Internal of Application Servers Application Platform

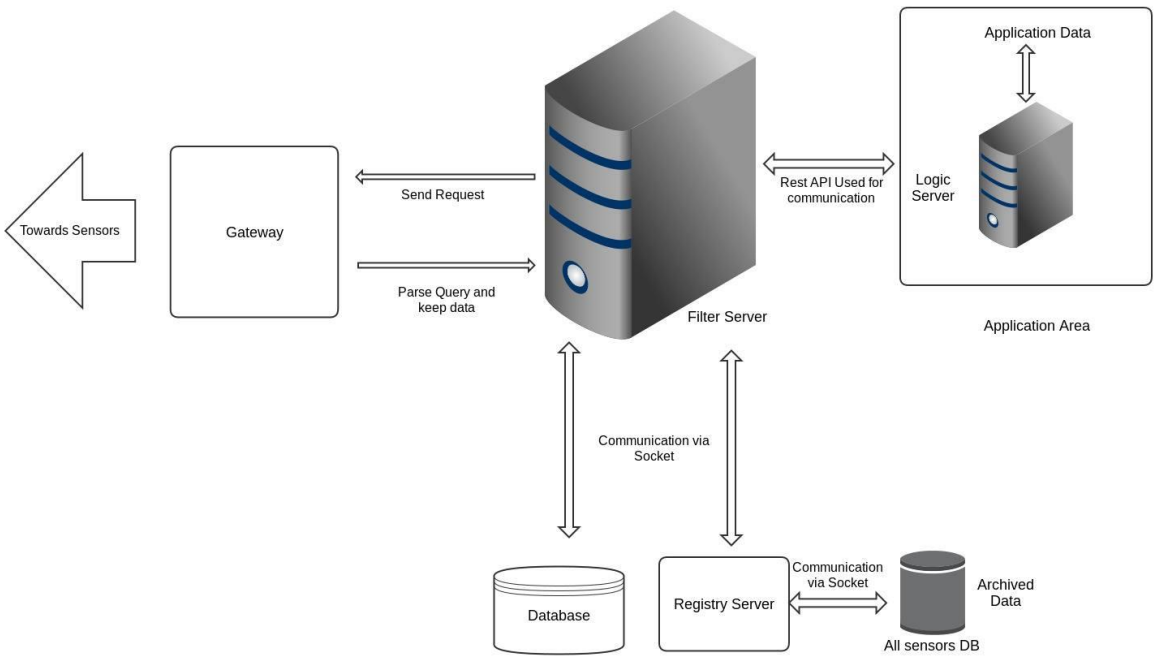
User Guide

Submitted By:

