Declaration

This is certified that this project is an original work and is done by us. Neither it nor part of it has been submitted elsewhere for the requirement of any degree or diploma or for any other purposes.

_____

Md. Aumit Hasan

ID: 142-35-738

Department of
Software Engineering

Daffodil International University

_____

Farhadur Raja Fahim

ID: 141-35-647

Department of
Software Engineering

Daffodil International University

Letter of Acceptance

This project is entitled "**Distributed Simple Reflex Agent Over IoT**" submitted by Md Aumit Hasan (142-35-738) and Farhadur Raja Fahim (141-35-647) to the department of Software Engineering, Daffodil International University, Dhaka, Bangladesh is accepted by the department in partial fulfillment of requirements for the degree of Bachelor of Science in Software Engineering on April, 2018.

Board of Examiners

Supervisor

_____

K.M. Imtiaz Uddin
Assistant Professor
Department of Software Engineering
Daffodil International University

Abstract

People are using IoT devices their everyday life at home, working place and everywhere. Everyday field of IoT is increasing as well as devices and users. Researchers are trying to create new dimension in the field of internet of things. Every year the number of devices are increasing be 15% to 20%. Expert says that within 2020 the connected devices over IoT will be thirty billion, which is the four times larger than the total human being in the world. Many multinational organization are creating new service for users but that cannot fulfill the demand of people because it is not possible for organizations to figure out all types of services for users. The basic concept of this research project is to isolate the sensors, actuators and logic using distributed architecture and using this distributed sensors and actuators user will create their required services using web platform. Every part of the system are independent. If one part cause any damage the other parts will work without any error. This paper will contribute to give a basic idea how end user can create their own service using distributed sensors and actuators using web application.

Contents

List of Figures

Chapter 1

1. Introduction:

Everyday IoT field is increasing as well as devices. Most of the devices are individual system with functionality. If any part cause damaged then the whole system affects. . It's a real challenge to manage those devices and enormous data individually. Also the user has less power to create composite service when needed.

That's why we want to build a loosely coupled system where sensors and actuators and agent function will be distributed. In traditional system different service provider provides a specific service with predefined functionality. User will get more freedom to create different services using different sensors and actuators. User will be able to create composite services using a easily understandable user interface.

This chapter introduces our motivation to do this research in Section 1.1. Section 1.2 describe the objective of this research. Section 1.3 will give a short overview of our proposed system. And Section 1.4 gives the outline of the whole documents.

1.1 Motivation

One of the general goal of IoT is to give computation power to every object and gives intelligence to them. Most of the devices of IoT working tightly coupled with sensor, actuator and logic. They works as a single unit and aren't independent. If any sensor doesn't work well the whole system get affected. Because that part depends on others.

User cannot alternate the sensor with others as if the service provider give the option. But most of the time they don't give any option like that. It will be the same scenario for other parts of the system. One of the main reason for this, to hold the power to them and get rid of maintenance and other problems. We had thought to provide a simple solution for this kind of problem.

Some groups from IoT industry are working with loosely coupled distributed system but they do not give the full independence to the end user. Also these service very complex to understand for end user. But our proposed architecture is working as a simple reflex agent and very easy to understand for end user.

From the very beginning of our research we were fully concern about end user and their independence as they can easily interact with a system without having any kind of knowledge of programming. This motivate us to build such an easily understandable user interface.

1.2 Objectives

We want to build a system that will give end user the ultimate power to build their required services. They can change the system requirement when arises different needs. They can also change the working process. But it is not very easy for user to understand the process of changing and updating the working process when we think about the traditional IoT devices. But we have tried to solve these issue in this work.

We want to provide an easy interface to the end user. In our proposed system we also have built a very user friendly interface for end user as they can use them easily.

Traditional IOT devices works with sensors and actuators with logic as a tightly coupled system Sometimes a sensor or actuator use for a few purpose that are predefined. Normal user cannot change any process. But in our proposed system we want to provide sensors and actuators that are independent. End user can use them to create required service for multiple purpose. Nothing are predefined by us and all power holds for end user.

1.3 Overview

Our system is an IoT Application that is a combination of Android devices and Web Application. The system has two major parts. One is collect data using distributed sensors from devices then design different services using web application.

We have an Android app to collect data from user. So far we have worked with two sensors. One is Global Positioning System (GPS) and another is Accelerometer. Through this android app Global Positioning System (GPS) and Accelerometer data will be passed to database. Initially we are using Firebase as Database. User will install the app and they are done with this part. Maintaining the database is not the area for end user. We the developer will handle the database.

Now user have to use the web application. The web application will help them to create their required service using web interface. To use the system we create an authentication system that will specify the user and will provider their data. Using Sensor and Actuator service user will create different types of composite services.

So far we have worked with two actuators. One is Email Service and another is SMS Service. Based on user condition system will send notification.

1.4 Outline of the Thesis

Chapter 2 presents the beginning works of IoT and works thats are related to our propose system. It also discuss about IoT architecture, Simple Reflex Agent, related technology and IoT Frameworks.

Chapter 3 describes the proposed system architecture and how it works.

Chapter 4 discusses proof of concept that contains each and every part of the proposed system with code reference. It also shows the final output of this work. In this section we describe some required topics too.

Chapter 5 it concludes the thesis work. It contains our contribution, limitation of this thesis and predicts the future work of this thesis.

Chapter 2

2. Literature Review

Kevin Ashton co-founder of MIT's Auto-ID Center first uses the term "**Internet of Things**" in 1999 [1] [7] to describe a system where the physical world is connected with Internet via Ubiquitous Sensor [2] [3]. Ashton's idea of IoT focused on using Radio Frequency Identification (RFID) [16] to gain object information that is presorted electric information.

At very beginning using RFID Technology, GPS, Laser Scanner had used to smart identification, localization, track object, management and scheduling like Cargo Tracking System [7] and radio wave used for real time work [8].

Year 2000 was a milestone for IoT. LG first made world's first Internet-connected refrigerator had featuring IP connectivity and LAN port [6]. Industry was the most benefited sector other than many sector. Tracking enables companies to become efficient, reducing error, prevent theft and making the business fast [2].

IoT had got a huge push when Cloud and IoT start to work together. Cloud made the modern IoT possible and it will be also played a important role in future [6]. When working with cloud developer and researchers faced many critical issues like expanding the network's capacity [6]. Because in 2017 we have 8.4 Billion devices are activated and connected with clout and IoT [17].

