronic Engineering
Dhaka University of Engineering & Technology, Gazipur

External Examiner

DECLARATION

It hereby declere that this thesis has been done by us under the supervission of **K.M.Imtiaz-Ud-Din, Assistant Professor,** Department of Software Engineering, Daffodil International University. It also declere that nithor this thesis nor any part of this has been submitted elesewhere for award of any degree.

Abu Sayeed Bin Mozahid
ID: 151-35-843
Batch:16th
Department of Software Engineering
Faculty of Science & Information Technology

Daffodil International University

Md. Ariful Islam
151-35-937
Batch:16th
Department of Software Engineering
Faculty of Science & Information Technology
Daffodil International University

Certified by:

K.M.Imtiaz-Ud-Din
Assistant Professor
Department of Software Engineering
Faculty of Science & Information Technology
Daffodil International University

ACKNOWLEDGEMENT

This document presents our Bachelor of Science Thesis "**A framework for generating data stream & controller using pluggable IoT sensors and actuators**". Appreciatively, we received advice from various people to whom we want to explicit our acknowledgment towards in this section.

Our sincerest thanks to our supervisor and mentor **K.M. Imtiaz-Ud-din, Assistant Professor**, **Department of Software Engineering**, **Daffodil International University** for providing us to a chance to do this. Without his encouragement, guidance and motivation we would not be able to realize the structure and got an output.

We also thankful to our mentors **Kaushik Sarker** and **Md. Anwar Hossen** for all the support to execute this milestone.

Grateful to the Ambient Intelligence Lab group members especially **Diganta Protic Biswas**, **Avee Chakraborty**, **Md. Ashiqur Rahman** who always work and help with us in this research project.

We also thankful to **Professor Dr. Touhid Bhuiyan Head of Department of Software Engineering, Daffodil International University**, and all the teacher in our department.

We also thank to researchers for their works which help us to learn design and implement our research project.

Last but not least, we want to thank almighty Allah for giving us patience. We are also thankful to teachers, family-members, friends, seniors, juniors for providing their effective support and prayers.

Contents

List of Figures

ABSTRACT

In the advancements of technology the interest in Internet of Things domain has increased remarkably, specially in the end user kind. People are using various smart systems in order to improve their lifestyle. Therefore end users without any technical experience started using those systems. But there are no such end-user friendly smart systems where users who don't know any aspects of programming can control and add different services to the existing system based on their needs. In this work we had proposed an architecture where end-users without programming knowledge can control over the services. On the other hand service composition and adaptation at runtime is necessary in order to satisfy the users specifications. We showcase a prototypical implementation of our architecture in the domain of smart home.

Chapter 1

INTRODUCTION

Last two decades, Peoples have addicted to the internet. They are so interested in any kind of product which is spontaneously connected with internet. In the early age Internet of things (IoT), people had two options to using IoT device: Owner has to write program for their device or can use rigid IoT hardware device where no need any kinds of programing. But static IoT hardware device do not solution for dynamically work. It is okay when owner know the programing knowledge but researcheres are so worried about End-user who do not have any programming idea. Nowadays there are many home automation tools exist such as IFTTT, openHAB, Zapier, Node-RED and StreamBase Studio where users can interact with their devices within their home to perform different tasks according to Mahda Noura, Sebastian Heil and Martin Gaedke(2018). There are some industrial solutions too from different companies such as Samsung Smartthings hub & Philips Hue but those systems are rigid. They don't provide users that support to configure services based on their needs .Users can not integrate more sensors and actuators with the system and they can not compose different services together. So, here we reviewed researches done in this area and the industrial solutions but no one is concerned about hardware program where end-user will write or customize a program for their device to connect and modify their own services based on their needs in any IoT system. In these aspects we are trying to strictly use the technology for creating a self-adaptive system where any sensor or service can adapt to the system with owner desires and an intelligent user-interface can discover services automatically.

We propose an architecture which supports microservices and distributed service and end-user gets ease to customize his hardware.To address this we have identified four major components:

Sensor or Actuator Service: A device that can be used to measure a physical property of the real world or can perform an operation or control a system.

Web Service: Supports RESTful APIs

Middleware: Computation Engine where Service adaptation and task triggering actions performed.

User Interface: An Intelligent GUI to support users to make service composition based on their needs.

Two groups have worked in this concept. Group A handle Sensor or Service and Web Service components and Group B handle Middleware and User Interface. We are working with Group A thats why rest to thesis we will be talking about sensor or service and web service.

1.1 MOTIVATING SCENARIO

Last year, Ambient Intelligence Research Lab represented an architecture where all components were distributed and end-user can compose any service with other a third party service such as an actuator service can perform with a weather API by Md. Raihan Uddin, Aziha Kamal, Shuvo Prosad, Antara Saha(2018). We have extended their approach and improving their given architecture. By a real scenario, we will describe where we have contributed and how to improve their architecture.

Scene 1: Mr. Akbar bought an automated water pump system for his house to automatic filling the tank when it's empty. He has a user interface to control the service. He was so happy with that system and service because he don't have to worry about water tank anymore. So he wants to add more services in the system, he wants to register a new sensor which will help him to automatically open his super shop's door when anyone comes closer to the door. So, Mr. Akbar bought another sensor for his door and recently changed his WIFI router SSID and password and after that, the system is not working well. He had tried to fix it up but alas he doesn't know any programming related things. So he was so worried and call a professional for help with his system. Then a professional fix his existing sensor and its working as before but when Mr. Akbar wants to add new sensor the professional told him that he has to manually register the sensor with the server.