Group 2

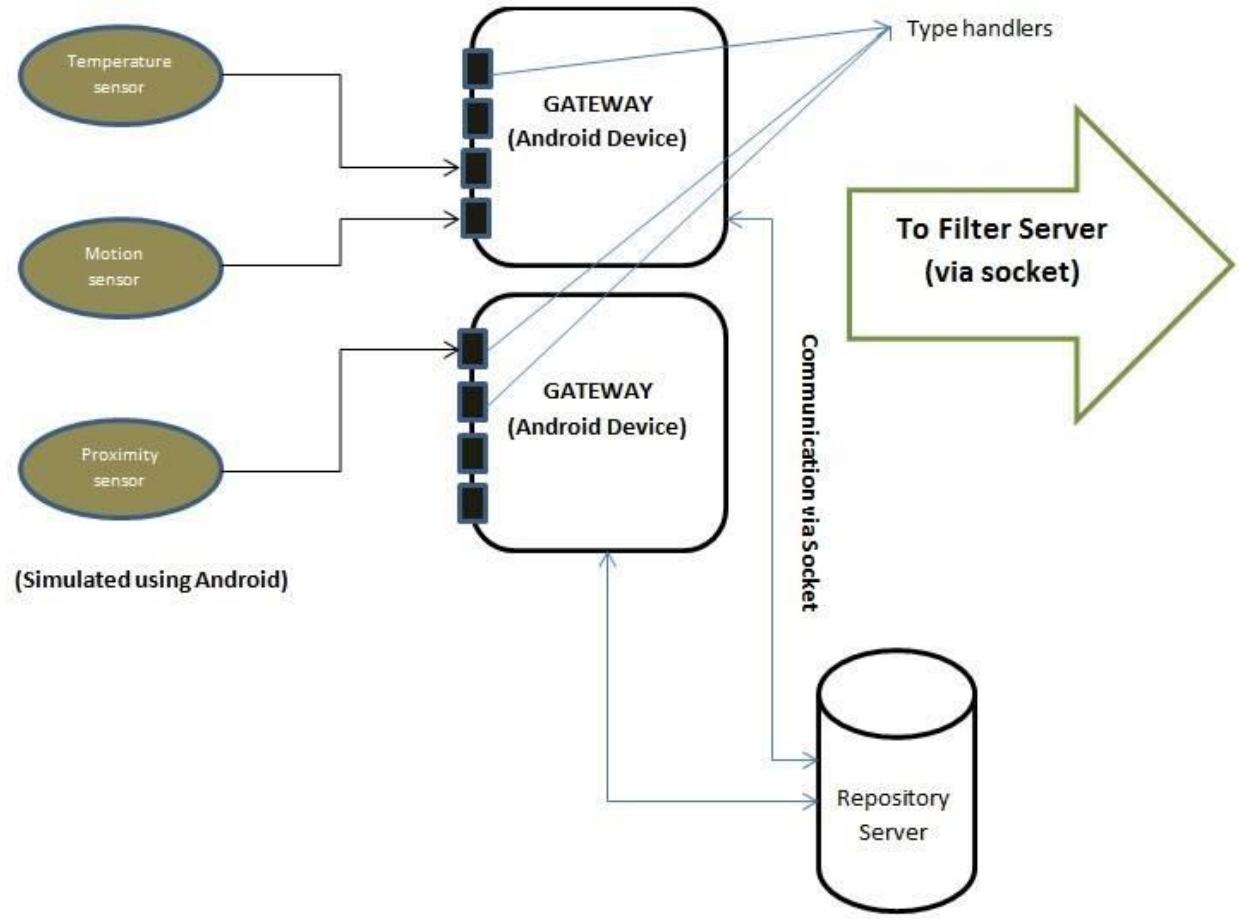
Contents

The Internet of Things is based around 3 subsystem, namely the Input Subsystem, Control Subsystem and the Output Subsystem. The App Developer needs to configure the first and third one to be up and running. From the top view, there are basically 3 things that the developer needs to configure. First, the sensors his app is going to use, the connection between sensors and the gateway and the communication between Filter server and the App/Logic Server.