IoT has connected real life and physical activities with virtual world [10]. It connects human-human, human-thing, thing-thing [11]. Dealing with those tremendous amounts of device and cloud IoT faces privacy and security challenges [12]. It's a big issue because this device collects our everyday life information and monitor us [6]. IoT connects almost everything at our society [9]. But security was not a big deal and thinking at early stage [11]. These problems happens not only for technical problem but also related with infrastructure, planning, security and other aspects problems [12]. But it is hopeful that it is becoming intelligent day-by-day [12].

Cloud is a big challenge with manage huge amount of storage. In IoT device collect data using sensors and passes them using wired or wireless transmission network. It is a big challenge to handle large number of data and protect them from data loss when transferring [13].

The architecture of IoT is based on data communication but at early stage it worked with RFID tagged items and basic identification. But at present days the definition of architecture of IoT has changed. Past days it worked as tightly coupled architecture as sensor, actuator, and agent program are work with together and they depended on each other. But today is the era of distributed architecture and they works as loosely coupled [5]. Currently IoT has three layers are [15]

- Application Layer

- Network Layer
- Perception Layer

We can control devices using other devices and transfer sensor data and actuator data [5]. But at the site of end user there don't have enough work. In this paper we are working to build a user friendly interface using that end user can create composite services.

"The IoT is getting bigger and bigger every year but we are still in the very early stages of this technology" - Jason Aiginitis.

2.1 Related Works

We review the present situation of IoT according to distributed architecture, simple reflex agent and End User development but we don't find any work that uses distributed sensors, actuators and made different permutation that create composite service easily as ours. But few works have done about distributed IoT architecture, controlling device using another device, loosely coupled distributed sensors and actuator. Also we found few works that working for self-adaptive service, end user programming and their independence to create their own service. It is important to control those devices by end user [19].

In this thesis we are expanding a previous group's work [5]. They create a distributed architecture isolating sensors, actuators and logic. They showed that using this loosely coupled sensors actuators and logic can work together through cloud and can control devices, take input and give output.

A group have worked at the area of self-adaptive service that take requirements and change the service at runtime [21]. This work is related with our work as it review the end user service composition and change service. This group shows that End User control over service composition and automatic adaptation are complementary contest [21].

Few years ago at the site of End User independency there have a very few works. But those days has changed. Recently a work have done that provides end user three building blocks to support end user configuration using sensor actuator and provides recommendation [20]. Some workshop was arranged to share researcher's knowledge and experience to deal with end user side development [18].

2.2 Related Topic and Framework

2.2.1 Distributed Agent Architecture

In distributed architecture sensors, actuators and logic are isolated. They are loosely coupled. Every part is independent. Sensors depends only on itself. If any part of actuator causes any damage it cannot affect sensors.

Our proposed system has followed this architecture. Our sensors situated at android phone, actuators land logic works through cloud. We have successfully made our system as distributed.

2.2.2 Simple Reflex Agent

In artificial intelligence, an intelligent agent is an autonomous entity which observes through sensors and changes environment using actuator depends on its goal and rationality [25]. Intelligent agents may be very simple and very complex. Normally agents are five types [25] [26].

- Simple Reflex Agent
- Model-based Reflex Agents
- Goal-based Agents
- Utility-based Agents
- Learning Agents

Simple Reflex agent acts according to current percept and gives result using IF-ELSE rules [26]. In our thesis we have designed a simple reflex agent that use distributed sensors, actuators and logic.

It is a Simple Reflex Agent because when we use GPS sensor we define that if current location of device is in/out of a range give a notification. It needn't any previous data or to give output it doesn't depends on any previous stored data. That's why it's Simple Reflex Agent [25].

2.2.3 End User Service Composition

Most of the end user are normal people don't have enough IT knowledge. They just use any service over online or offline they needed. So if someone think that end user will create their own service then it becomes a big deal.

When we are working with end user programming we have designed this part as visualization technique that enables to drag and drop services and combine them. User will drag and drop needed block to the logic function provided by us and the work will be done

2.2.4 Dracula.js Library

Dracula.js is a combination of tools that display and create graphical layout of graphs and networks, here has used different algorithm from graph theory.

This had released under the MIT license, so everyone can use this. It's a free service. Anything like graphical presentation it can be easily used.

2.2.5 IoT Framework

2.2.5.1 IBM BlueMix

IBM BlueMix is a cloud server based service that helps to building running and managing apps and services.

BlueMix has different services and interfaces. Many IoT working related people use this for different purpose. It allows to select apps, data and created different types of graph using the collected data by user. The data are stored at their cloud server.

BlueMix provide excellent security support as they use different protocol to send data and collect data. It use enterprise APIs. It provide space flexibility. Also students are getting free access to paid material and space here.

2.2.5.2 AWS IoT

AWS IoT is an Amazon Product. AWS IoT allows to connect device with different service and alternating data. It helps developer many flexibility to develop IoT services.

AWS IoT provides SDK that helps to collect data and hosting and running the application. They can make analysis using data.

2.2.5.3 Azure IoT

Azure IoT Suite is a product of Microsoft. It is designed to make understandable the business value using IoT. It helps to automate the business using IoT. Predefined solution can be used easily. It takes care of task deploying and various services to give a complete end to end solution. The SDK will help to do different tasks.

2.2.5.4 Cloud IoT Core

Cloud IoT core is provided by Google. It allows to connect, manage and ingest data from millions of globally deployed services. Cloud IoT is a combination of IoT platform, providing business solution, collecting processing analyzing and visualizing data.

Chapter 3

3. Description of Proposed System

3.1 Required Concept

3.1.1 Sensor service

Sensor is an input device, module or subsystem which detects event or changes in its environment and send the information to the predefined system or computer processor. The system will process the input signal as human readable data. Different types of sensor are available to use like

- Thermal Sensor
- Electromagnetic Sensor
- Mechanical Sensor
- Motion Sensor

Now-a-days mobile phone contains many sensor according to phone types. In this work we have used sensor from mobile as it's easy to collect data from mobile and user doesn't need other sensor device.

We have used Accelerometer and Global Positioning System (GPS) to take input.

