TEAM - 4 MODULE Requirement

PREPARED FOR

IAS PROJECT - GROUP 4 - Team 4

IIIT-Hyderabad

PREPARED BY

TEAM 4

Akshay Choudhary- 2020202013

Shweta Arya -2020201047

Module Covered-

- Sensor Registration
- Sensor Management

1. Overview

1.1 Introduction

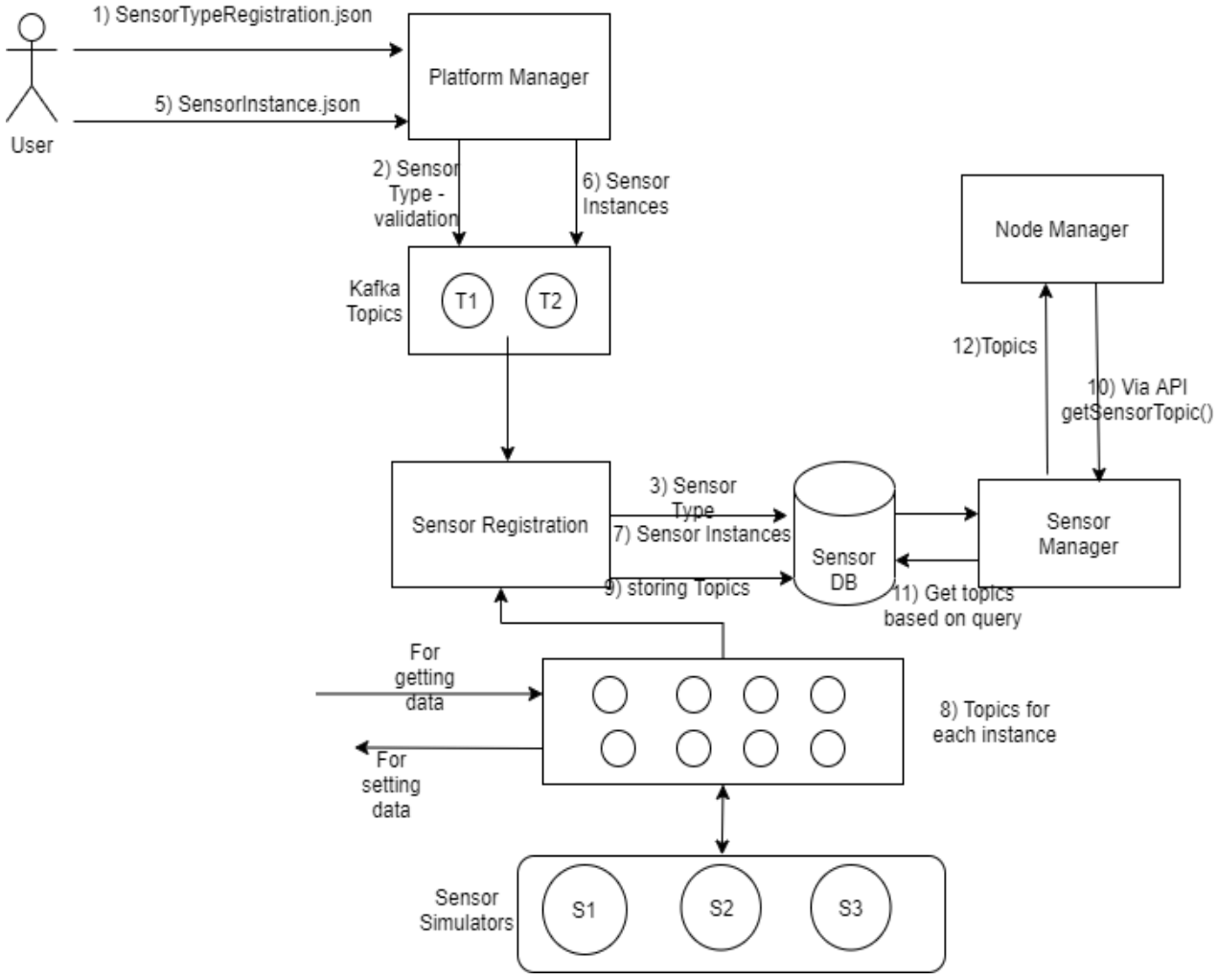
It is a distributed platform that provides build, development and deployment functionalities of applications. Most data becomes useless just seconds after it is generated, so having the lowest latency possible between the data and the decision is critical. With this platform, we bring IOT capabilities to the field. This IoT Application platform directly integrates with the IoT devices(sensors) and is capable of performing remote actions on these devices. We need to communicate and get data from sensors to be used by algorithms deployed on the platform and also perform respective actions on the sensors. Our module focuses on this area of the platform.

