NOVATEUR PUBLICATIONS
INTERNATIONAL JOURNAL OF INNOVATIONS IN ENGINEERING RESEARCH AND TECHNOLOGY [IJIERT]
ISSN: 2394-3696
VOLUME 5, ISSUE 6, June -2018

RESTFUL COMMUNICATION APPROACH

JASMEET SINGH
University School of Information, Communication & Technology, GGSIPU New Delhi,India
jasmeetschanna@gmail.com

RAVINDRA KUMAR PURWAR
University School of Information, Communication & Technology, GGSIPU New Delhi,India
ravindra@ipu.ac.in

ABSTRACT

There are devices performing various functions and providing services to other devices. These devices provide output in various formats like xml,html, csv, pdf etc. It becomes difficult to parse this data into required format as it requires extra overhead.To send this data we are considering two options namely SOAP services or RESTful Services. RESTful services are replacing SOAP services because of their various advantages described further in this paper,So to provide the interoperability between devices we suggest to use RESTful webservices susing json as output. REST is an architectural style to develop webservices which represent each device with an unique URI.We will discuss as to how we can use OAuth as a security mechanism for these webservices and how effective it is to use with IoT devices.We provide general guidelines on how to design a REST API(Application programming interface) device description for machine-to-machine interactions. We will be discussing about the pull approach where only client can initiate communication and apush approach where a server can also start an interaction with client based on the triggers defined which in our case is a voice command and how sall these fit in to define a complete communication approach.

KEYWORDS: REST,SOAP,Android,WebServices,Google AIY kit, Intercommunication,Raspberry Pi 3 Model B,Voice Assistant,IoT

I. INTRODUCTION

During the past few years, the vision of future Internet has been evolving into Internet of Things (IoT). IoT refers to connecting the physical world things, like sensors and small devices, into the Internet. The Internet of Services is a vision of the future Internet where everything that is needed to use and develop software applications are available on the Internet. The developers must be able to independently develop and deploy services [1]. Thus, the context providers and the consumers should be loosely coupled, interacting with each other only through the middle-ware or APIs. The IoT environment is highly heterogeneous, which is one of the main challenges in IoT [2]. This includes different communicationtechnologies, capabilities, and types of data generated by the IoT devices. IoT paradigm should be able to handle such heterogeneous interoperability. Thus, the designed solution should be scalable and should be able to integrate with systems implemented in different programming languages and platforms. [4].Some APIs can be used with or without authentication,for those which require authentication we can use OAuth protocol.

II. COMMUNICATION APPROACHES

A. SOAP Web Service

SOAP stands for Simple Object Access Protocol. It defines a standard protocol used for exchanging XML-based messages. It is defined as a protocol specification for exchanging structured information in the implementation of Web Services in computer networks [1]. We provide specificationswhich defines an XML based envelope for exchanging messages and the protocol defines a set of rules for converting machine specific data types into XML representations.

B. RESTful Webservices

REST, short for REpresentational State Transfer, is attracting attentions of researchers all over the world. The concept was first proposed by Doctor Roy Thomas Fielding in his PhD thesis [4]. It’s thought that

RESTful Web Service improve visibility, reliability, and scalability, and is consistent with the initial design ideas of Internet.

Key REST principles:

- Each resource is given the unique resource identifier.
- The communication between clients and servers use standard HTTP methods.
- Resources are with multiple representations.
- The communications are stateless.

REST have four basic operations:-
GET: Methods for getting data from the server. These are read only operations.

POST: Methods for adding details to the server.These operations always make a change to server as these are non-idempotent operations.

PUT: Methods for updating the data at the server. These are idempotent operations.