Accelerometer is an electromagnetic device that measures acceleration which is rate of changes of velocity of an object. It is an electromechanical device that senses static of dynamic forces acceleration. Static forces includes gravity and dynamic forces includes vibration and movement. When an object changes its position at any axis accelerometer can measure the changes.

Global Positioning System (GPS) is satellite based navigation system made by minimum 24 satellites. GPS works at any condition at any place in the world. The U.S. Department of Defense put the satellite in the orbit and later they made available this for civilians.

At any place in the world four satellites are visible and each one transmit information about its position and time at regular interval. This signals are traveling at light speed. When these signals are received by GPS receiver, calculate the distance of the satellites. Using this information GPS can pinpoint the exact location using a process called trilateration.

3.1.2 Actuator Service

When a machine, robot or computer program works it makes decisions according to the design or purpose. After making a final decision it has to do the work to change the environment and it's done by the actuator.

In a computer program print instruction is the actuator.

3.1.3 Microservices

Microservices known as microservice architecture. It is an architectural style that construct an application is combination of loosely coupled services. These services are independent and can use at multiple purpose for the different work. This architecture enables continuous development of a large and complex applications like enterprise software and the software needed continuous development.

3.1.4 Composite Service

Composite service is made by different existing services to create complex services. It uses the microservices to achieve the working goal. These microservices can be web service, XML service or other services.

Different combination of microservice create new services. When someone create it he can specify the order of execution. The order can be Sequential or Concurrent.

3.1.5 Loosely Coupled Architecture

Loosely coupled architecture refers to a system that has different parts that are not only independent but also may not be with together. In IoT normally has three components are Sensor that enables to take percept and understand the environment, Logic that is like a brain made decision and Actuator that try to establish the decision. Every component are distributed over network and stayed as different permutation.

3.1.6 Tightly Coupled Architecture

Tightly coupled architecture refers to a system that's all component are highly depended on one another. Traditional IoT device are tightly coupled. Sensor, actuator and logic are stayed together. Any component damage affects the whole system.

3.2 Proposed System Architecture

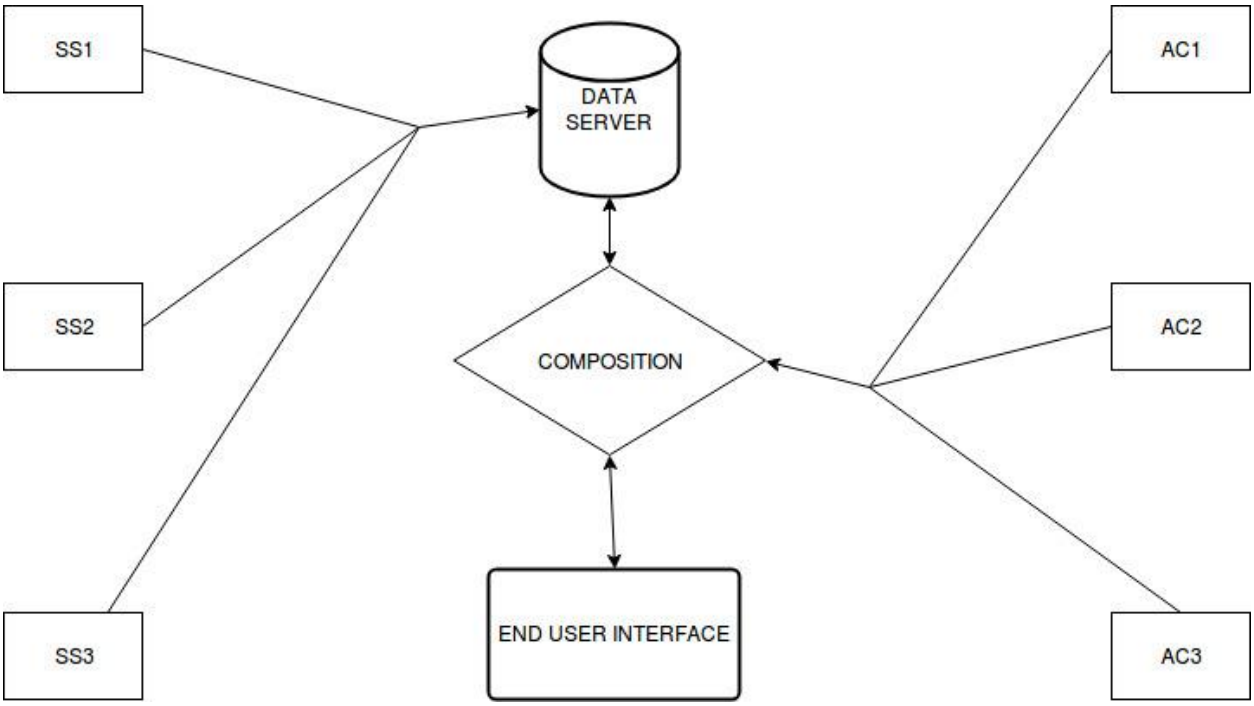

Figure: Architecture of Proposed System [24]

3.3 Mechanism

Internet of Things is a system where we have to isolate the sensors, actuators and logic. Here we are working on a distributed agent architecture means sensors, actuators and logic can be stored in different place or in same place but they will work together simultaneously. This paper proposed a design where a device can act as a sensor and actuator and an end user can make composite application as per their needs with sensor, actuator using a user friendly interface. Like an user will notify through sms or email service according to their GPS location. Our main goal is to provide end user an easy interface to create their required service using existing Sensor and Actuator Services. The system will collect data from user using different sensor.

3.3.1 Application Installation

To proof our concept we used an android application that will collect sensor data from android device. Any android OS device can perform this action. We build an android application to collect data. Initially we work on GPS and Accelerometer sensor.

So to perform action with this application GPS and accelerometer sensor required in android device. To install this application simply copy the apk in android devices and install the application in the device.

3.3.2 Device Registration

After successfully install registration will need to send and receive data to a database.
We used here simple registration with an email address.

After registered a unique key will generated for every registered user. This key will used as a user id.

We can build an own server to store data but for simplicity we used Firebase [23] for storing data.

3.3.3 Data Into Data Server

Now we have registered android device with a user id. The device will sent GPS and accelerometer sensor data into database after certain amount of time.

A user can give time difference for sending data into database. It will provide real time data publishing facility.

3.4 Protocol Used

Protocol are rules and regulation for a system. A system have single or multiple protocol. In our system we have one called Hypertext Transfer Protocol (HTTP) [22].

Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. In our system HTTP used for sending request from device with a user id and server responded with HTTP protocol.

3.5 Server Side Application

In server side of our system we have firebase application to saved data that comes from device. The database saved data with a used id so that we can separate data for different user. We write some rules for that application.

The server responded according to this rules. This will saved data that comes from devices and serve data based on logic that gives by user in user application.

3.6 Web Application

End-user is a big concern for our system. An end-user can easily access, customize, and create composite application according to their needs through our system. So we try to build a user friendly web application so that a user can easily interact with our system.

In that web service user visualize sensors, actuator and provide logic based on that the system will show the result.

The user interface is very important part of our work. Our main goal is to control distributed sensors and actuators from one place creating services when needed.

User can access his sensor data and using those he can design an agent program for decision making. The web platform has different actuator to do specific works. After making any decision and to achieve the decision goal user can use the actuators when needed.

We have developed a quiet simple user interface to prove our concept.

Chapter 4

4.1 Proof of Concept Implementation

4.2 Application Details

The web application or platform is very important part of our work. Our main goal is to control distributed sensors and actuators from one place creating services when needed. User can access his sensor data and using those he can design an agent program for decision making. The web platform has different actuator to do specific works. After making any decision and to achieve the decision goal user can use the actuators when needed.

We have developed a quiet simple interface to prove our concept.

At this platform user can create simple reflex agent based on the current sensor data. At present we have two sensors and two actuators.

First of all the agent program has two blocks.

1. If Block
2. Else Block

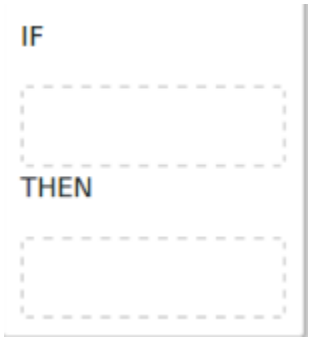

Figure 4.2.1

If Block: If block contains all logics. At this step user have to decide which sensor he will. User can select GPS or Accelerometer.

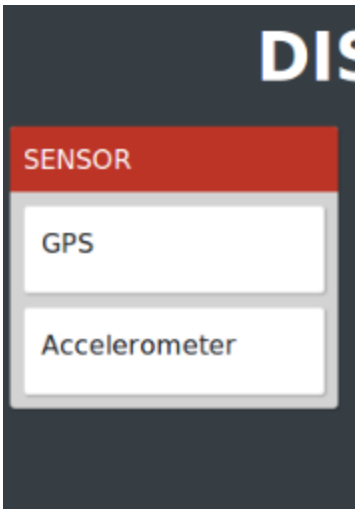

Figure 4.2.2

In figure 4.2.2 If user select GPS then have to drag in the If block. Now the system needs some information to process the program.

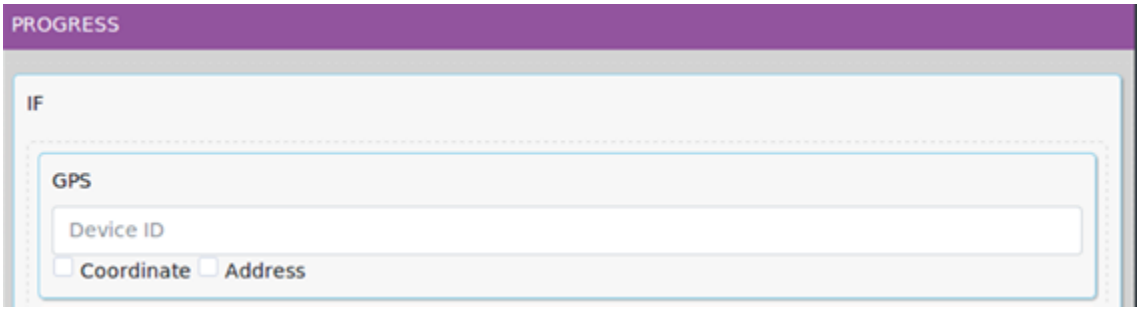

Figure 4.2.3

In figure 4.2.3 When the Android app has installed it generates a unique Device ID. To know the position of the device user must put a valid Device ID. This protects the privacy of every user. If anyone don't want to track by someone then he has to share his Device Id otherwise it isn't possible to track him.

After provide a valid Device Id user has to decide whether he want the device location according to Coordinate or Address. If he select Coordinate then the system needs to know the coordinate (Latitude and Longitude) of a place. For Address he has to provide an Address of a specific location. According to this location the system will give a result whether the device is within an area or not.

Now user has select the Operator from the right sight and put at IF Block after GPS. Here user has to provide the distance. If he want to check the device within a range then has to select the Less Than (<) operator. Also he can select Greater Than (>), Equal (=).

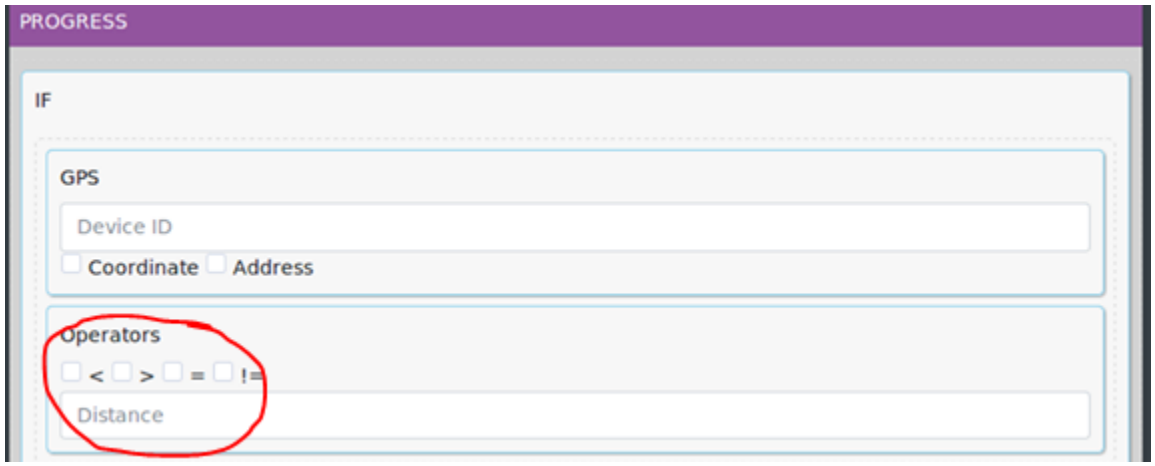

Figure 4.3.4

At this step user has completed the agent function. Now user has to work at the Else block. It is the area where user has to put the Actuator.