Scene 2: Mr. Rahim wants to use that sensor to do two different things. He wants to turn on the light and open the door while there is someone closer to the door. So here he wants to compose two actuator services together Two actuators actions here will be control the lights and fan inside the shop, when there is nobody left on the shop lights and fans will automatically turn off and if there is someone enters through the door lights and fans will turn on automatically.

Scene 3: Jihad wants to check the weather status and based on that he wants those lights and fans to be turned on or off. So here Mr. Jihad wants to compose web services and IoT based services together. Based on the weather condition two actuators will perform certain actions.

1.2 OBJECTIVE

Our ambition is to build an intelligent system where users can collaborate on different IOT based services and web services and can configure the services based on their needs and the services can be adapted at run time.Each service is loosely coupled.Users only have to connect the sensor or actuator with his computer and our intelligent system will give him/her a interface to enter some pieces of information regarding the sensor or actuator and the system will automatically create the service against the sensor or actuator. And after creating the service user can customize the service and can use based on their needs. That's where our main objectives are met, This can be defined as:

1. Pluggable IoT sensor and actuator
2. Generating data stream and controller for pluggable IoT sensors and actuators

1.3 OUTLINE OF THE THESIS

Chapter 2 deigned with literature review and related technologies. Here briefly describes the related research work in this area and which technologies we are using.

In chapter 3 describes the proposed system architecture and how it works.

In chapter 4 cover the proof of concept of the research project following the proposed architecture.

In chapter 5 describes evaluation with a scenario and show an implementation with practical works with picture.

In chapter 6 concludes the thesis work by specifying contribution, limitation and future work.

Chapter 2

LITERATURE REVIEW

We investigate the state of the art in the research areas addressed in this paper. To the best of our knowledge, almost all the researchers concerned about handling middleware where end users will compose any service without any programming knowledge but they are not thinking about Hardware programming for end-users and how will they connect and register with the system to create composite services. We divided this section into two parts, one is related researches and another is related existing industry solution.

2.1 RELATED RESEARCH WORKS

A few works have done in the context of hardware such as sensors and actuators. Among them, the most remarkable one is personal application in the IoT through visual end-user programming by Valsamakis, Y., & Savidis, A. (2018), they used RFID tags for identifying different sensors and actuators.

Actinium:A Restful runtime container for scriptable internet of things applications by Matthias Kovatsch,Martin Lanter,Simon Duquennoy, they proposed a runtime environment that exposes the scripts themselves through a RESTful interface and also indicated WSN programming as the network is built upon a simple set of actions: periodic sensing,alarm triggering and actuation.

A Multi-Agent-Based Intelligent Sensor and Actuator Network Design for Smart House and Home Automation by Qingquan Sun , Weihong Nikolai Kochurov, Qi Hao, and Fei Hu.

According to them based on the contextual data from the sensor a smart house can provide services, such as energy efficiency control, security monitoring and medical assistance.

Moving Application Logic from the Firmware to the Cloud: Towards the Thin Server Architecture for the Internet of Things by Matthias Kovatsch, Simon Mayer, Benedikt Ostermaier.

2.2 EXISTING INDUSTRY SOLUTIONS

Samsung SmartThings Hub : It provides home automation solutions. It can connect via Wifi with a wide range of smart devices including lights, cameras, locks, thermostats, sensors etc and makes them perform actions. End users has to use the Google Assistant, Amazon Alexa, or one of the SmartThings apps to control the system.

Philips Hue: it's a smart lighting system. Philips Hue system is the Bridge which acts as a smart hub, connecting devices to smart lights. End users can add up to 50 Philips Hue lights and accessories to one Bridge. It can be controlled using Philips Hue app.

Apple smart home (HomeKit) : HomeKit is a framework developed by Apple. It connects home automation products. End users can control them with voice, via Siri, and taps, via the Home app for iOS.

These smart home solutions are rigid. Users can't integrate different sensors or actuators with the system as per their requirements and can't make composition with various services together whatever is needed.

2.3 RELATED TECHNOLOGY AND FRAMEWORK

Microservice architecture: Microservice architecture is an architectural style that structures an application as a collection of services that are :

- Maintainable and testable
- Loosely coupled
- Can be deployed independently

Its an architecture that structures the application as a set of loosely coupled, collaborating services.

RESTful APIs: An API is an application programming interface.Its a set of rules that will allow different programs to communicate with each other.By creating API on the server, we are allowing clients to communicate with the server.

REST gives a set of rules that we have to follow when creating the APIs. RESTful Web services allow the requesting systems to access and manipulate textual representations of web resources by using a uniform and predefined set of stateless operations.

In our system the clients and server exchange representations of resources by using a standardized interface and protocol HTTP. RESTful access uses the following four HTTP verbs to determine which action to take on a particular object:

- GET: Retrieves the current state of the object
- PUT: Sets the current state of the object
- POST: Creates a new object
- DELETE: Deletes the object

Joseph Benharosh had decribed full restful api working process in his website by Figure 2.1 diagram.

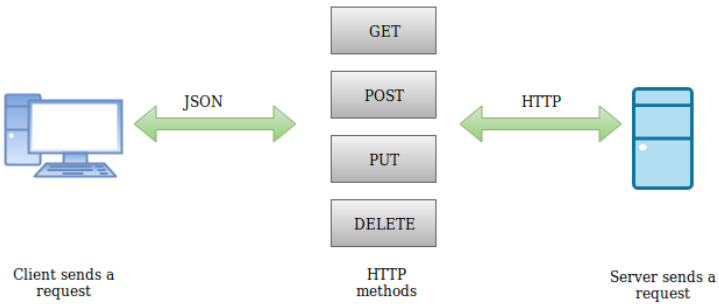

Figure 2.1: RESTful API working process

Wireless Sensor Network(WSN): Wireless Sensor network is a network that can certainly cover a huge number of spatially distributed, little, battery-operated, embedded devices that are networked to caringly collect, process, and transfer data to the operators, and it has controlled the capabilities of computing & processing.