DELETE: Methods for deleting the data at the server.

```
@Path("/usict")
public class RestServiceClass {

@GET
@Produces("application/json")
@Path("/AllStudents")
public String getAllStudents() {
Connection conn = null;
String jsonResponse="";
try {
        Class.forName("com.mysql.jdbc.Driver");
        conn = DriverManager
        .getConnection("jdbc:mysql://localhost:3306/jaydb","root","root");
        String sql = "SELECT * FROM usict";
        jsonResponse= resultSetToJson(conn, sql);
        return jsonResponse;
}}}
```

Fig.1 . A basic RESTful Web Service using Jersey Framework

There is close internal relationship between REST and IoT. In RESTful Web Service, each resource has a unique identifier[3], URI. As a result, all the objects can be traced accessed and operated through unique URI. The visibility of objects is greatly improved. RESTful Web Service can work as a middleware between the physical world and the virtual digital world, which provides standard HTTP methods to users or third-party developers. The users can access or operate the resources without any care about the underlying interfaces.
Multiple representations of same resources provide a solution to integrating various kinds of real-world devices into the platform. The platform can provide advanced representations to smarter devices such as smart phones, while simple representations to just ordinary devices with limited functionality.

CRUD Operations	**REST Keywords**	**Database Operations**
CREATE- Create or Add new entries	POST	INSERT
READ – Read, Retrieve	GET	SELECT
UPDATE – Update or Edit Existing Data	PUT	UPDATE
DELETE – Delete Existing Data	DELETE	DELETE

Fig. 2. Types Of REST Operations

III. DESIGN AND IMPLEMENTATION

Author will be using Eclipse, Apache Tomcat with Jersey framework to develop these RESTful webservices as described in Fig. 1.

Resource archetypes[7] for device resources and rules relating to the URI design are given in the following:

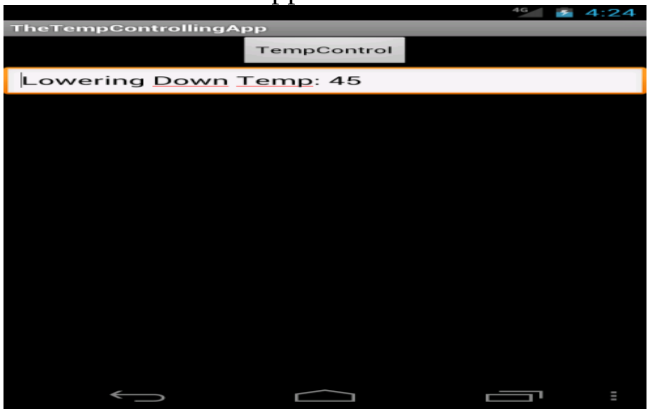
- Each API should contain one device resource (a mobile service or application e.g. usict/), which is a parent to its subresources.
- Collections are server-managed directories. A plural noun should be used for collection names (e.g. /students).
- Path segments may be substituted with identity-based values (/usict/studentId).
- URI can also contain optional query part, which contains a set of parameters
  e.g. URI=scheme://authority/path?query.

RESTful APIs provides the basic CRUD operations:

- GET /resources/ lists all resources
- POST /resources/ creates new resource
- GET /resources/:id read a single resource
- PUT /resources/:id update a resource
- DELETE /resources/:id deletes a resource

A. Android application with only pull approach

As author has already defined the approach to develop these RESTful webservices author will be using these web service to communicate between an android application and another machine on a local network.



Lowering Down Temp: 45

Fig. 3.Android application with only Pull Approach

For above application we have hosted a RESTful webservice on a laptop which acts a server or as a gateway which provides services to the applications on a local network which can be
used in home automation if we do not need anything from internet and we can fulfill our purpose by communicating among devices only. If we need services from internet we can expose our gateway to internet. In the same way, if needed we can host our web services on cloud instead of hosting it locally. In above application as soon as we hit TempControl button it will call /RESTful Term Paper/ rest/ usict/ room Temp which has its own server logic to handle and return an appropriate response.

Till now we have used only ‘pull’ implementation of REST as in this style each time we make a request from client to server which will provide us the results needed but what if we make a request and after that processing of that request takes couple of hours or we want to get notified from server whenever there is change on server side content such as a database change.

B. Two way communication

For this two way communication we have few methodologies such as:-

Short polling:

In this technique client constantly ask server for any change that happened at server side If yes then send me the response. Client will constantly ask server for any change with some regular interval of time between these requests.

Client - Is it done yet?
Server - No
Client – Is it done yet?
Server –No
Client – Is it done yet?
Server – Yes. Here is the response.