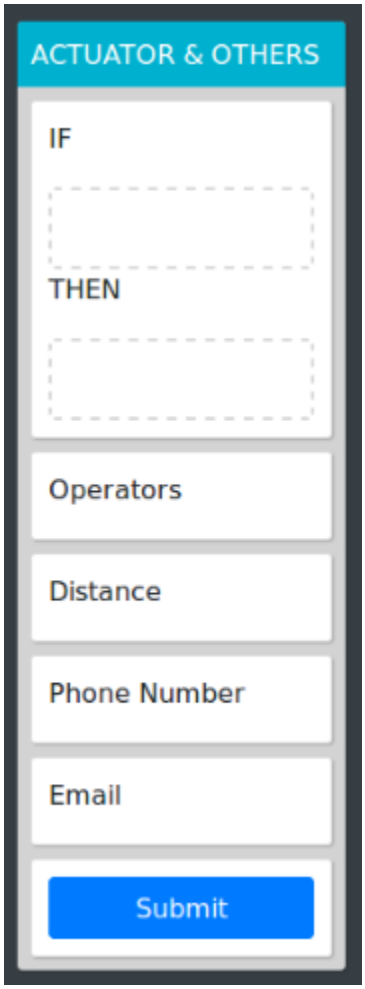

Figure 4.3.5

Now he has to decide which actuator he want to use. He can use Phone Number of Email Microservice to get the result weather his targeted device within his defined range or not.

If he select Phone Number and drag this at Else block then he will get two field to fill up. User will give the phone number to get notification and the Text field needs the massage.

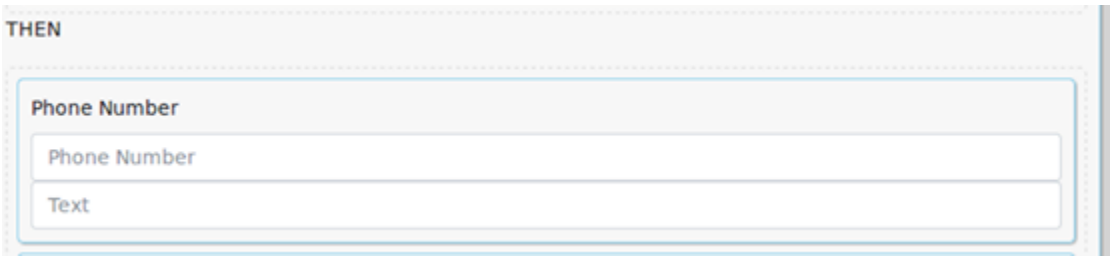

Figure 4.2.6

If user select Email Service then he has to provide an email and the text massage.

This is the final step. Let's have a look the final agent program. User has to submit now.

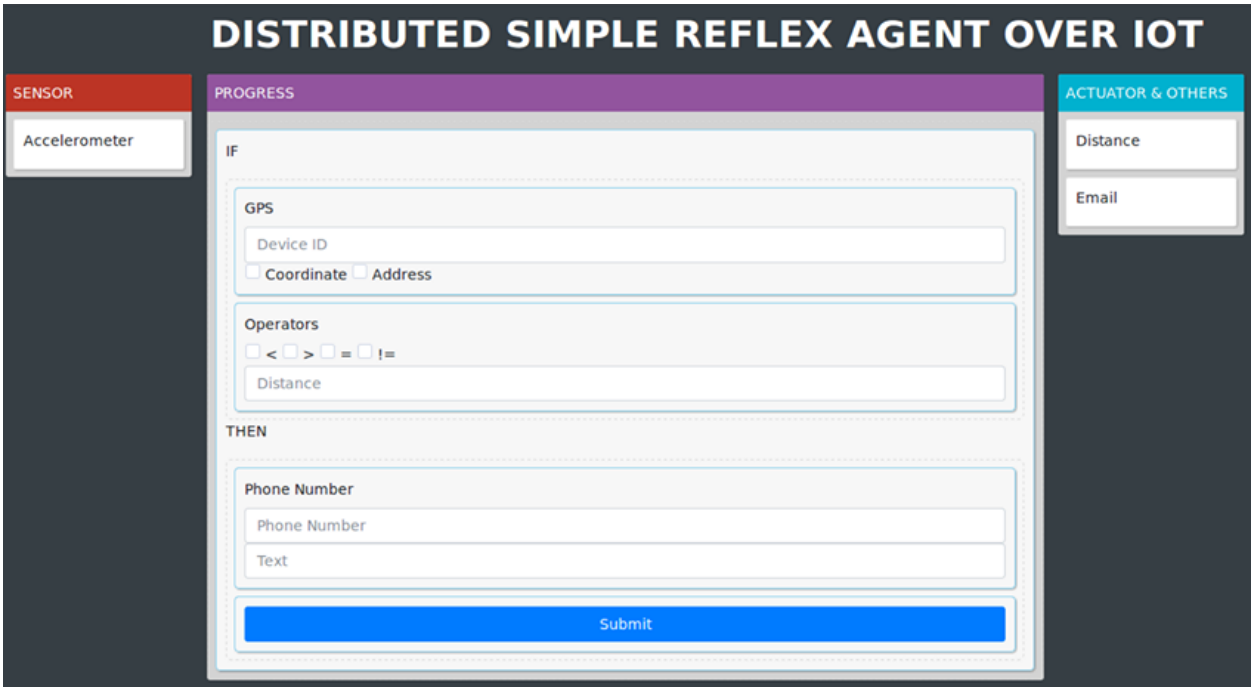

Figure 4.2.6

Here is the device location. After submit the program agent function will calculate weather the device is within a range or no.