Distributed Service Architecture: All the components such as sensors, actuators, and web services are independent and isolated.They can function asynchronously.If any of the service is not working properly the other one will not stop to function.

Arduino-cli: Command Line Interface(CLI) is a tool that launched by arduino for compile and uploading arduino code from using windows command prompt or Linux and MAC Terminal.

Chapter 3

DESCRIPTION OF PROPOSED SYSTEM

3.1 Proposed System Architecture

Figure 3.1 describes an architecture where users can compose their services based on their specific needs and our two objectives are met. The main components of this architecture are Sensor service, Actuator service, Terminal Agent, server, watcher, end-user Interface and composite service generation engine, code execution engine.

End-user Interface and composite service generation engine: The user interface provides the means to coordinate the user inputs to interact with the system. End users can easily create composite services without any programming knowledge just by drag and drop. This is where the goals are issued.

Terminal Agent: The terminal agent is responsible for gathering information about the new sensor or actuator from the user and generate customize program suitable for the IoT device and connect them with the system to create service against them.

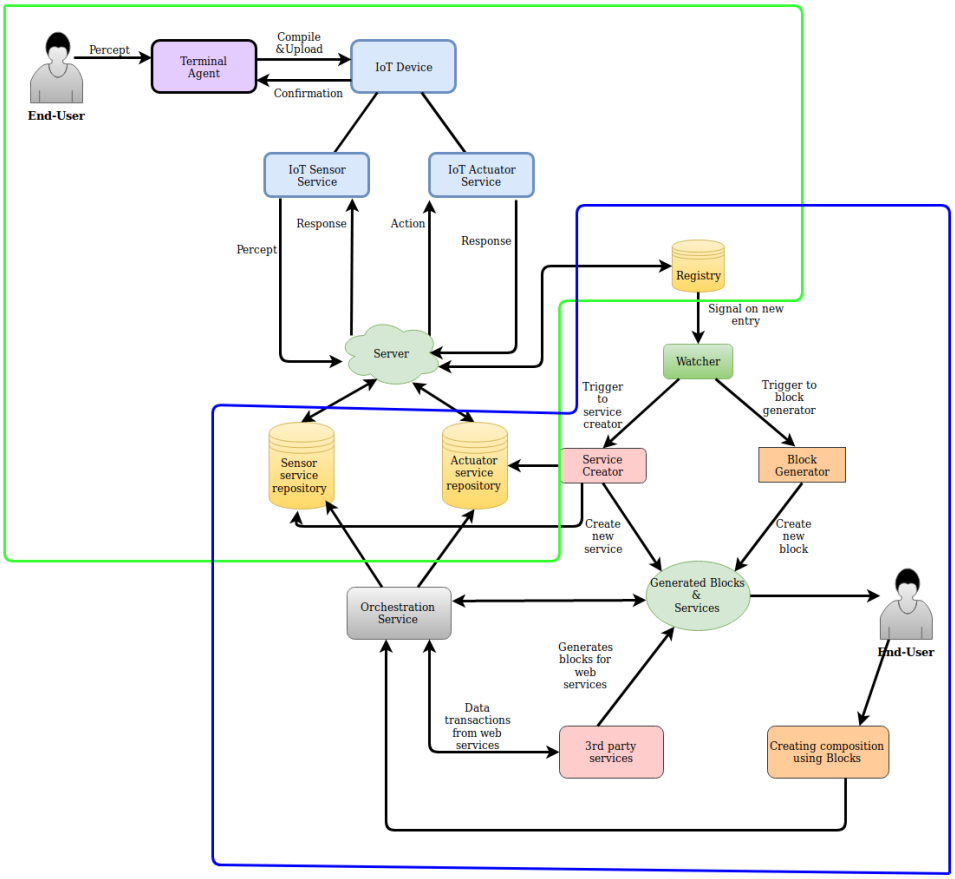

Figure 3.1: Proposed System Architecture

Sensor Service: Sensor can be defined as a physical object that precepts a change in the environment and responds to that change with an electrical signal output. The role of sensor service is to simultaneously monitor the environment for any state changes. If occurred the data is sent to the server and those data can be processed, analyzed, reported or can be used to trigger another event. Consider a situation where users would like their water pump to be switched on when the water level in the tank goes below a certain level. That's where the role of the sensors comes in. The sensor will measure the level of water. Our system is designed such a way that once the sensor learns that the water has gone down a particular level, it will send an alert to the controller (Middleware). The controller, in turn, would trigger an actuator to switch on the water pump.

Actuator Service: Actuator is a device that can control certain materials.Its a mechanism to turning energy into motion. It's generally used to apply force on something. In our example above, the actuator would apply force to switch on the motor of the wa-

ter pump. Actuators can create linear, oscillatory or rotatory motion based on how they are designed. It can trigger various devices into operation based on the dynamics of the data.

We can classify actuators as either binary or continuous devices based upon the stable states available in their output. For example, a relay is a binary actuator as it holds two stable states, either energized and latched or de-energized and unlatched. Whether the motor is a continuous actuator because it can rotate through a full 360-degree motion.