Long polling:
In this technique Client will make a request to server and this connection will remain open till the time server have any response to send back.If the server does not have any information available for the client, instead of sending an empty response, the server holds the request and waits for some information to be available.
Client – Is it done yet?
Server – waits till the time the request gets processed and then send response “Yes its done”

Websockets:
This is a full duplex communication mechanism in which there is a persistent connection between two ports with one on each machine. Both parties can use this connection for sending data at any time.The client establishes a WebSocket connection through a process known as the WebSocket handshake. This process starts with the client sending a regular HTTP request to the server.A header is included in this request that informs the server that the client wishes to establish a WebSocket connection.

Pub/Sub Technique:
In this technique[5] we have defined subscriber to a particular event or a topic.In case, we want to send notification to only one user we can use its unique key which we will get after providing correct credentials and after that we will be using this Unique token and Api key(OAUTH) for authentication and identification purpose.In case,when we want to send notification to multiple devices we have to subscribe to topics for example /topic.All the subscribers will be notified whenever there is a request made having ‘topic’ in it.
In firebase,incase where we will be sending notification to a particular user which maybe on android or ios.We will be passing body as defind below

```
{"to":"token",
"notification":
 {"body": “MyMessage”,
"title": "Message from Jay"}
}
```

And we have to pass this API key in Authentication header.

```
@GET
@Path("/push/{param}")
@Produces("application/json")
public String pushMessage(@PathParam("param") String myMessage) {
        String jsonResponse="test"+myMessage;
        String url="https://fcm.googleapis.com/fcm/send";
        String apiKey="AAAAe9MB6kI:APA91bFI–IOmwwRMklcTehB";
        String appKey="key="+apiKey;

        Client client = Client.create();

        WebResource webResource = client.resource(url);

        String body="{\"to\": \"fKNX4t7uZPY:APA91bGOL9Fjn4aM0Ine2pXicsq8\","
        +"\"notification\":"
        +" {\"body\": \""+myMessage+"\",\"title\": \"Message from Jay.\"}}";

        ClientResponse response = webResource
                      .header("Authorization", appKey)
                      .header("Content–Type", "application/json")
                      .post(ClientResponse.class, body);
                      if (response.getStatus() != 200) {
                  throw new RuntimeException("Failed : HTTP error code : "
                        + response.getStatus());
              }
                      String output = response.getEntity(String.class);
              System.out.println("Output from Server .... ");
              System.out.println(output + "\n");
              return jsonResponse;

        }
```

Fig.4 .A RESTful Webservice Definition triggered via voice Assistant

C. Proposed Communication Approach

To demonstrate this communication process let us consider an example where we have one device which wants to be notified anytime a request is made to a particular resource(a REST api).To achieve this we have summarized our setup as described below:-

- Setup a Raspberry pi by installing raspian operating system on it using NOOBS[6] installer.
- Assemble Google AIY voice Kit.
- Attach Raspberry pi to Google AIY voice kit which we will be using to pass a command(or voice trigger) which will call a REST service hosted on a Server(in our case it is hosted locally on a laptop) and a device which wanted to be notified when a request is made to this resource.
- Host a webservice on Service on a Server which will eventually communicate with Firebase cloud which will send notification to the device.

Setup client application on an android device which will receive the notification sent by Server.

1)Raspberry pi Setup

The Raspberry Pi[8] is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. We can also use by using secured shell(SSH) or using remotely accessing it using softwares like VNC viewer .It is a capable little device that enables people to learn how to program in languages like Scratch and Python. It’s capable of doing everything you’d expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

- Format SD card which we will be using to install raspian operating system on it by formatting it first.
- In SD Formatter, select the drive letter for your SD card and format it.
- Use a computer with an SD card reader to install Raspian OS.
- Visit Raspberry pi website download section[10] and Click on NOOBS, then click on the Download ZIP button under ‘NOOBS (offline and network install)’, and select a folder to save it to.
- Extract the files from the zip downloaded earlier.
- Copy downloaded files to SD card.
- Once files has been copied,insert SD card having raspian operating system into Raspberry Pi and power it up using usb cable plugged into wall power socket and select ‘raspian’ as the operating system you want to install.