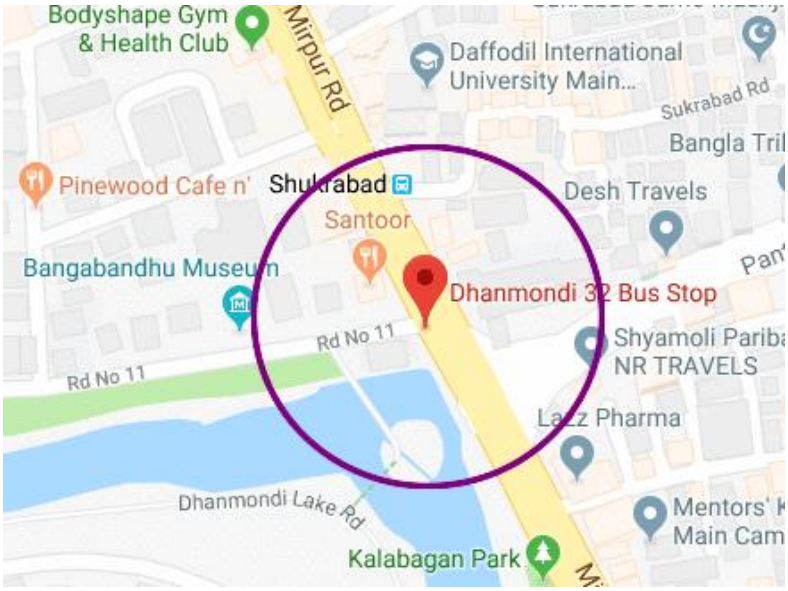

Figure 4.3.7

If the device isn't within a range the he will get a notification via the actuator.

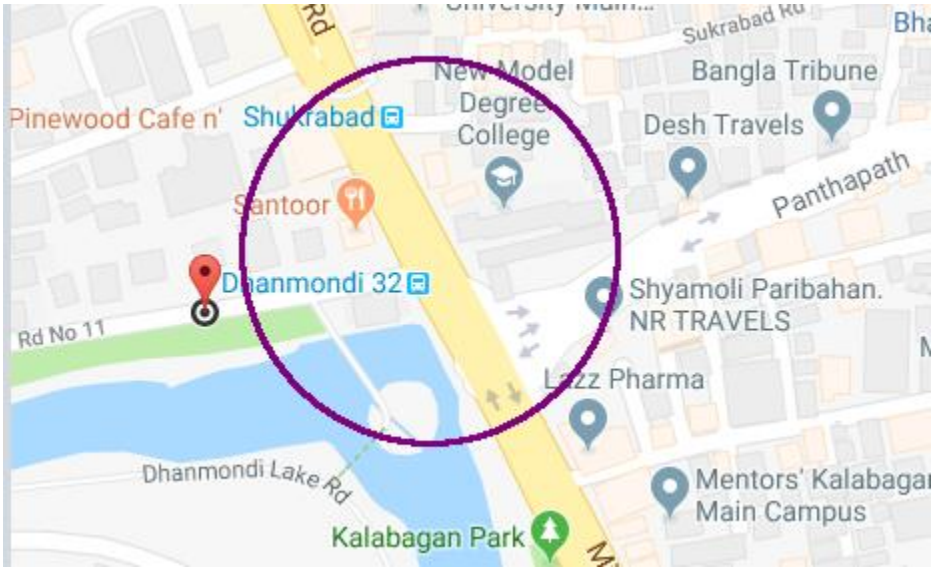

Figure 4.2.8

Chapter 5

Conclusion

This chapter summarize our current work. Section 5.1 describe our contribution of the proposed system. Section 5.2 presents the limitation regarding Distributed Simple Reflex Agent Over IoT. Section 5.3 forecast about our future work.

5.1 Contributions

Through the proposed system End user can control distributed sensors, actuators and logic. End User can create their required service related to these sensors and actuators easily using our proposed system's user friendly interface.

We have motivated to do this work by a group of students from EWU [5]. They have also used distributed architecture where sensors, actuators and logic is isolated and showed that devices would be controlled using another device to take input and give output through sensors and actuators.

But they [5] don't focus on end user programming and service composition. We create significance working on this part. Our proposed system can control distributed sensors and actuators as like [5]. But giving the power of service composition to end user we have made a new dimension.

5.2 Limitations of this System

Our system can interact with only one user. The android app that we are using to collect data using phone sensors is not well designed. It needs some modifications. GPS and Accelerometer cannot work in a logic function together. We have used server as Firebase Database but if we can build our own web server it will be more secure and easily usable for us because each and every rules would be defined by us. We have used a free service to send Notification through Actuators (Email and Phone Text Option) that sends limited number of notification and sometimes it gets blocked.

5.3 Future Works

At present our proposed system can deal with only a single user but in future we want to make capable of handling multiple users and give security to their personal data collected by Sensors.

We will try to make the process of add microservices (Sensors and Actuator Service) more dynamic. It will be pluggable. Any developer can add a pluggable service and every user can use that. It will be something like app store.

Reference

[1] Kevin Ashton, "That 'Internet of Things' Thing", RFID Journal, 22 June 2009.

[2] Goetz, Thomas (19 June 2011). "Harnessing the Power of Feedback Loops". Wired.

[3] https://en.wikipedia.org/wiki/Kevin_Ashton#cite_note-RFID_Journal_2009-3

[4] Ferguson, T. (2002) Have Your Objects Call My Object. Harvard Business Review, June, 1-7.

[5] Fahad, Md. Junaid; Shafiullah, Mohammad, (2016), Distributed Agent Architecture Using Internet of Things, Unpublished Undergraduate Thesis, East West University, Bangladesh.

[6] Christopher Tozzi, (2016), IoT Past and Present: The History of IoT, and Where It's Headed Today, Channel Futures

[7] Lixin Zhou, Catherine Xiaocui Lou (2012), Intelligent Cargo Tracking System Based on the Internet of Things. 15th International Conference on Network-Based Information Systems. Page 489-493, 978-0-7695-4779-4/12 © 2012 IEEE.

[8] Xiaolin Jia, Quanyuan Feng, Taihua Fan, Quanshui Lei, (2012). RFID technology and its applications in Internet of Things(IoT), IEEE Xplore 2012, Consumer Electronics, Communications and Networks (CECNet), Yichang, China.

[9] J. Zheng, D. Simplot-Ryl, C. Bisdikian, and H. Mouftah, "The Internet of Things," in IEEE Communications Magazine, Volume:49 , Issue: 11, pp:30-31, 2011.

[10] Y. Huang and G. Li, "A Semantic Analysis for Internet of Things," in International Conference on Intelligent Computation Technology and Automation (ICICTA), May 2010.