1.2 Scope

Our module basically handles Sensor Manager and Sensor Registration . Sensor manager module is responsible for dynamically binding sensor data with running algorithms/instances, and sensor registration for adding a new sensor at platform level, via minimal effort of application developer.

2) Brief overview of the Module

a) Block diagram



b) Functional Overview

Our team basically deals with the sensor registration and management.

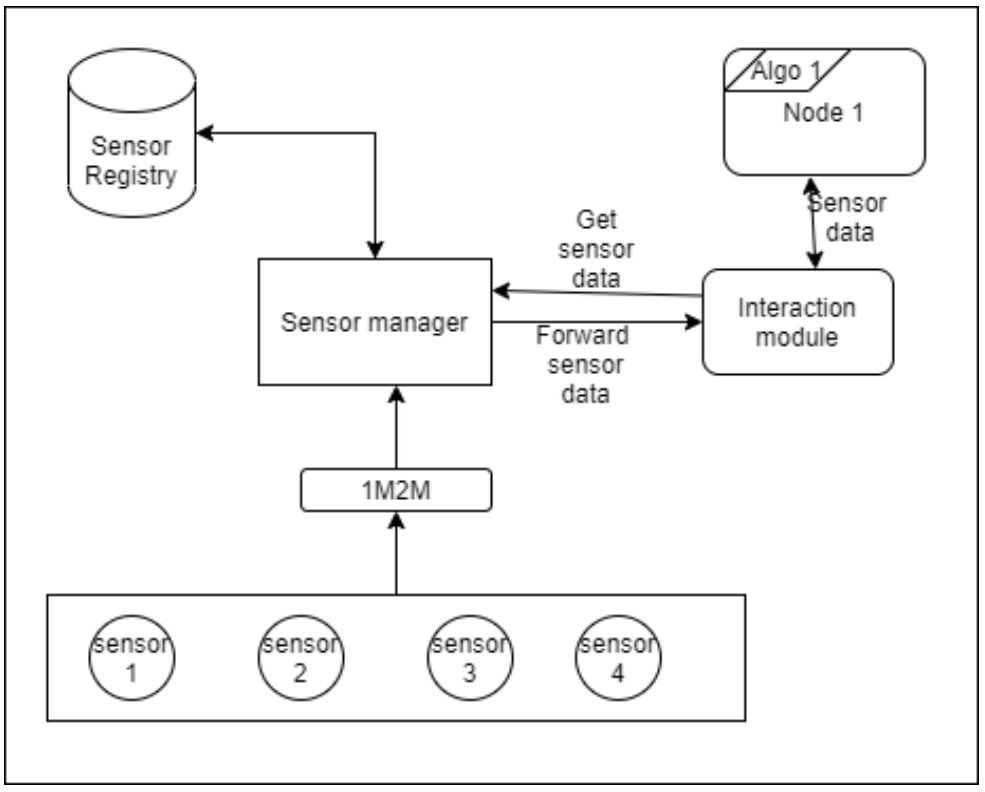
1. **Sensor Manager-**
   This sub-system connects with sensors and gets the raw data of streams. It processes the raw data into proper model input form and starts streaming it via the Message Queue which is consumed by the application on the server. This is the module to manage the sensor data and provide it to the algorithm via the API provided by the platform.