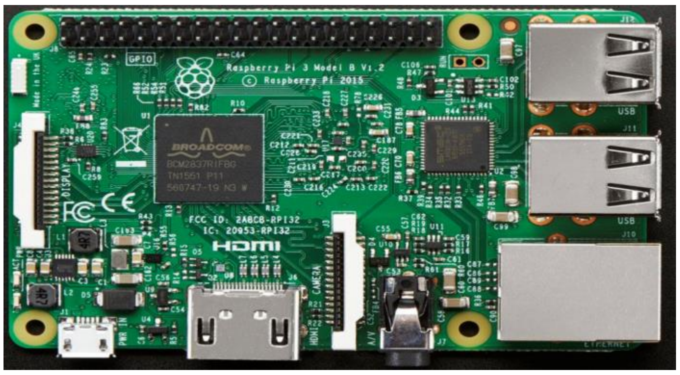


Fig. 5. Raspberry Pi 3 Model B[9]

Specifications[8]

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data

- Upgraded switched Micro USB power source up to 2.5A

Once you are done with initial setup you can access it from CommandLine using tools like putty(or VNC viewer if we need GUI access) for which you need to know the IP address of Raspberry pi, for this we can use tools like Advanced IP scanner. Benefit of accessing it remotely is that we do not need to attach mouse,keyboard,monitor each time.

2) Assemble and Configure GoogleAIY kit

Google AIY voice kit is natural language processer which we can manipulate and train according to our need and pass on the instructions to perform. We have python files where we can code our instructions and actions we want to perform corresponding to these instructions.

- Assemble the hardware following instructions from aiyprojects website[11].
- Use either the aiy project code available on github[12] or get latest voice AIY voice kit image from the link[13] given on aiyprojects website[11] and flash it on SD card using any image writing tool.

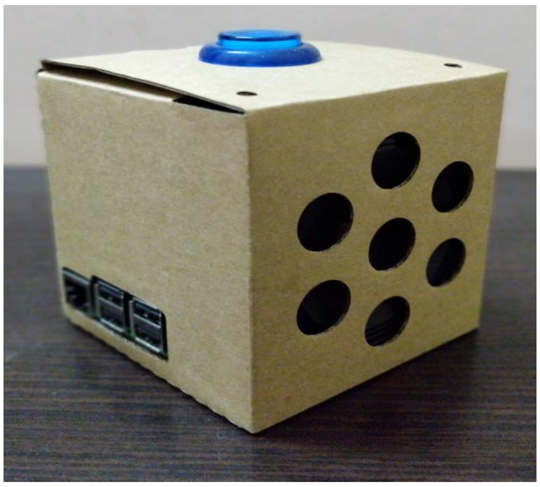
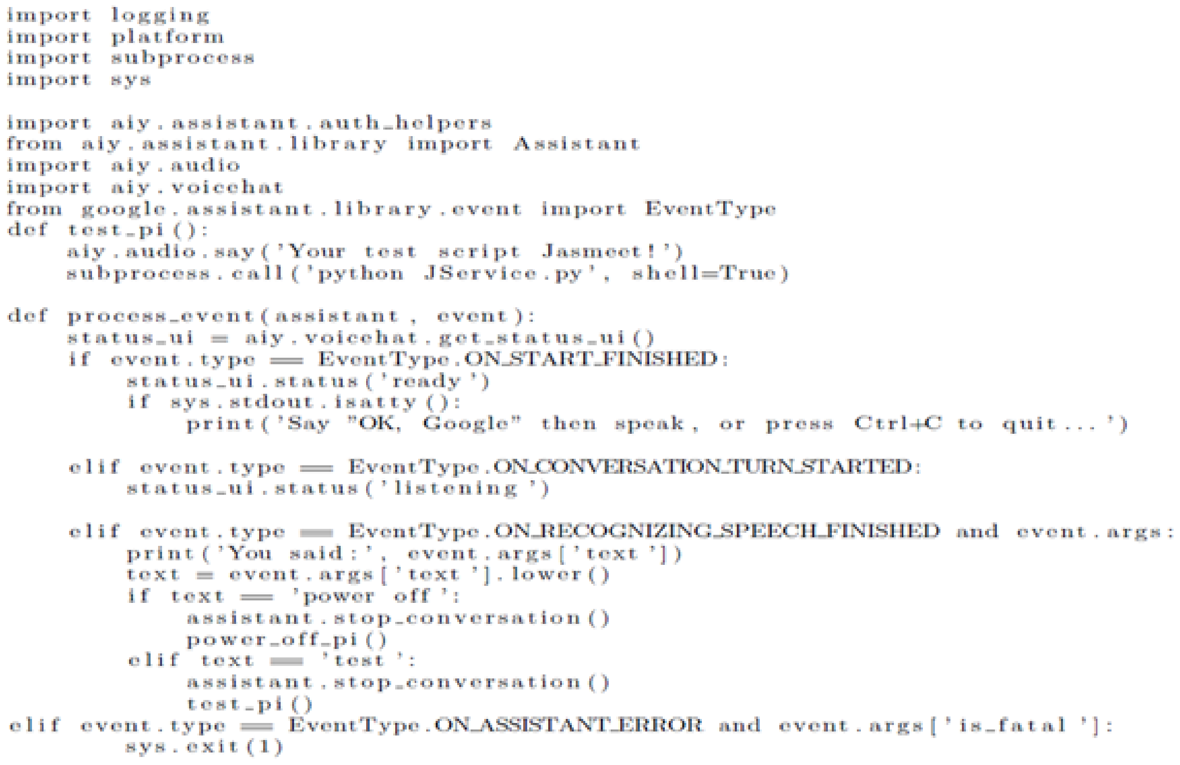