In our architecture, we used those three types of actuators. One of them used for automated door and another two used for water tank and light fan control.

Server: As we mentioned before that we are extending previous works done by Md. Raihan Uddin, Aziha Kamal, Shuvo Prosad, Antara Saha(2018) under Ambient Intelligence Lab, they used CloudMQTT Protocol Server for connecting with gadgets, sensors and to publish and subscribe packets. We have replaced it with HTTP server for exchanging and storing data from our various IoT based devices.

The core architectural principles to the web can be defined as Representational State Transfer(REST). It shares a similar goal as web services. Web services are for increasing interoperability between the parts of a loosely coupled distributed applications, whether REST is for achieving this in a more lightweight manner seamlessly integrated to the web.

In our architecture, we proposed a system where applications are realized through scripts running on the server accessing the initial functionality of IoT based services through their RESTful interfaces. REST uses URIs for identifying different services on the web, resources can be linked, bookmarked, cached,searched for and the results are directly visible on the browser. The lightweight and ubiquitous aspects of REST makes it ideal to build APIs for our system.

Registry: Registry is responsible for holding services. When any new services added to the system registry repository got one new entry for that particular service. If the services is already registered then registry repository will ignore the request.

Sensor & actuator service repository: Sensor service repository is for bearing various sensor data and actuator service repository is responsible for holding actuator data. By using sensor data from the repository users can perform various actions by using the actuators.

How this architecture solves our two objectives: For making the IoT sensor and actuator pluggable we are using Terminal Agent and its responsible for gathering required informations and customize the program to integrate any IoT device with the system. That makes IoT devices pluggable and it where our first objective is met.

After terminal agent upload the program file to a new IoT device, it will be trying to register with the system and the server will check that service is exist or not. If it is new service, it will register with system. By registration, sensor will be streaming the data to sensor service repository and actuator will be waiting for an action to control the device from actuator service repository. Thus we created data stream and control for IoT based devices and this is how we solve our second objective.

Chapter 4

PROOF OF CONCEPT

4.1 SENSOR AND ACTUATOR MODULE

For creating prototype, we has created sensor and actuator module where every sensor or actuator will connect with a wifi communication base micro-controller such as NodeMCU at Figure 4.1 and Figure 4.2 . By this purpose, sensor and actuator module can easily satisfy our first object.

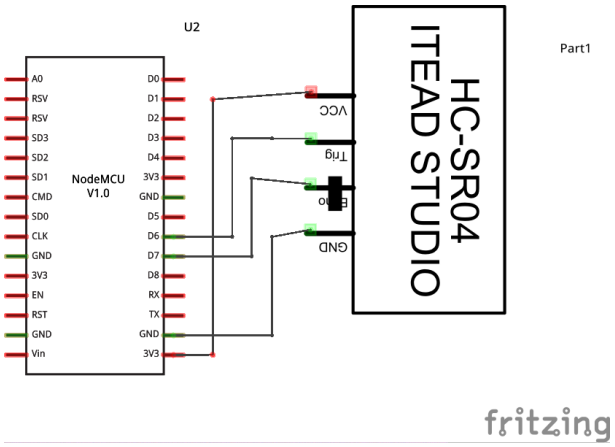

Figure 4.1: Sensor Module

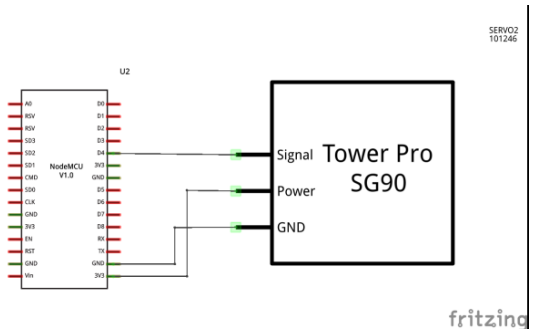

Figure 4.2: Actuator Module

4.2 IoT Sensor, Actuator and Web Service Working Module

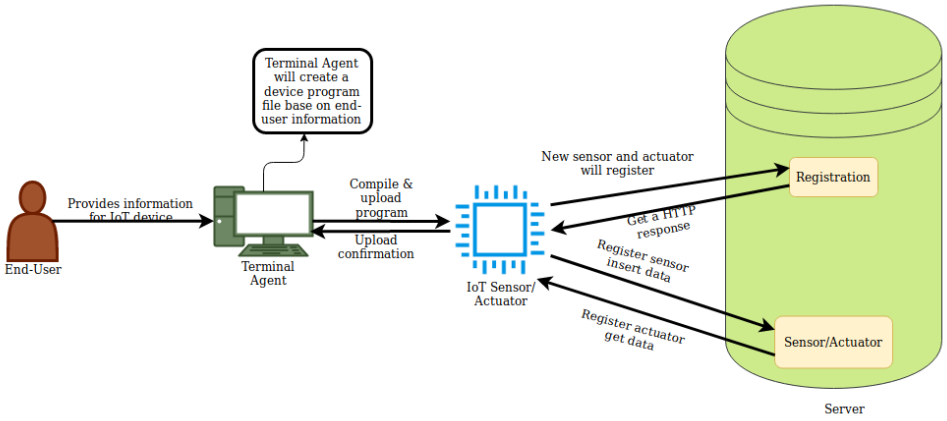

Figure 4.3: Sensor, Actuator and Web Service Working Module

Figure 4.3 depict that End-User will buy a sensor or actuator service from market and to our system for download user manual. That manual will describe how end-user will connect their iot device with system.