2. **Sensor Registration**- Whenever a new application is deployed in the platform, it also gives in the details of the sensors it needs. The detail about the sensor is provided in the form of a configuration file. Sensor Registration service parses this config file and stores the details about the sensors in the Sensor database. It further sets up the sensors and their gateways.

c) List of sub-systems in this part

a) Brief description of each component

Sensor Manager-

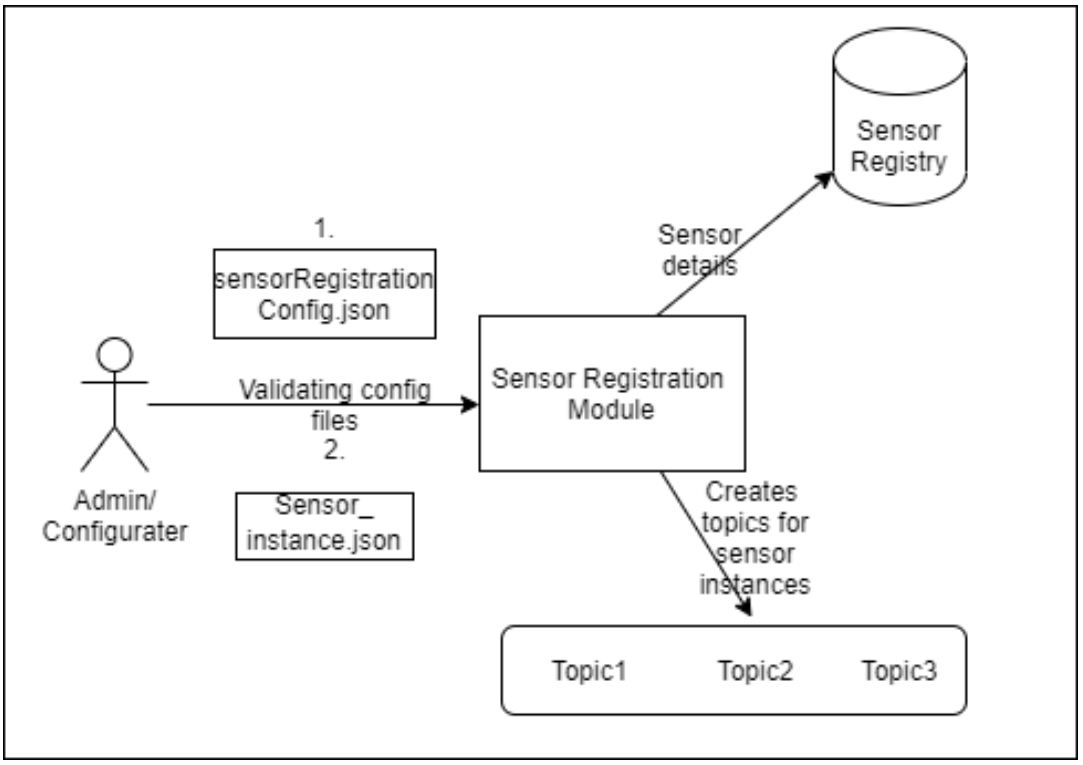


a) **Sensor Data Processing**: It processes the raw data into proper input form (as required by the application) and starts streaming to the Message Queue at a particular data rate which is consumed by the application deployed on the server.

b) Data Binding: The first part is to validate the sensor against the sensor details available. Then identifying the algorithm which requires the sensor data. Then a temporary topic is created for the communication between the application and sensor.

c) Sending Sensor Data: Sensor manager upon getting a sensor data pushes it to the respected topics on the topics. From where the application can pull the sensor data.

Sensor Registration-



a) Parsing and verifying the Config file obtained from the application developer to get the details of the sensor and its geolocation.

b) Registering the details of the sensor in the platform repository.

c) Registration of type and instance of sensors.

d) Setting up the newly registered sensors with their gateways.