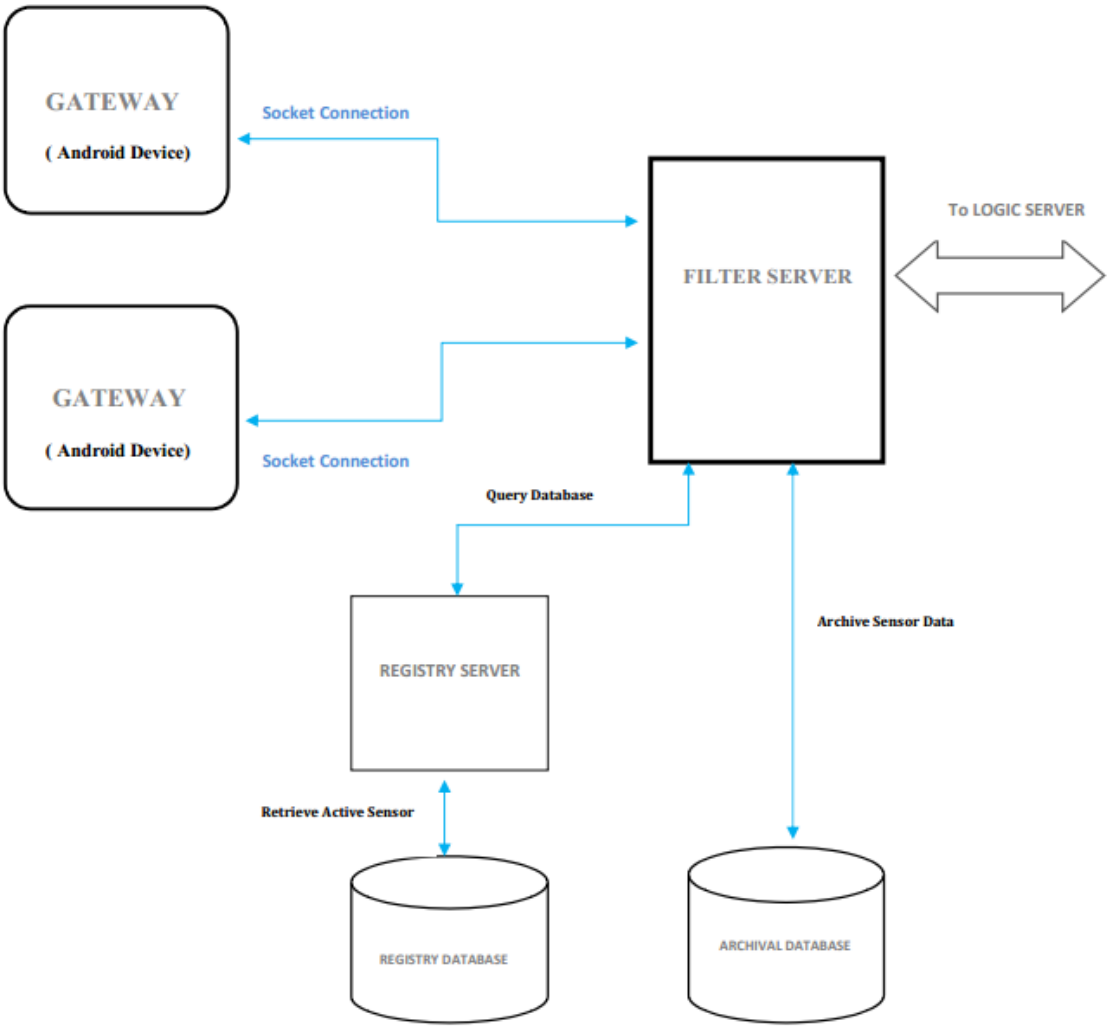
The above diagram enlists an overall design specification of how the platform will look like. First of all, the data sensed by sensors are sent to the respective gateways which collect the information and then cumulatively send all the data to the master gateway. The master gateway identifies the type of sensor and data by consulting the Sensor Type Repository. Since, raw data cannot be sent across the communication medium, master gateway encloses the data, type, location and other properties concerning the sensor into a protocol header which is then transferred across the communication medium using RESTful web services on TCP/IP protocol. The filter server then filters out only those sensor data that are being requested by the apps that are built using the APIs provided by the platform. The Sensor type repository uses MongoDB as the database and the Gateways are instances running on an Android system to simulate the functionality of an actual gateway.

Module 1



The above diagram illustrates the Sensor-Gateway Interaction. Socket programming is used for interaction between sensor and gateway. There is a collection of gateways which are simulated using an Android system that runs on Bluetooth. The sensors send their sensed data to the gateways they are connected and all these gateways then send the collected data to the Filter Server for further processing.