4.2.1 Coordinates

For position of the sensor or actuator users has to enter lattitude and longitude before conecting to the system and our system does not provide any services for aquiring coordinates. So users has to find coordinates from another websites or apps such as www.latlong.net.

4.2.2 Terminal

After collecting all information, user will download device program customize tool from our system and provide some information. Those are

- Owner Name
- Sensor Name
- WiFi SSID and password
- Lattitude and longitude

Firstly users have to enter their name in the feild of Owner name,For entering sensor name our terminal agent will show examples of naming conventions for sensor name and users has to follow the exact same fashion. Then agent will ask for wifi SSID and password,users have to enter exactly same as their internet configuration.Lastly it will ask for coordinates and users have to gather them(lattitude,longitude) from other apps or websites. After entering these informations our terminal agent will automatically generate a customized program to connect the IoT device to our system.

See code Appendix A.1

Tag

Our terminal program will combine the given informations by users to create an unique tag for the connected iot device.With this tag users can identify the service against that iot device as well as our system.

See code appendix A.2

Upload

After generating the suitable program for specific iot device our terminal program will ask to user: Do you want to upload the program? y/n. If user press 'y' then program will check the operating system whether its windows or linux.Based on the operating system it will generate commands for compiling and uploading program to the device.After successfully uploading the iot device will be connected to the system and end users can easily create composite services using the device.

See code appendix A.3

4.2.3 Context sensing:

The role of context sensing is to monitor the status of the IoT devices found at runtime and inform the corresponding components upon state change. Sensor services always monitor the status of the connected physical products.

4.2.4 RESTful API

When the IoT device is connected with our system, device will provide its tag to the server as a json data. See code appendix A.4 Server will take that json format and deserialze the data. After deserialze data, server will check that with its registry table. If that tag is unique, server will save it otherwise it will feedback 400 HTTP response. Every device will follow this process at least three times after device can get power. If the service is already registered then the server will create a row regarding the service in the sensor or actuator registry.

We have three APIs integrating with the server.Such as:

1. serveraddress/api/serviceregistry
2. serveraddress/api/sensors
3. serveraddress/api/actuators

Every IoT device have two APIs. Service registry API is common for every device and sensors API for sensors and actuators API will exist in actuators devices. See code appendix A.5, A.6 & A.7

Chapter 5

EVALUATION

To evaluate our concept we adopted an environment where users would like to open his door when anyone comes closer to the door and turn on the light or fan when needed based on the context.Our main goal was to prove that the implemented support for self-adaptive service composition at runtime based on user requirements works as expected.There are many services can be created by the user and we kept the functionalities as simple as possible.

When instantiated our system will provide a graphical user interface with some informations such as registered services, actuators, instructions and service compostion engine. We followed the steps below:

1. User connects a sensor with his computer and by following the instructions user has to run terminal agent application. Terminal agent will take some information from the user and generate a customized program for connecting their IoT device with our system. Then it will ask end-user to upload the program to microcontroller. If uploading is successful the IoT device is configured with our system.

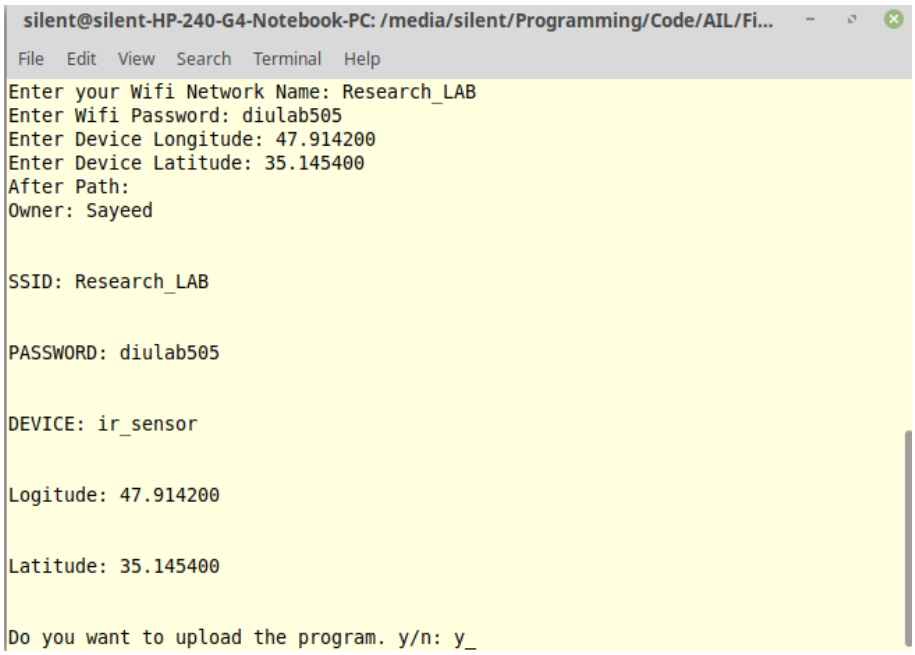

Figure 5.1: Terminal Agent procedure

2. If uploading is successful a service will be created based on the IoT device on the blocks. Our system will automatically place that block in an appropriate place whether its a sensor service or actuator service.