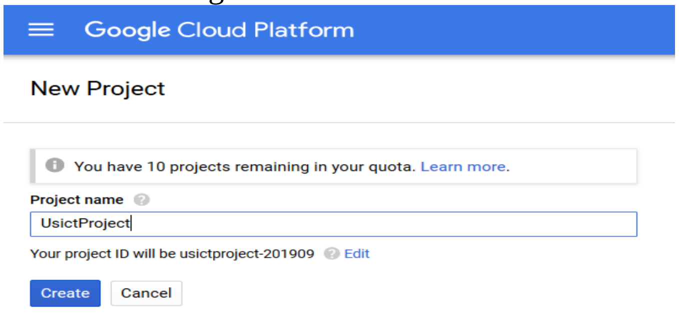
Fig.6 . Assembled Voice Assistant



```
import logging
import platform
import subprocess
import sys

import aiy.assistant.auth_helpers
from aiy.assistant.library import Assistant
import aiy.audio
import aiy.voicehat
from google.assistant.library.event import EventType
def test_pi():
    aiy.audio.say('Your test script Jasmeet!')
    subprocess.call('python JService.py', shell=True)

def process_event(assistant, event):
    status_ui = aiy.voicehat.get_status_ui()
    if event.type == EventType.ON_START_FINISHED:
        status_ui.status('ready')
        if sys.stdout.isatty():
            print('Say "OK, Google" then speak, or press Ctrl+C to quit...')

    elif event.type == EventType.ON_CONVERSATION_TURN_STARTED:
        status_ui.status('listening')

    elif event.type == EventType.ON_RECOGNIZING_SPEECH_FINISHED and event.args:
        print('You said:', event.args['text'])
        text = event.args['text'].lower()
        if text == 'power off':
            assistant.stop_conversation()
            power_off_pi()
        elif text == 'test':
            assistant.stop_conversation()
            test_pi()
elif event.type == EventType.ON_ASSISTANT_ERROR and event.args['is_fatal']:
        sys.exit(1)
```

Fig.7 .Python Script to Handle voice Command

- In order to use Google assistant API we need to sign in to Google Cloud Platform.
- Create a new project and enable Google Assistant API for it.


