User Guide

April 2015

Version 1.0

Group 4

Table of Contents

1 Introduction

1.1 Scope and Purpose

It is a free platform for the Internet of Things. Users can deploy their Android applications which use sensory data on our platform. Our goal is to create a simple and efficient platform to deploy the applications for the IoT.

Key features and benefits of this platform are:

1. **Location aware sensing**:
   GPS location of each sensor is kept in the database. User applications can specify the location requirement such as 'data from x, y location within range z' and retrieve sensor data only from those sensors.
2. **Network failure tolerance:**
   Gateways are connected to each other in a mesh topology. This design provides significant level of network failure tolerance. If a connection from some gateway to filter server is broken, that gateway can reach out to its neighbor and forward data packet.
3. **Data filtering:**
   Not all kind of data from all the sensors is forwarded to the app engine. Filter server strips off un-necessary information, which no application is interested in, and forwards only required data to app engine on demand.
4. **Secure platform:**
   We ensure that no unauthorized device (sensor, gateway) can be added to our platform without administrative access. Each device has a unique identifier. Filter server checks that the data is received from a registered sensor and gateway.
5. **Easy to deploy platform:**
   A developer, who wants to build an application on top of our platform, need not worry about the intricacies of internal design. We have provided RESTful APIs for developers to perform set up tasks such as adding new device, registering new app, get list of sensors.
6. **Scalable:**
   It is highly scalable as any number of sensors and applications can be added or removed.
7. **Convenient and easy to use:**
   It is easy to configure. Also it comes with simple to use APIs and Json data format along with javascript.
8. **Faster communications :**
   We use websocket for communication. Websockets are amazingly fast compared any other alternative implementations, so it can be used for creating. Websockets are standardized being part of HTML5 and support across all major browsers.

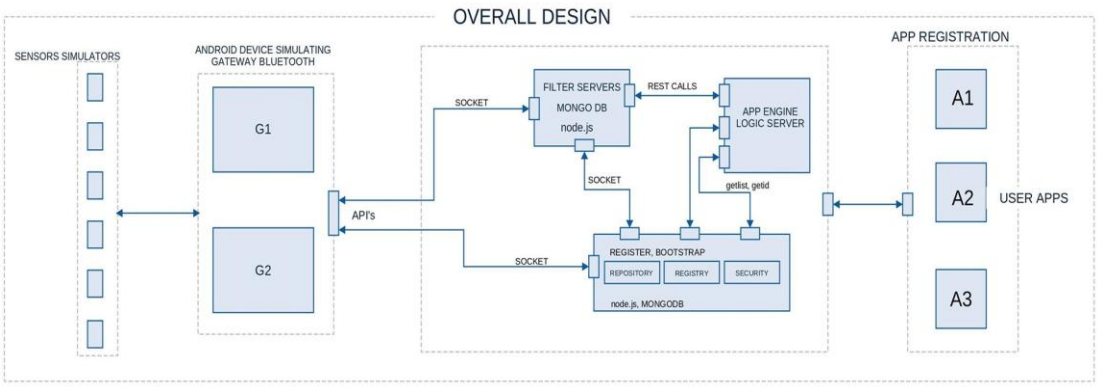
This user guide will take you through architecture of platform, overview of workflows along with installation and configuration details.

Users require having basic knowledge of JSON, Java and javascript.

1.2 System Overview

This section describes in short, the components that make up the platform and how they relate to each other.

Architecture Diagram:



As shown in the architecture diagram, this platform is composed of many different components that are wired together to provide the core platform. In the sections below, we will cover the individual components in more detail.