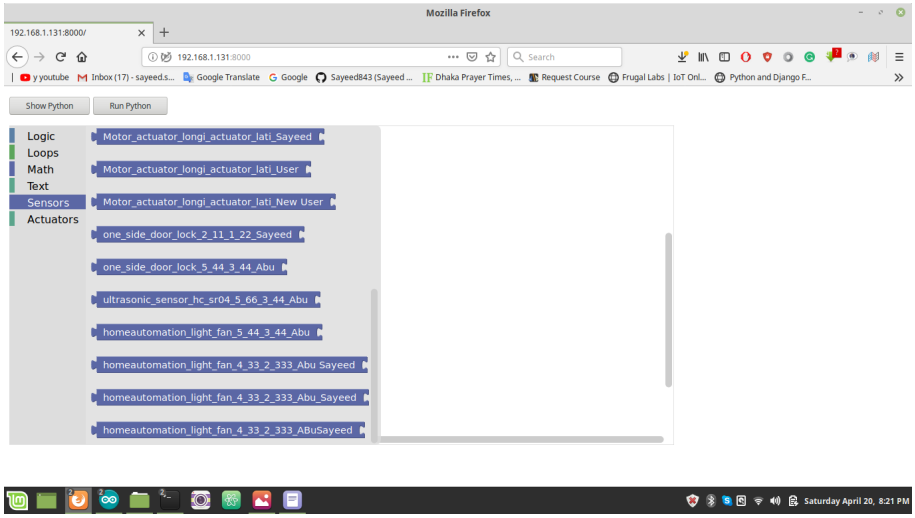

Figure 5.2: Generated Blocks

3. Now, the user can create service composition by drag and drop from the blockly blocks. They can customize and set conditional steps using the blocks.

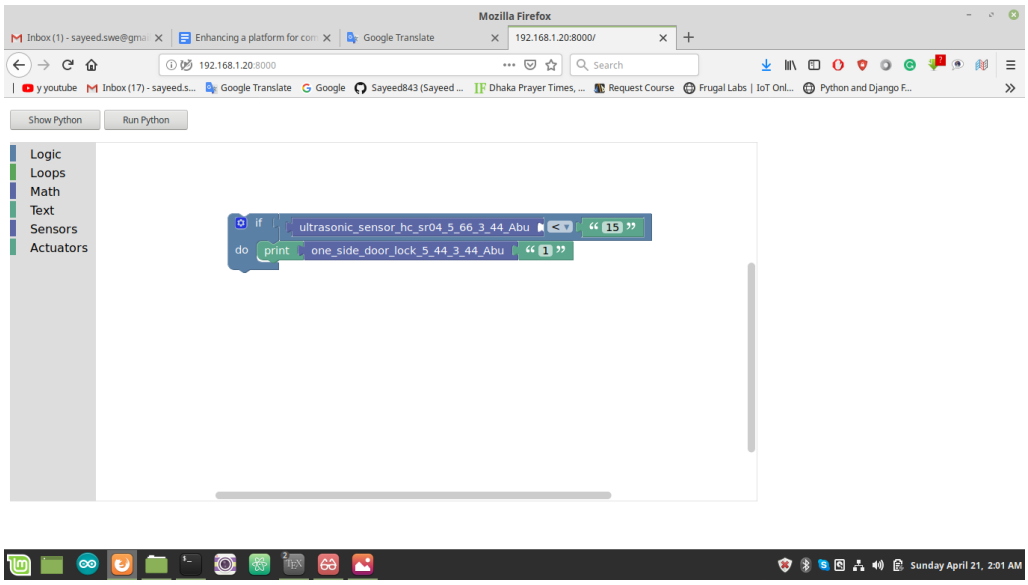

Figure 5.3: Door unlock program

After clicking run the program will send a command to the corresponding actuator for opening the door.

Figure 5.4: Door opens

4. For evaluating the concept of pluggability to create new services end-users can reuse the same sensor for multiple actuator services.

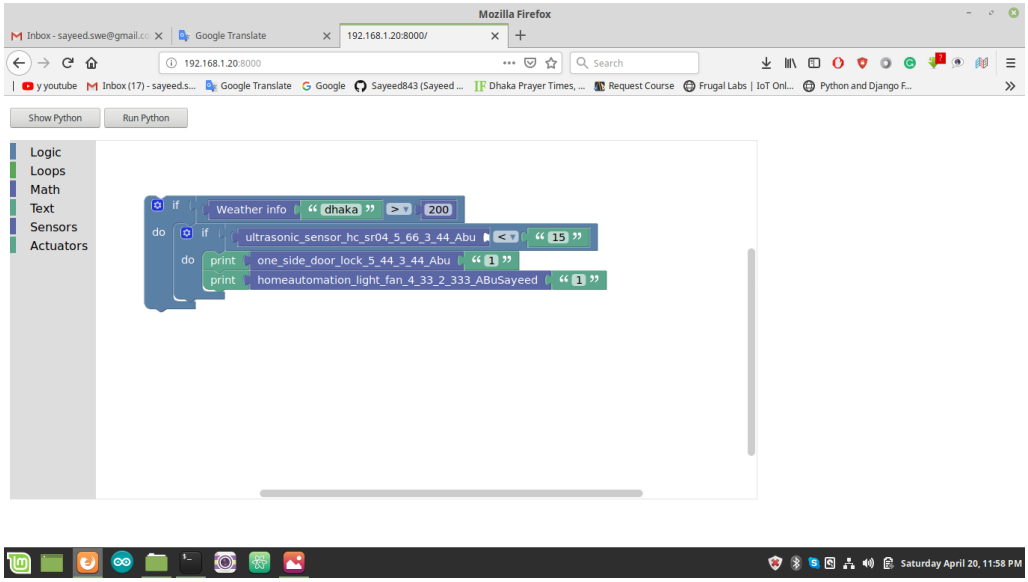

Figure 5.5: Multiple actuator service creation

5. User can unlock the door and turn on the light using the same sensor. We have set an exact range for percepting obstacles. If the sensor finds out that there is someone closer to it then it can trigger two actuator actions, one for opening the door and another is for turning on the light.