Google Cloud Platform

New Project

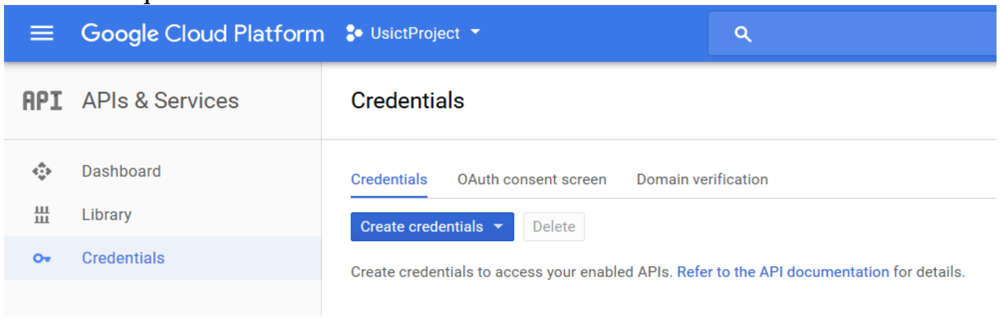
You have 10 projects remaining in your quota. Learn more.

Project name
UsictProject

Your project ID will be usictproject-201909 Edit

Fig. 8. Creating a new Project on GCP

- Create OAUTH credentials by clicking on ‘Create credentials’ which will be eventually used to authenticate our request.


Google Cloud Platform UsictProject

API APIs & Services

Credentials

Create credentials to access your enabled APIs. Refer to the API documentation for details.

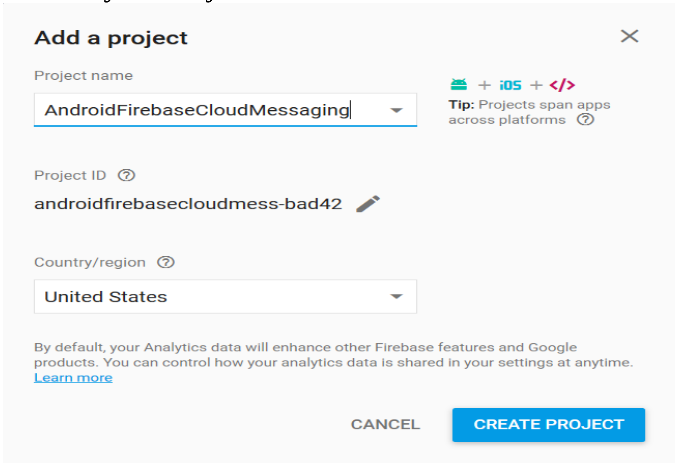
Fig. 9. Creating OAUTH Token

- A download option will be there to download assistant.json which we have to copy in /home/pi of our Raspberry pi.
- Now to define actions on our voice based commands we have to edit and add our custom command control.
- In Fig. 7 script when we say ‘test’ it will call our JService.py which will eventually make a server call to our webservice https://hostname:port/RESTfulTermPaper/rest/usict/push/{MyMessage} which is defined in Fig. 4.

In our Webservice we have specified the device which we want to send notification to.We have various ways to send notification or Server to client communication.Here we have choosen Firebase for that but for that we have to setup firebase first.

3)Firebase setup

- Go to Firebase cloud Website[14] and login.
- Create a new Project with any name you want.


Add a project

Project name
AndroidFirebaseCloudMessaging

Tip: Projects span apps across platforms

Project ID
androidfirebasecloudmess-bad42

Country/region
United States

By default, your Analytics data will enhance other Firebase features and Google products. You can control how your analytics data is shared in your settings at anytime. Learn more