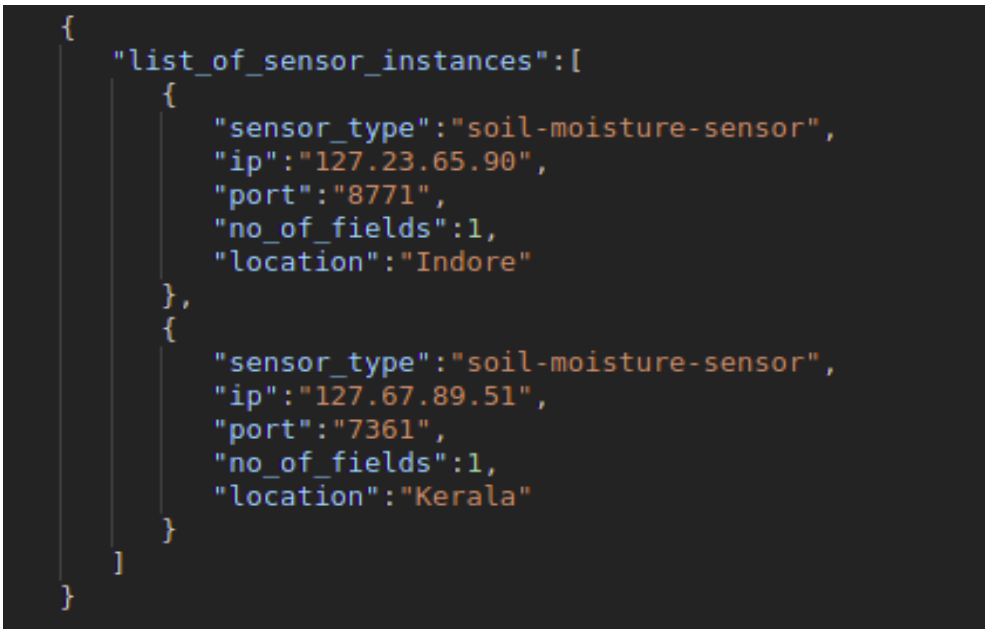
Different Config files Used-

1) **SensorTypeRegistration**

```
{
   "sensor_type_list":[
      {
         "sensor_type_name":"soil-moisture-sensor",
         "company":"samsung",
         "sensor_data_structure":{
            "moisture":"float"
         },
         "control_functions":{
            "number_of_functions":1,
            "function_details":[
               {
                  "name":"switchOnSprinkler",
                  "number_of_parameters":1,
                  "params":[
                     {
                        "time":"int"
                     }
                  ]
               }
            ]
         }
      },
      {
         "sensor_type_name":"air-condition-sensor",
         "company":"lg",
         "sensor_data_structure":{
            "temperature":"int"
         },
         "control_functions":{
            "number_of_functions":0,
            "function_details":[

            ]
         }
      }
   ]
```

2) **SensorInstanceRegistration-**



```
{
   "list_of_sensor_instances":[
      {
         "sensor_type":"soil-moisture-sensor",
         "ip":"127.23.65.90",
         "port":"8771",
         "no_of_fields":1,
         "location":"Indore"
      },
      {
         "sensor_type":"soil-moisture-sensor",
         "ip":"127.67.89.51",
         "port":"7361",
         "no_of_fields":1,
         "location":"Kerala"
      }
   ]
}
```

d) List of services/capabilities

Sensors

- Simulating the sensor types.
- Generating Topics where the sensors produce data and topics to consume data from so as to trigger actions on the sensors.

Sensor Registration

- Providing a sample config file for sensor type registration and registering the sensor types.
- Providing a sample config file for registering sensor instances and registering the sensor instances to be used by the sample application.
- Storing the sensor details of the sensors in the sensor db to be used by the sensor manager.