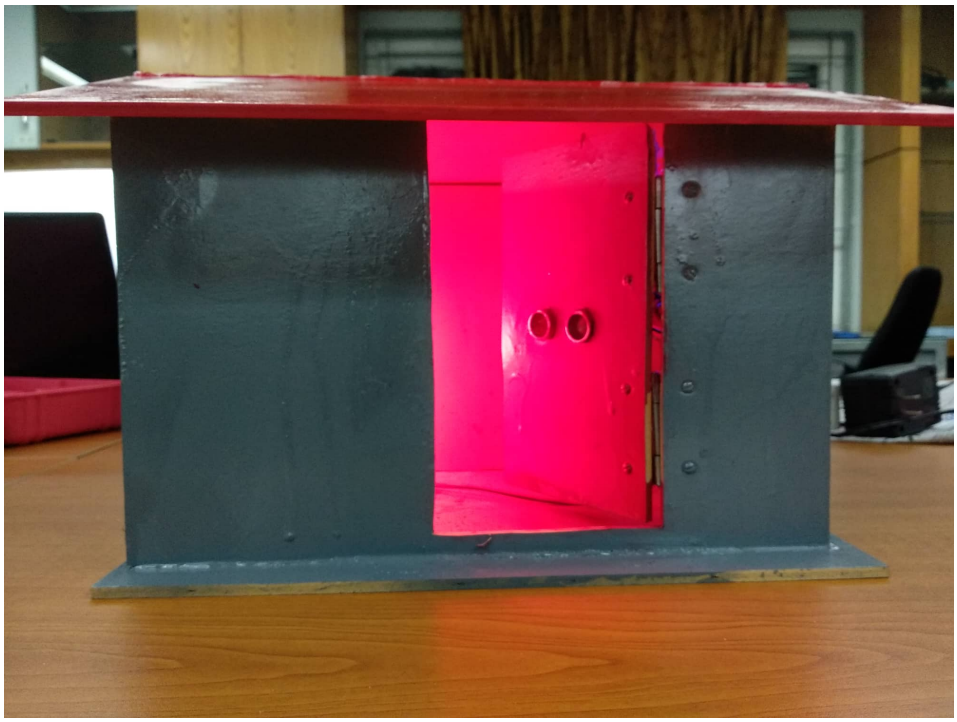

Figure 5.6: Multiple actuator actions

6. Other than IoT based services our architecture can support third party web services such as user can monitor weather conditions. With this service user can create a composition using IoT based services. We can compose weather api and sensor services data together and trigger actuator actions based on that data.

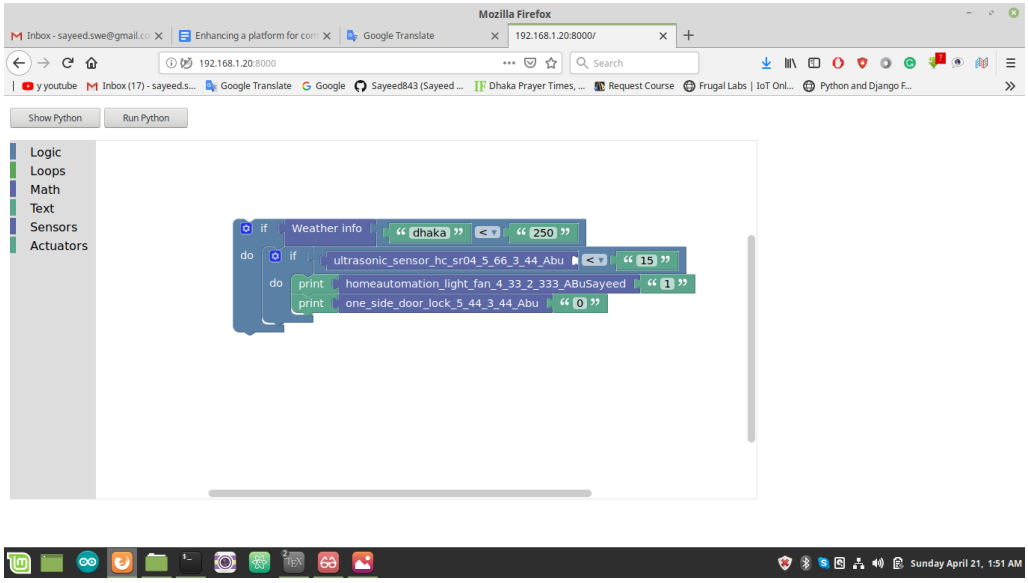

Figure 5.7: Actuator service creation with weather api and sensor

7. Suppose we have a situation where the weather condition is really bad and user needs to turn on the light. He can create a service composition with weather api and ultrasonic sonar sensor and can trigger the actuator for turing on the light and closing the door.

Figure 5.8: Actuator service action with weather api and sensor

As we declared three scenarios in section 1.1 those are working properly as we illustrate three different service compositon above.

Chapter 6

CONCLUSION

We presented in this paper a platform we are building for the Internet of Things, where end-users can easily compose their services based on their necessities. Our architecture can excite end-user programming in a new way. They can control their services through service composition and runtime adaptation.If consumers want to add new services we are providing the mechanism to connect the IoT device to our platform for creating a iot based service and that makes our architecture pluggable. Services are loosely coupled if any of them failed another one can run smoothly.