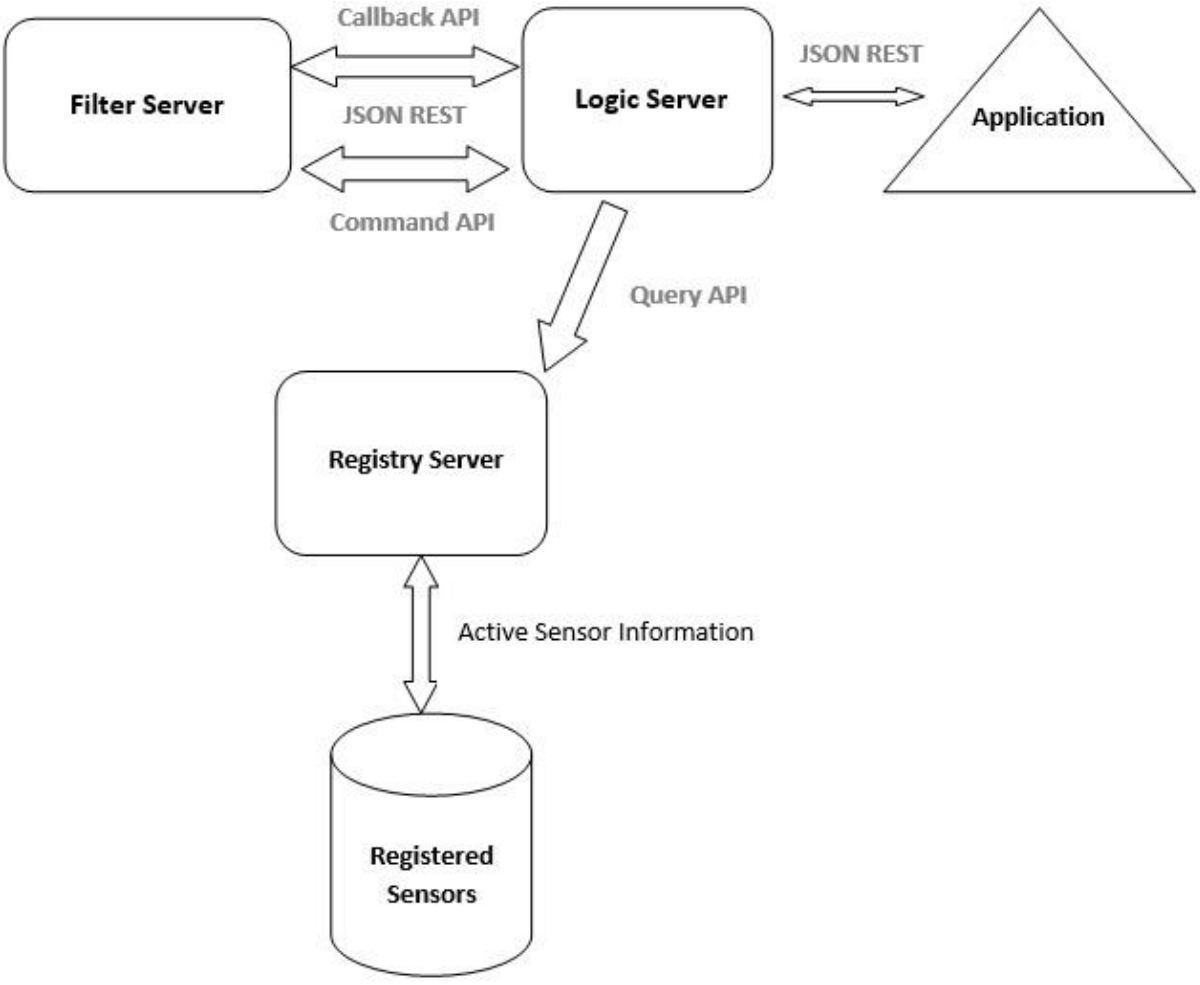
Module 2



The above diagram illustrates what goes on between the Gateway and the Filter Server. RESTful services are used for interaction between Gateways and filter server. The Gateway creates a protocol header containing the required information to transfer across the communication medium. The communication medium is the Internet

and the data is sent over the TCP/IP protocol. The message header containing information like id, type, location etc. is transferred to filter server for further processing. Type handlers are used at the Gateways to convert the received sensor data to a known format that the platform understands.

Module 3



RESTful services are used for interaction between Gateways and filter server. Since filter server receives data of all the types of sensors that are installed in the area, it must filter out only those that are requested by the application, i.e. an application requesting temperature information must not be shown humidity information and vice versa. For this, the filter server consults the sensor type repository and then sends only valid data to the application. Also, the filter server validates the source of generation of data. This is primarily required from the security standpoint otherwise malicious data might be shown to the user.

CONFIGURING THE SENSORS

✓ The ***Application Developer (user)*** needs to first register the sensor to the system. He/She has to assign a gateway to each sensor to communicate data. Since the sensors work on Bluetooth, it can send data to only one gateway as in Bluetooth, device pairing is initially required.

✓ Once the sensor is registered all its information like below are stored in the repository database.

- **ID** – Each sensor will have a unique ID associated with it.
- **Type** – This will indicate the type of sensor like temperature, motion etc.

- **Location** - This will indicate the position of the sensor.
- **Gateway Assigned** - Each sensor will have a gateway assigned to itself.

✓ Necessary JAR files need to be included for each of the type handler. Since the sensors can be heterogeneous, the developer needs to supply jar files for any new sensor type he wishes to use with the platform. Since, the data sent by different sensors is of different type, the jar file will convert the corresponding data into something that the platform understands. The developer can check whether a type handler already exists in the database. If not, the developer needs to provide the type handler to be used at the gateway.

✓ Above information about the sensor will be stored in the repository documents. All the information about the sensor will be stored in JSON format in the database. MongoDB is used as a database of choice.

Example JSON format

```
{
        “ID” : “Alphanumeric”,
        “Type” : “String”,
        “Location” {
                “Longitude” : “In Radians”
                “Latitude” : “In Radians”
        },
        … … … …
       “Gateway Assigned” : “String”
}
```

CONFIGURING THE GATEWAY

The next step is to configure the gateway so that proper communication occurs between the sensors and the gateway.