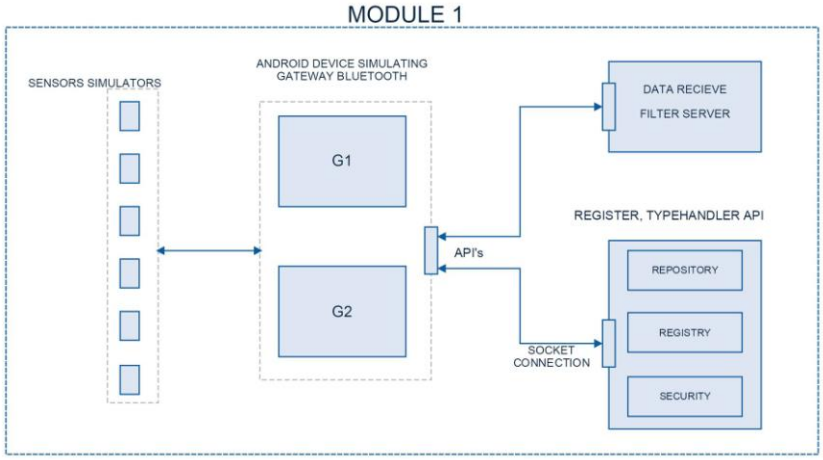
- **Sensors:**

  Sensors are gathering devices, which collects raw data from environment. They communicate to the gateway through Bluetooth.

- **Gateways:**

  Gateways are android devices which forwards the sensor data to filter server. Communication between gateway and server is socket communication.

  Following diagram explains relation between Gateways and servers, databases.



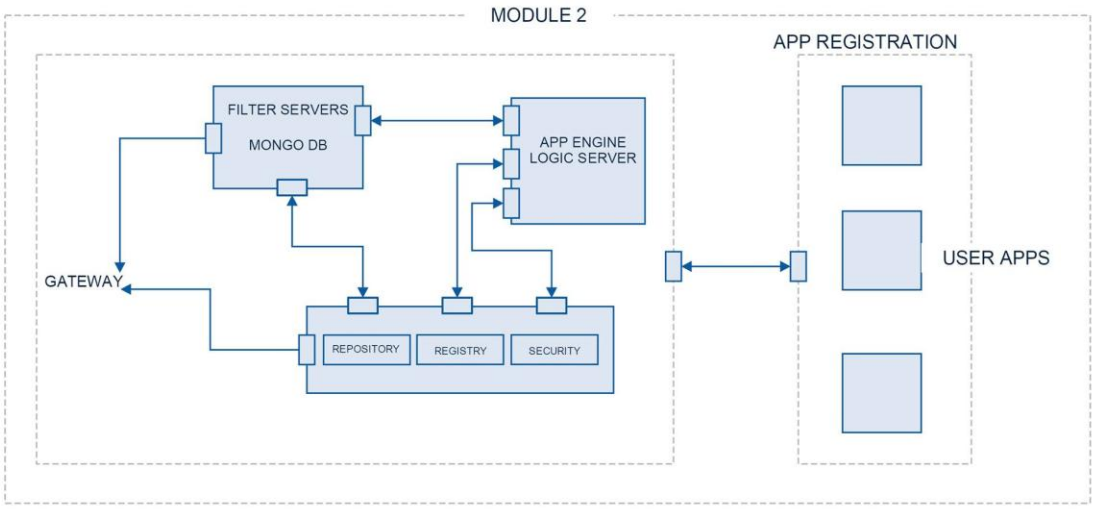
- **Filter Server:**

  Filter server filters the data according to the application need and stores in database/forward to application. It is written using Node.js. Communication is through the REST API with logic server.

- **App Engine/ Logic Server:**

  The application is deployed on logic server. It facilitates data transfer between filter server and application.

  Following diagram shows the communication between filter server and logic server.



- **Repository:**

  Repository is database, which contains entry for application id and its respective sensors. We use NoSQL database like MongoDB.

- **Registries:**

  Registry contains the sensor information like Sensor id, sensor type and installation timestamp. It also maintains the type handler for each sensor data.

2 Installation Details

Before installing the platform, system need to have below mentioned dependencies installed.

2.1 System requirements

- ***MongoDB***:
  MongoDB is a NoSQL database that can be used to store data. It is very quick and easy to install, has great performance, and does not require the computing resources of HBase. MongoDB is a great choice for smaller installations.
- ***Node.js****:*
  Node.js is an open source, cross-platform runtime environment for server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime.
- ***Express:***
  It is a Web Development Framework for Node.JS
- ***Socket.IO:***
  Socket.io is a node module available through the npm,It has two parts: a client-side library that runs in the browser, and a server-side library for Node.JS. Both components have a nearly identical API. Like node.js, it is event-driven.(socket.io).The Client side JavaScript to be included will be generated dynamically by this module.