Fig. 10. Creating New FireBase Project

- Click on Add Firebase to your Android app

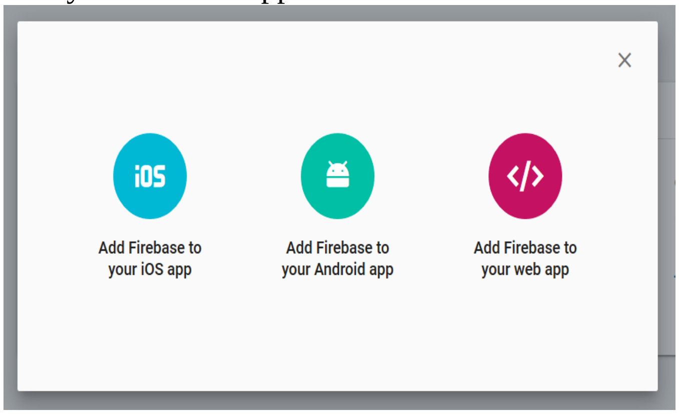
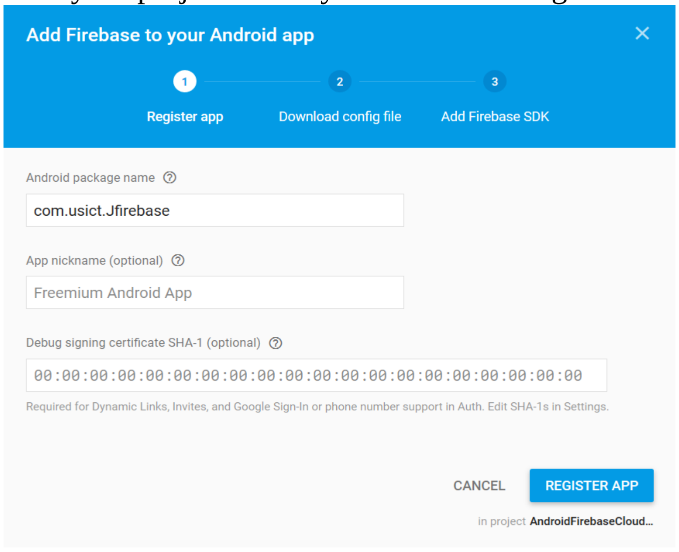

Add Firebase to your iOS app
Add Firebase to your Android app
Add Firebase to your web app

Fig. 11. Firebase Support on different Platforms

- Add the package name of your project which you will be making in Android studio.



Android package name
com.usict.Jfirebase

App nickname (optional)
Freemium Android App

Debug signing certificate SHA-1 (optional)
00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00

Required for Dynamic Links, Invites, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

in project AndroidFirebaseCloud...

Fig. 12. Adding Package name to Firebase Project

- Download the google-services.json file which we will be needing while making our Android App.

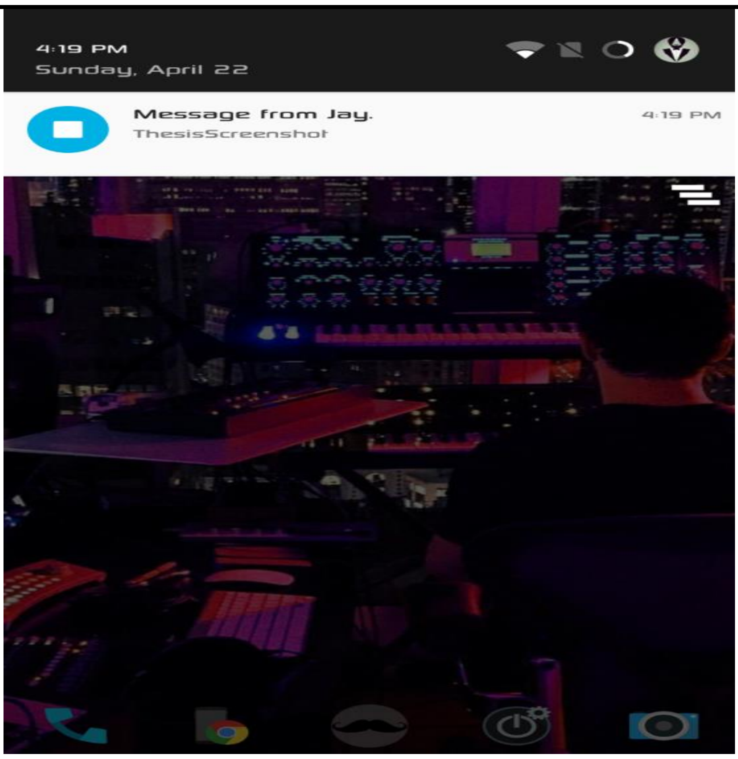
4) Creating Client Android Application
Add below dependencies for FirebaseCloudMessaging(FCM) to your project.
dependencies

```
{ compile 'com.google.firebase:firebase-messaging:15.0.0'
  compile 'com.google.firebase:firebase-core:15.0.0'}
```

Fig. 13.FCM dependencies

Add the google-services.json downloaded earlier to your project which have all the information regarding the client application having its package name as well as its API key.
Build the project and install its apk to a device which will eventually receive the notification.
Once we have each device configured we can proceed further by passing voice command to our voice assistant.
Firstly set voice assistant to listening mode by saying “Ok Google” followed by “Test” which is the keyword defined to execute a python script which will make a rest call to our REST service which will make a POST call to google cloud to notify all the registered devices by sending a sync notification.