[11] L. Tan and N. Wang, "Future Internet: The Internet of Things," in 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), August 2010.

[12] G. Gang, L. Zeyong, and J. Jun, "Internet of Things Security Analysis," in International Conference on Internet Technology and Applications (iTAP), August 2011.

[13] Rafiullah Khan, Sarmad Ullah Khan."Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges". 10th International Conference on FIT, 2012.

[14] Y. Huang and G. Li, "Descriptive Models for Internet of Things," in IEEE International Conference on Intelligent Control and Information Processing (ICICIP), August 2010.

[15] M. Wu, T. Lu, F. Ling, J. Sun, and H. Du, "Research on the Architecture of Internet of Things," in 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), Aug. 2010.

[16] X.L.Jia,Q.Y.Feng, C.Z.Ma,"An efficient anti-collision protocol for RFID tag identification," IEEE Communications Letters, vol.14, no.11, pp.1014-1016, 2010. (RFID)

[17]https://en.wikipedia.org/wiki/Internet_of_things#cite_note-mohammadi2018deep-108

[18] Daniel Tetteroo, Panos Markopoulos (2015), End-User Development in the Internet of Things Era, Seoul, Republic of Korea ACM 978-1-4503-3146-3/15/04

[19] Margaret Burnett, Todd Kulesza (2015) End-User Development in Internet of Things: We the People, In International Reports on Socio-Informatics (IRSI), (Vol. 12, Iss. 2, pp. 81-86)

[20] Matúš Tomlein, Sudershan Boovaraghavan (2017), CharIoT: An end-user programming environment for the IoT, Carnegie Mellon University (https://doi.org/10.1145/3131542.3140261).

[21] K. M. Imtiaz-Ud-Din,Dr. Mohammad Ullah Khan (2014), Runtime Adaptation of End-User Composed Collaborative Services, WETICE Conference (WETICE), 2014 IEEE 23rd International.

[22] https://www.w3.org/Protocols/

[23] https://firebase.google.com/docs/

[24] https://www.draw.io/

[25] Russell, Stuart J.; Norvig, Peter (2003), Artificial Intelligence: A Modern Approach (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2, chpt. 2.

Appendix

Backend functionality

```
from flask import Flask, render_template, request

app = Flask(__name__)


@app.route('/')
def return_form():
    return render_template('index.html')


@app.route('/data', methods=['POST', 'GET'])
def get_data():
    from firebase import firebase
    from geopy.geocoders import Nominatim
    from math import sin, cos, sqrt, atan2, radians
    geolocator = Nominatim()
    fb = firebase.FirebaseApplication(
        'https://gpslocation-93d6a.firebaseio.com/Locations/')

    userid = request.form['userid']
    R = 6373.0*1000
    current_location = request.form['current_location']
    location_address = geolocator.geocode(current_location)
    data = fb.get(userid, None)
    lat1 = radians(float(data['latitude'][10:]))
    lon1 = radians(float(data['longitude'][11:]))
    lat2 = radians(location_address.latitude)
    lon2 = radians(location_address.longitude)
    dlat = lat1 - lat2
    dlon = lon1 - lon2
    a = sin(dlat/2)**2 + cos(lat1)*cos(lat2)*sin(dlon/2)**2
    c = 2*atan2(sqrt(a), sqrt(1-a))
    distance = R*c
    user_distance = request.form['distance']
    if(float(user_distance) < float(distance)):
            check = "User is out of range"
    else:
            check = "User in distance"

    loc = data['latitude'][10:] + ", " + data['longitude'][11:]
    last_location = geolocator.reverse(loc)
```

```
    Return render_template('index_data.html', data=last_location[0], data2=check)



@app.route('/sms', methods=['POST', 'GET'])
def send_sms():
    from twilio.rest import Client
    # MY_ACCOUNT_SID
    account = "test_account"
    # auth token
    token = "test_token"
    client = Client(account, token)
    tonumber = request.form['number']
    fromnumber = "test_number"
    message_body = request.form['message']
    message=client.messages.create(to=tonumber, from_="test_number", body=message_body)
    return render_template('success.html', data=message_body)


if __name__ == '__main__':
    app.run(debug=True)
```