- ✓ When the ***Gateway*** boots up for the first time it obtains information about the sensors which are registered with the system from the repository documents. The gateway then checks whether the sensors are online or not. Since, each sensor communicates with a single gateway as they both work on Bluetooth, if the gateway fails or is down, the sensor cannot be reached. The Gateway also loads up the Type Handlers for different sensors that are associated to it.

Sensor ID	Sensor Type	Type Handler	Gateway

- ✓ Once the gateway boots up successfully it is now ready to receive data from the sensors. It caches the list of sensors that send data to it. The sensors continuously sense data at an interval of every 5 seconds and send the reading to the corresponding gateway.
- ✓ Each type of sensor will have its associated ***Type Handler*** with the gateway. The job of Type Handler is to convert the heterogeneous data of sensors of different types to a single format which the platform understands.

- ✓ If the developer intends to register a new type of sensor with the system then appropriate ***JAR files*** need to be included for the type handler to recognize the newly registered sensor. This would be required to convert the data to the required format.
- ✓ ***Health Ping*** messages are sent by sensors to the gateway so that it knows the corresponding sensor is online.

APIs to Fetch Data

The Application developer needs to develop a logic server that handles the logic of the query given to the platform. The logic server will process the query sent to the platform requesting the desired data. For e.g. receiving an alert when the temperature of sensor 1 rises above 30 degree Celsius. These kind of conditions are handled by the logic server. The logic server has at its disposal three APIs available:

1. **Query API**: The Query API has parameters such as type, location (GPS coordinates) and range. This Query API return a list of all the sensors that match the criteria of the condition. This is required for instance when Application demands a temperature of a specific area only. Then the logic server needs to check whether there is a sensor available in that are or in a location nearby. The response is in JSON format containing the list of sensors.

URL:

/query/?type=”type”&x=”x-coord”&y=”y-coord”&range=”in range”

The above URL is the format on which the query is sent and the response is received after getting data from MongoDB. The request parameters are used to refer to the type of sensor, the x coordinate, y coordinate and the range for which the sensors are to be identified.


```
{"sensors":[
    {"id":"1", "type":"temperature",
"location":"103", "alive":"yes"},
    {"id":"2", "type":"humidity",
"location":"101", "alive":"yes"},
    {"id":"3", "type":"smoke", "location":"105",
"alive":"no"},
 ]}
```

2. **Command API**: The Logic Server can register commands onto a Filter server to fulfil the user applications demands. The command can be of any form. Eg. Fetch the data of sensor with id = 2 or Fetch the data of sensor with type = “temperature” and location = “103”. This call is directly registered with the Filter server and it starts sending the required data to logic server as and when it receives it.

   URL: /command/

3. **Callback API**: The CallBack API is the one in which data is sent by filter server based on some condition. For e.g. send me the humidity data when the humidity crosses 25 unit mark. In this case, the data is only sent when the value crosses this limit. All the other times when the value is lower than 25, the data is discarded by Filter server and not forwarded to logic server. The response received is in JSON format which needs to be parsed accordingly by the developer to display on the application.

   URL:

   /callback/?cd1=”$gt|20”&cd2=”$lt|40”&type=”temperature”

   The callback API registers a callback event for the condition given in the URL. The condition in the URL specifies that an event should be fired when the temperature is between 20 and 40.

```
 {"sensors":[{
   "id":"1",
   "type":"humidity",
   "location":"103",
   "data":"26.7",
   "timestamp":"reading_time"},
   {
   "id":"2",
   "type":"humidity",
   "location":"101",
   "data":"30.7",
   "timestamp":"reading_time"},
   {"id":"3",
   "type":"humidity",
   "location":"105",
   "data":"27.2",
   "timestamp":"reading_time"},
]}
```

Defining Rules

Rules are conditions defined for some specific action to be taken. The rules can vary from simple conditions to complex conjunctions of different conditions. The actions can be specified with the rules that are to be taken once the specified conditions are met. For e.g. when there is smoke, blow alarm and call this number for help.

```
{"rules":[{
    "condition":"smoke",
    "action":"alarm&call",
    "location":"home",
    },
    {
    "condition":"temperature",
    "action":"increase",
    "location":"dining",
    "threshold":"30",
    "time":{
        "start": "startTime",
        "end": "endTime"}
    },
 ]}
```