2.2 Installing dependencies

- ***MongoDB*** :
  Usercan install the latest version from mongoDB and install the same.
- ***Node.js****:*
  Download the latest version from Node.js and install .
- ***Express:***
  To install express, enter following command.
  ```
  npm install express
  ```
- ***Socket.IO:***
  To install express, enter following command.
  ```
  npm install socket.io
  ```

2.3 Configuring the setup

Now that we have all the required softwares installed in our system, it's time to install the platform and start working on it.

- Run the startup.sh script with communication port as parameter to start the platform. The script will start the servers and setup gateways. Default port is 8080.

  ```
  bash startup.sh 8080
  ```

- User will be asked to enter following details in.
  - If user is running the platform for the first time, she needs to register as Admin and set password. Otherwise user can login using your user id and password.

Once user has setup the system correctly, she can now move on to deploy the application in next section.

3 Deploying Application

3.1 Registration and Authorization:

Login and Generating token:

User has to register and login to the platform via web UI. User has to register each app before deploying. The platform will generate the token to be used by the app. It will have the app id concatenated with a unique value.

Authorization API:

Application will be authorized by the token as follows.

Function : `authorize_app(oplat_token)`

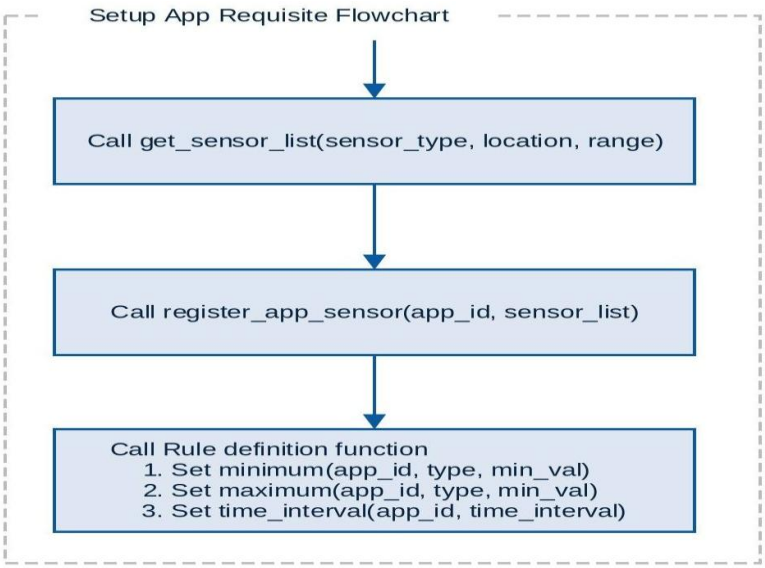
Status_code : `200 ok, 300 password failure, 400 app not registered`

Response:

```
{
      "status_code": {
      }
}
```

3.2 Adding/Selecting new devices:

For selecting or adding the sensors to platform the following API are exposed to the developer. Using these API, user can add sensors, select sensors from already available sensor list. Following diagram shows various functionalities provided to user through various APIs.



Sensor APIs:

1. *Function : get_sensor_list(type,lat,long,range)*
   Response:

```
{
      "Sensors":[
      { "sensor id":3423423,
      "type":sample_des
      "status":up/down
      "max_val": 100
      "min_val": 10
      },
      { "sensor id":9923423,
      "type":sample_des
      "status":up/down
      "max_val": 100
      "min_val": 10
      }]
}
```

2. *Function : select_sensor(sensor_ids_list)*
   Response:

```
{
      "sensors": [{
      "sensor id": 234112
      "status_code": 200
      },
      {
      "sensor id": 345211
      "status_code": 200
      }]}
```

3. *Function : add_sensor(type, latitude, longitude, type_handler)*

   Response:

```
{
      "sensor id": 345123
      "status_code": 200
}
```

Type Handler and JSON schema:

While adding any sensor, user also needs to give JSON schema and type handler for that data type supported by the sensor.

- **TypeHandler.jar:**
  Here, user has to implement getJSONObject method in TypeHandler.jar. The prototype of method is given as follows:

```
public getJSONObject(String fileName, String data)
```

Input:

fileName- It is the name of file which contains JSONSchema which was given while configuring the sensor.