Home page where user create composite system as their needs

```
<!DOCTYPE html>
<html>

  <head>
    <title>IoT Project</title>
    <script type="text/javascript" src="static/js/jquery-3.3.1.min.js"></script>
    <script src="static/js/dragula.min.js"></script>
    <script src="static/js/script.js"></script>
    <link rel="stylesheet" href="static/css/bootstrap.min.css"/>
    <link rel="stylesheet" href="static/css/style.css"/>
  </head>

  <body>
    <h1>Distributed Simple Reflex Agent Over IoT</h1>
    <div id="kaban">
      <div id="scroller">
        <div id="boards">
          <div class="board" id="board1">
```

```
          <header>Sensor</header>
          <div class="cards" id="b1">
            <div class="card">
              <label for="usr">GPS</label>
              <input type="text" class="form-control" placeholder="USER ID"
name="userid">
              <span >
                <input type="checkbox" id="geoCoordinate" onclick="coordinateFunction()">
Coordinate
                <input type="checkbox" id="geoLocation" onclick="locationFunction()"> Address
              </span>
              <input id="latitude" style="display:none" type="text" class="form-control"
placeholder="Latitude" name="latitude">
              <input id="longitude" style="display:none" type="text" class="form-control"
placeholder="Longitude" name="longitude">
              <input id="address" style="display:none" type="text" class="form-control"
placeholder="Location" name="current_location">
            </div>
            <div class="card">
              <label class="cardtitle noselect">Accelerometer</label>
              <input type="text" class="form-control" placeholder="Accelerometer"
name="accelerometer">
            </div>
          </div>
        </div>

        <div class="board" id="board2">
          <header>Progress</header>
          <form class="cards playground" id="b2" action="{{ url_for('get_data') }}"
method="POST">

            </form>
            <!-- <div class="cards playground" id="b2">

            </div> -->
        </div>
        <div class="board" id="board3">
          <header>Actuator & Others</header>
          <div class="cards" id="b3">
            <div class="card form-group">
              <label>IF</label>
              <div class="cards playground hide-con" id="b4">

              </div>
              <label>THEN</label>
              <div class="cards playground hide-con" id="b5">
```

```
            </div>
          </div>
          <div class="card form-group">
            <label for="usr">Operators</label>
            <span class="radio-button">
              <input type="checkbox" id="less-than" onclick="changeCheckBox1()"> LESS
THEN(<)
              <input type="checkbox" id="gretter-than" onclick="changeCheckBox2()">
GRETTER THEN(>)
              <input type="text" class="form-control" placeholder="Distance"
name="distance">
            </span>
          </div>
          <div class="card form-group">
            <label for="usr">Phone Number</label>
            <input type="text" class="form-control" placeholder="Phone Number"
name="number">
            <input type="text" class="form-control" placeholder="Text" name="message">
          </div>
          <div class="card form-group">
            <label for="usr">Email</label>
            <input type="text" class="form-control" placeholder="Email Addres"
name="email">
            <input type="text" class="form-control" placeholder="Email Text" name="email-
body">
          </div>
          <!-- <div class="card">
            <input type="submit" class="btn btn-primary" name="send" value="SEND">
          </div> -->
          <div class="card">
            <button type="submit" value="Submit" class="btn btn-primary">Submit</button>
            </div>
        </div>

      </div>
    </div>
  </div>
  </div>
  <script>
    function coordinateFunction() {
      var geoCoordinate = document.getElementById("geoCoordinate");
      var geoLocation = document.getElementById("geoLocation");
      var latitude = document.getElementById("latitude");
      var longitude = document.getElementById("longitude");
      var address = document.getElementById("address");
```

```
  if (geoCoordinate.checked == true){
      latitude.style.display = "block";
      longitude.style.display = "block";
      address.style.display = "none";
      geoLocation.checked = false;
  } else {
      longitude.style.display = "none";
      latitude.style.display = "none";
      address.style.display = "none";
  }
}
function locationFunction() {
    var geoCoordinate = document.getElementById("geoCoordinate");
    var geoLocation = document.getElementById("geoLocation");
    var latitude = document.getElementById("latitude");
    var longitude = document.getElementById("longitude");
    var address = document.getElementById("address");
    if (geoLocation.checked == true){
      address.style.display = "block";
      latitude.style.display = "none";
      longitude.style.display = "none";
      geoCoordinate.checked = false;
  } else {
      longitude.style.display = "none";
      latitude.style.display = "none";
      address.style.display = "none";
    }
}
function changeCheckBox1() {
  document.getElementById("less-than").checked = true;
  document.getElementById("gretter-than").checked = false;
}
function changeCheckBox2() {
  document.getElementById("less-than").checked = false;
  document.getElementById("gretter-than").checked = true;
}

dragula([
  document.getElementById('b1'),
  document.getElementById('b2'),
  document.getElementById('b3'),
  document.getElementById('b4'),
  document.getElementById('b5'),
])
.on('over', function (el) {
  el.className += ' ex-over';
```

```
    }).on('out', function (el) {
      el.className = el.className.replace('ex-over', '');
    });

    // Scrollable area
    var element = document.getElementById("boards"); // Count Boards
    var numberOfBoards = element.getElementsByClassName('board').length;
    var boardsWidth = numberOfBoards * 316 // Width of all Boards

    // disable text-selection
    function disableselect(e) { return false; }
    document.onselectstart = new Function()
    document.onmousedown = disableselect
  </script>
 </body>

</html>
```

Show result page

```
<!DOCTYPE html>
<html>
<head>
    <title>IoT Project</title>
    <script type="text/javascript" src="static/js/jquery-3.3.1.min.js"></script>
    <script src="static/js/dragula.min.js"></script>
    <script src="static/js/script.js"></script>
    <link rel="stylesheet" href="static/css/bootstrap.min.css"/>
    <link rel="stylesheet" href="static/css/style.css"/>
  </head>

 <body>
  <h1>Distributed Simple Reflex Agent Over IoT</h1>
  <div id="kaban">
    <div id="scroller">
      <div id="boards">
        <!-- <div class="board" id="board1">
          <header>Actuator & Sensor</header>
```

```
          <div class="cards" id="b1">
            <div class="card">
              <label for="usr">GPS:</label>
              <input type="text" class="form-control" placeholder="Latitude" name="latitude">
              <input type="text" class="form-control" placeholder="Longitude"
name="longitude">
            </div>
            <div class="card">
              <label for="usr">GPS:</label>
              <input type="text" class="form-control" placeholder="Location"
name="current_location">
            </div>
            <div class="card">
              <span class="cardtitle noselect">Accelerometer</span>
            </div>
          </div>
        </div> -->
        <div class="board" id="board2">
          <header>In Progress</header>
          <form class="cards playground" id="b2" action="{{ url_for('get_data') }}"
method="POST">
              <div class="card form-group">
                 <label>Last Location: </label>
                 <p>{{data}}</p>
              </div>
              <div class="card form-group">
                 <p>{{data2}}</p>
              </div>
          </form>
        </div>
        <!-- <div class="board" id="board3">
          <header>Others</header>
          <div class="cards" id="b3">
            <div class="card form-group">
              <label for="usr">Device ID:</label>
              <input type="text" class="form-control" name="userid">
            </div>
            <div class="card form-group">
              <label for="usr">Distance:</label>
              <input type="text" class="form-control" name="distance">
            </div>
            <div class="card form-group">
              <label for="usr">Phone Number:</label>
              <input type="text" class="form-control" name="number">
            </div>
            <div class="card form-group">
```

```
            <label for="usr">Email:</label>
            <input type="text" class="form-control" name="email">
          </div>
          <div class="card">
            <input type="submit" class="btn btn-primary" name="send" value="SEND">
          </div>
          <div class="card">
            <input type="submit" class="btn btn-primary" name="send" value="SUBMIT">
          </div>
        </div>

      </div> -->
    </div>
  </div>
 </div>
 <!-- <script>
  dragula([
    document.getElementById('b1'),
    document.getElementById('b2'),
    document.getElementById('b3')
  ])

  // Scrollable area
  var element = document.getElementById("boards"); // Count Boards
  var numberOfBoards = element.getElementsByClassName('board').length;
  var boardsWidth = numberOfBoards * 316 // Width of all Boards

  // disable text-selection
  function disableselect(e) { return false; }
  document.onselectstart = new Function()
  document.onmousedown = disableselect
 </script> -->
</body>

</html>
```