NOVATEUR PUBLICATIONS
INTERNATIONAL JOURNAL OF INNOVATIONS IN ENGINEERING RESEARCH AND TECHNOLOGY [IJIERT]
ISSN: 2394-3696
VOLUME 5, ISSUE 6, June -2018



.

Fig. 14. Notification on Registered Android Device

IV. CONCLUSION

We have presented an approach for describing end devices as RESTful services.We have described a prototype implementation of an android application interacting with another machine.Here we are using only pull approach which is enough when there is no need for server to initiate an interaction process. JSON is used as a machine to machine device description format.

We have also proposed another approach where we have one voice assistant triggering an action to call a RESTful webservice which will be interacting with Firebase cloud which will be ultimately sending out the notification to the concerned devices.This approach also tackles the need where we need realtime communication and we need to push data from server to client i.e even server can initiate a communication process. Device descriptions can be used for end-user UI and HTTP request to interact with the devices via RESTful APIs.We are using OAUTH to authenticate a valid user when they are interacting with cloud services using OAUTH keys which we got from respective services host.

Heterogeneity is one of the main challenges in IoT. We have tackled heterogeneity by RESTful design, which means that heterogeneous resources are exposed as standard RESTful services. This simplifies service consumption and allows development of loose coupled services.These systems can be on same local network isolated to external internet or exposed to internet as required i.e if we only need to interact among devices on same network and do not have any need for services hosted outside the network. The other case where we not only need to interact among devices outside the network but there is also the need for real time communication alerts on your mobile devices.

REFERENCES

I. Lina Yao; Quan Z. Sheng; Schahram Dustdar,Web-based management of Internet of things, IEEE Internet Computing,Volume 19,2015,pp. 60-67.
II. Phillip J. Windley, API Access Control with OAuth: Coordinating interactions with the Internet of Things, IEEE Consumer Electronics Magazine,Volume 4,2015,pp. 50-58.
III. Sergio Trilles; Óscar Belmonte; Laura Díaz; Joaquín Huerta, Mobile Access to Sensor Networks by Using GIS Standards and RESTful Services,Volume 14,2014,pp. 4143-4153.
IV. Meng Wang; Chunxiao Fan; Zhigang Wen; Shan Li; Jie Liu, Implementation of Internet of Things Oriented Data Sharing Platform Based on RESTful Web Service, 2011 7th International Conference on Wireless Communications, Networking and Mobile Computing,2011,pp. 1-4.

V. Son N. Han, Soochang Park, Gyu Myoung Lee,Noel Crespi, Nguyen Van Luong, Kyoungwoo Heo, Mihaela Brut, and Patrick Gatellier,DPWSim: A device profile for Web servicces(DPWS) Simulator,IEEE Internet of Things Journal,Volume 2,2015,pp. 221-229.
VI. Raspberry Pi - Teach, Learn, and Make with Raspberry Pi, https://www.raspberrypi.org/help/noobs-setup/2/
VII. Leonard Richardson and Sam Ruby. "RESTful Web Services". Dongnan Industry Press, 2007, pp243–251
VIII. Raspberry Pi - Teach, Learn, and Make with Raspberry Pi,https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/
IX. Kiwi Electronics ,https://www.kiwi-electronics.nl/raspberry-pi-3-model-b/
X. Raspberry Pi - Teach, Learn, and Make with Raspberry Pi,https://www.raspberrypi.org/downloads/
XI. AiyProjects- Voice Kit(v1),https://aiyprojects.withgoogle.com/voice-v1/
XII. Github,https://github.com/google/aiyprojects-raspbian/
XIII. AiyProjects-Voice Kit(v1),https://dl.google.com/dl/aiyprojects/ aiyprojects-latest.img.xz
XIV. Firebase,https://console.firebase.google.com/