Data - It is raw data sent by sensor.

Output:

This method will return sensor data in json format as specified in the JSONSchema.

- **Sample JSONSchema** for temp sensor is given as:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "TempSensor",
    "description": "Temparature sensor",
    "type": "Object",
    "properties": {
        "temp": {
            "description": "Temparature value sensed by sensor in K
format",
            "type": "integer"
        }
    },
    "required": ["temp"]
}
```

- **Sample JSON Object** corresponding to above JSONSchema

{

''temp'' : 322

}

3.3 Configure application requisites:

These API are exposed to set the filtering criteria for filter server like setting maximum threshold value, setting minimum threshold value and time interval. (Interval in which application wants the data)

Logic server API:

1. *Function:* `set_minimum(app_id, type, minimum_val)`

   This function will take a value and set it as lower threshold for sensor data has requested for.

   Response:

   ```
   {
       "status_code": 200
   }
   ```

2. *Function:* `set_maximum(app_id, type, maximum_val)`
   This function will take a value and set it as upper threshold for sensor data has requested for.

   Response:

   ```
   {
       "status_code": 200
   }
   ```

The two functions above set the filtering criteria for each app.

3. *Function:* `set_time_interval(app_id, time_interval)`

   It will take number of minutes, time_interval as input. All the data received in last time_interval minutes will only be considered valid.

   Response:

   ```
   {
       "status_code": 200
   }
   ```

3.4 Connecting application to the server:

Now user needs to add to code for connecting to the server. We use web-sockets as a mean by which the front end JavaScript can interact and send/collect data to the server.

- User first need to setup connection.
- Then she needs to send authorization message.
- After getting the authorization token, user needs to send a message to get the initial message by using APIs mentioned above.
- Following is sample example for client side javascript using websocket. It can be modified as per the users need.

Sample code for connecting to the filter server from application end:

```
var socket = io.connect("/");
/*Initializing the connection with the server via websockets */

socket.on("message",function(message){

      /*When server sends data to the client it will trigger
      "message" event on the client side , by using
      socket.on("message") , one can listen for the ,message
      event and associate a callback to be executed . The
      Callback function gets the dat sent from the server*/
      console.log("Message from the server arrived")
      message = JSON.parse(message);
      console.log(message); /*converting the data into JS object
*/
});

$(function(){
      $('#submit').click(function(){ /*listening to the button
      click using Jquery listener*/
            var data = { /*creating a Js ojbect to be sent to the
            server*/
            message:$('#message').val(), /*getting the text input
            data      */
            author:'karthic'
            }
       socket.send(JSON.stringify(data));

      $('#message').val('');
       //Emptying the text box value using jquery

      });
});
```

4 Maintenance

If user needs to make some changes after deployment, follow following procedure. The users can remove defective sensors or remove the applications which are not needed.

4.1 Removing dead sensors:

For removing dead sensors from platform this API is exposed to user so that unwanted sensors can be removed from database.

Function: `remove_sensor(sensor_id)`

This function takes sensor_id as parameter. Server removes the sensor from database and removes all the data collected from that sensor.

Status_code : `200 success, 300 failure`

Response:

```
{
      "status_code": 300
}
```

4.2 Removing application:

If developer wants to remove application from application engine then related data should be removed from the filter server. Henceforth filter server will not send unnecessary data to application engine, as now there is no application at the other end.

The following API is exposed to user to remove application.

Function: `remove_sensor_linking(app_id)`

This function call results in removal of all application data at filter server and sensors linkage to this application.

Status_code : `200 success, 300 failure`

Response:

```
{
      "status_code": 200
}
```

5 Credits

This platform was developed by following students under the guidance of Mr. Ramesh Loganathan as part of Internal of Application server course.

1. Gangasagar Patil - 201305549
2. Poorva Bhawsar - 201305555
3. Sushant Makode - 201305562
4. Kedar Biradar - 201305531
5. Omprakash Shewale - 201305567
6. Nikhil Barote - 201305560
7. Harshad Jalan - 201305536
8. Kapil Chhajer - 201305550
9. Diwas Joshi - 201305573

If there are any issues with the platform or queries, you can send an email to kapil.chhajer@students.iiit.ac.in or sushant.makode@students.iiit.ac.in .