For validating our approach we had designed three scenarios and we have two objectives. By using our architecture we had built a prototype and it works as expected based on those three scenarios and our two objectives meet.

In future we would like to add more functionalities to our architecture so that end users can create more complex service composition and to make our system more effective and intelligent. We want to analyze end-users previous data to automatically discover and identifying specific services for better decision making and performance.

Chapter 7

REFERENCE

1. Mahda Noura, Sebastian Heil, & Martin Gaedke (2018). GrOWTH: Goal-Oriented End User Development for Web of Things Devices. *Proceeding of the International Conference on Web Engineering.*

2. Valsamakis, Y., & Savidis, A. (2017). Personal Applications in the Internet of Things Through Visual End-User Programming. *Proceeding of the Digital Marketplaces Unleashed, 809-821.*

3. Matthias Kovatsch, Martin Lanter , Simon Duquennoy (2012) .Actinium: A RESTful runtime container for scriptable Internet of Things applications. *Proceeding of the 3rd IEEE International Conference on the Internet of Things.*

4. Qingquan Sun , Weihong Nikolai Kochurov, Qi Hao, and Fei Hu (2013) .A Multi-Agent-Based Intelligent Sensor and Actuator Network Design for Smart House and Home Automation. *Proceeding of Journal of Sensor and Actuator Networks.*

5. Kashif dar, Amirhosein Taherkordi, Roman Vitenberg, Roman Rouvoy and Frank Eliassen (2011). Adaptable service composition for very-large-scale Internet of Things systems. *Proceedings of the Workshop on Posters and Demos Track.*

6. Joseph Benharosh. What is REST API? in plain English. *Describe on his blog www.phpenthusiast.com at July 31, 2018.*

Appendix A

SOURCE CODE

```
    def create_client_file(self,
                           type,
                           owner,
                           ssid,
                           password,
                           device_name,
                           longitude,
                           latitude):

        if(type == ``digital_sensor"):
            conn_file = self.digital_sensor()
        elif(type ==``analog_sensor"):
            conn_file = self.analog_sensor()
        elif(type == ``ir_sensor"):
            conn_file = self.ir_sensor()
        elif(type == ``ultrasonic_sensor_hc_sr04"):
            conn_file = self.ultrasonic_sensor_hc_sr04()
        elif(type == ``soil_moisture_sensor"):
            conn_file = self.soil_moisture_sensor()
        elif(type == ``actuator"):
```

```
            conn_file = self.actuator_default_file()

        folder_path = self.create_dir(device_name)
        print(''After Path: '')

        print(''Owner: ''+owner)
        print('\n')
        print(''SSID: ''+ssid)
        print('\n')
        print(''PASSWORD: ''+password)
        print('\n')
        print(''DEVICE: ''+device_name)
        print('\n')
        print(''Logitude: ''+longitude)
        print('\n')
        print(''Latitude: ''+latitude)
        print('\n')

        for line in conn_file:
            line = line.replace(''owner'',owner)
            line = line.replace(''wifi_name'',ssid)
            line = line.replace(''wifi_password'',password)
            line = line.replace(''device_name'',device_name)
            line = line.replace(''sensor_long'',longitude)
            line = line.replace(''sensor_lati'',latitude)

            with open(os.path.join(folder_path,(device_name+''.ino'')),
                    ''a'') as new_file:
                new_file.write(line)
```

Listing A.1: IoT device customize code

```
void create_tag()
{   sensor_tag = '' '';
    sensor_tag += sensor_name;
```

```
    sensor_tag +="_";
    sensor_tag += longitude;
    sensor_tag +="_";
    sensor_tag += latitude;
    sensor_tag +="_";
    sensor_tag += owner;
    Serial.print("Sensor Tag:");
    Serial.println(sensor_tag);
}
```

Listing A.2: Device tag creating Program

```
    def upload_command(self,sensor_name):
        print('*******Upload********')
        command1= 'arduino-cli compile --fqbn esp8266:esp8266:nodemcu'
        command1 += ' '
        command1 += os.getcwd()
        command1 += '/sketchbook/'
        command1 += sensor_name
        print('Command-1: '+command1)

        command2 = 'arduino-cli upload -p /dev/ttyUSB* --fqbn'
        command2 += ' '
        command2 += 'esp8266:esp8266:nodemcu '
        command2 += os.getcwd()
        command2 += '/sketchbook/'
        command2 += sensor_name
        print('Command-2: '+command2)

        cmd = [command1,command2]

        for c in cmd:
            self.terminal_task(c)
            print('****Command****')
```

```
23          print(``END")
```

Listing A.3: Terminal Agent Upload Command

```
1 void service_registry_json()
2 {
3   registry_encoder[``name"] = sensor_tag;
4   registry_encoder[``service_type"]= ``sensor";
5   registry_encoder.prettyPrintTo(service_registry_buffer,sizeof(
     service_registry_buffer));
6   Serial.println(service_registry_buffer);
7 }
```

Listing A.4: Sensor tag as a json data code

```
1 class ServiceRegistryApi(viewsets.ModelViewSet):
2     queryset = Service_registry.objects.all()
3     serializer_class = ServiceRegistrySerializer
```

Listing A.5: Service registry code

```
1 class LowerSensorApi(viewsets.ModelViewSet):
2   queryset = LowerSensor.objects.all()
3   serializer_class = LowerSensorSerializer
```

Listing A.6: Sensor data inserting code

```
1 class ActuatorApi(viewsets.ModelViewSet):
2   queryset = Actuator.objects.all()
3   serializer_class = ActuatorSerializer
```

Listing A.7: Actuator data inserting code