Sensor Manager

- Creating temporary topics to be used by the Applications for getting sensor data.
- The applications use API to get the topic names from the sensor manager and use it to get and set data from the sensors.
- Processing of the sensor data to be provided in the format required by the applications.

e) Interactions between this and other parts. Nature/purpose of interactions, likely interchange info/services

1. Platform manager and Sensor Registration- The config files received from from the admin/configurator is forwarded via api from platform manager to the sensor registration to validate and register the sensor types and instances.
2. Sensor Manager and Node(with an application instance running) , the application forwards a request via api of a sensor data it requires , so the sensor manager gets the sensor data , binds it and forwards the sensor topics to the interaction module and it sends the data to the application.
3. Control/Notification manager and Node(with an application instance running) , the output generated by the application is given to the notification manager to notify the end users or trigger actions on the sensors .

For interaction between the modules, we are using KAFKA producer-consumer architecture.
In our model we have used this communication at different levels regarding our module-

Between Platform manager and sensor registration-
Two topics are created , one for sending the sensor type registration config and another for instance config. Platform manager acts as a producer and sensor registration acts as consumer over the topic.

For the sending data from sensors to sensor manager and sending data from topics to the application instance. Here also the kafka producer and consumers are created.

Different topics created-

Kafka Topic Names:

Producer	Topic Name	Consumer	Purpose
platform_manager	pm_to_sensor_type_reg	sensor_registration	Send sensorTypeRegistration.json contents
platform_manager	pm_to_sensor_ins_reg	sensor_registration	Send sensorInstance.json contents
platform_manager	pm_to_sensor_binder	sensor_binder	Send deployConfig.json contents
sensor_binder	sensor_binder_to_scheduler	scheduler	Send deployConfig.json contents
scheduler	scheduler_to_deployer	deployer	Send deployConfig.json contents

3. Use Cases/Requirements

1. **End to End farm management System App(Primary).**
   To increase the productivity of agricultural and farming processes in order to improve yields and cost-effectiveness with sensor data collection by sensors like soil moisture, temperature sensor,pH sensor,water level sensor.We will use sensor data collection for measurements to make accurate decisions in future. It can be used as a reference for members of the agricultural industry to improve and develop the use of IoT to enhance agricultural production efficiencies.

   We have used two algorithms:
   - watercontent.py
   - airconditions.py

   The sensor types -
   - **air-condition-sensor** - For measuring the temperature of the soil
   - **soil-moisture-sensor** - For measuring the humidity of the sensor and turning off or on the sprinkler based on the threshold.

2. **Sample Application(Test use case).**
   A covid vaccination drive is going on at IIIT-H & to help people overcome the vaccine hesitancy IIIT-H is providing a transport facility using buses.College wants this transport experience to be comfortable, hassle free & want to span different areas of the city. Sensors used are GPS, biometric, temperature and light sensors.

Use cases:

1. Whenever someone boards a bus,he/she will do biometric check-in. Fare should be calculated based on distance multiplied by a fixed rate & should be displayed on the dashboard for that bus application instance (if your group does not have a dashboard, send a SMSnotification to guard) so that guard will collect that amount.
2. When the bus is not empty, if the temperature is more than a threshold switch on the air conditions or lighting is lower than a lux level switch on the lights.
3. Since college wants to span a maximum area, if more than two (>=3) buses come in a circle of given radius, except one send buzzer command to the rest. Run this service after every 2 minutes. (For the circle calculation part take suitable assumptions for connectivity).
4. Also,as such services require proper coordination from administration in covid times, whenever a bus comes closer to a barricade by a threshold distance start/trigger an algorithm which sends an email to administration mentioning unique identifier of the bus& an email body notifying them.

4. Test cases-

Sensor Module:

1. During sensor registration it will successfully be able to parse the sensorRegistrationConfig.json and sensor_instance.json file data from Platform manager and store the write information to sensor registry.
2. Topics for the sensor instance are created on registration or not.
3. For a given sensor type it will be able to filter out all valid sensor id based on registered sensors.
3. Will it be able to properly simulate the real time sensors .
4. Will the sensor module be able to provide the sensor data at the required rate and with the data type required by an algorithm.
5. Will the sensors be triggered successfully .