COSMOS

Ubiquitous Automation Solution

CSC 591 (059) INTERNET OF THINGS FALL 2018

CLASS PROJECT DESIGN DOCUMENT

DECEMBER 2ND, 2018

NC STATE UNIVERSITY

Jubeen Shah
Computer Science
North Carolina State University
jnshah2@ncsu.edu

Giang Nguyen
Computer Engineering
North Carolina State University
ghnguye2@ncsu.edu

Content

List of Figures

1. Introduction [0%]

Internet of Things (IoT) is a burgeoning field which is the next big thing in the world of automation. IoT devices usually have a universal application as they are not constrained to any one particular domain. With more advances in technology and the impact it can have on our daily lives, IoT can have various applications in our home that allow improving the way we live and carry out our daily chores. Smart homes are no longer an unexplored territory, and new devices are being invented to make our lives easy and make us feel secure. However, it has been known how expensive these individual devices tend to get. A single household has multiple lamps, and the cost tends to exponentially increase while the willingness to pay for each bulb by the consumers tends to decrease with increment in the number of devices being bought. [1]

Our project is a design and architectural extension of the project[1] undertaken by one of the authors (Jubeen Shah) in his undergraduate study. The idea of COSMOS, in brief, is to make affordable custom IoT end-points, that can be managed via a single device or custom designed interface. These devices could be any iOS, Android or any smart device which can access the internet in the desired automation environment. Few examples of the managed end-points that will be developed are intrusion sensors, fire sensors, connected RGB LED Lights, Connected Switches & Switchboards, integration of – temperature & humidity sensors, Air Quality Sensor, UV Sensor, Air Pressure Sensor and so on [2]. In a related trend, and keeping the user experience into consideration, other forms of digital interactions including personal digital assistants – like Siri, Alexa, and Google Assistant, are also considered for integration into the project.

This document outlines the business and technical requirement of COSMOS in Sections 2 and 3. The elements of the IoT system architecture including the things, people, sensors and actuators, involved are described in Section 4. Questions pertaining to the architecture of the project are addressed in Section 5, while alternate design methodologies considered with their limitations are discussed in Section 6. A detailed overview of the IoT-A architecture with the related diagrams and feedback model are discussed in Section 7 and 8. The type of analytics and computing models are discussed in Section 9. Details about the management tasks and virtualization are deliberated in Section 10 and 11. Finally, we wrap with some future work in Section 12.

[1] https://github.com/jubeenshah/COSMOS

2. Business Requirements [5%]

The *COSMOS* project should be designed to help the users/owner of the house in the following ways:

- **BR 1 –** Monitor various aspects of the environment of home to give the user a detailed overview of the conditions in the house.
- **BR 2 –** Record the values from sensors positioned in the house and store them in a persistent and reliable environment.
- **BR 3 –** Build a system that is scalable to allow for the addition of other devices, that were previously not considered for the implementation or add the same type of device.
- **BR 4 –** Visualize the data collected in the form of graphs to allow for easy interpretation by the user of the system.
- **BR 5 –** Develop a machine learning model from the collected data to allow for automated decision taking by the system.
- **BR 6 –** Manage the states of Switches, Irrigation system, and ambient lights in the environment.
- **BR 7 –** Manage the states of the devices in the environment automatically, manually and using a schedule from both, within the network and remotely.
- **BR 8 –** Manage the updates to the particular device automatically, manually, and using a schedule to ensure the security of the devices and vulnerabilities are sought after.
- **BR 9 –** Control and monitor the states of the things in the environment using a mobile application, and a web application and through a behavioral model.
- **BR 10 –** Integrate with voice recognition services such as Alexa, Siri, and Google Assistant to allow an additional layer of user experience and interaction with the system under consideration.
- **BR 11 –** Ensure durability of data in case of a disaster to allow for analysis of data
- **BR 12 –** Store the recorded data into a persistent data store.
- **BR 13 –** Make the model react to the predictions from the machine learning model.
- **BR 14 –** Monitor the state of doors and windows in the house which can be a point of entry into the house and allow for notifications to be sent to the user.
- **BR 15 –** Enable a way to automatically water the plants based on the type of plant the user has in his house in each room.

As a consequence of helping the user of the COSMOS in these ways, we would expect an increase in the productivity of the user for daily chores. We would also expect the users to save up on electricity costs. For this reason, we believe that this type of system could have significant commercial potential.

3. Technical Requirements [5%]

In order to meet the business requirements discussed in Section 2, the system must meet the technical requirements **TR 1-6** set out in this section. Even though we focus on these 6 technical requirements there are several more that the system would need as a part of a complete implementation.

- **TR 1 –** Custom designed hardware product. For example, sensors which compliment each other (Temperature, Humidity, Air Pressure, Light Intensity) can be put together on a single circuit board. This would enable the system to monitor the environment in a more complete fashion and thus satisfy **BR 1** and **BR 14**.
  - **TR 1.1 –** Obtain information about the environmental conditions.
  - **TR 1.2 –** Obtain the status of the devices in the environment.
  - **TR 1.3 –** Reacting to signals sent over from the cloud environment.
- **TR 2 –** A database cloud service (DynamoDB) would be needed to store the data collected into a NoSQL database. The high availability and reliability of the cloud service should satisfy the need for persistent storage for **BR 2** and **BR 3**.
  - **TR 2.1 –** Couple with AWS Machine Learning Service, **BR 5** can be satisfied.
  - **TR 2.2 –**The database solution with the machine learning service should help solve **BR 4** for visualization of data through IoT Analytics.
- **TR 3 –** With a streaming service already in place, a service such as AWS Simple Notification Service would enable to send notifications (email, SMS) to the registered users there yet satisfying **BR 6**.
  - **TR 3.1–** Fire off events such as lambda function in an AWS environment, **BR 7**, **BR 8**, and **BR 9** can be satisfied.
- **TR 4 –** A Mobile/Web application with API connection to the AWS services should allow for interaction between the devices in the house and the User. This would help satisfy **BR 10**.
  - **TR 4.1 –** For **BR 11** Depending on the operating system of the mobile being used; Siri or Google Assistant can be incorporated in the project.
  - **TR 4.2 –** Integrate with the local digital personal assistant available – Siri, or Google Assistant.
  - **TR 4.3 –** Provide Alexa integration if a digital personal assistant is not available for integration.
- **TR 5 –** Transfer of data from DynamoDB to an object-based storage (S3) with policies for lifecycle management and archival should help with **BR 12**.
  - **TR 5.1 –** Transfer the data from Standard S3 to Infrequently-Accessed (IA) S3 data store after 30 days.
  - **TR 5.2 –** Transfer the data from IA-S3 to Glacier after 60 days.
  - **TR 5.3 –** Transfer of data should not traverse the Internet, thus making use of NAT instances.
- **TR 6 –** Use of a pre-built machine learning model for classification of the plants and using inputs from the devices outlined in TR 1, should help with **BR 13** and **BR 15**.
  - **TR 6.1 –** Obtain information plant in the environment using the Amazon Recognition service.
  - **TR 6.2 –** Obtain the water requirement for the plant under consideration using a lambda function trigger from a DynamoDB.
  - **TR 6.3 –** Trigger AWS Lambda functions to take necessary actions to turn ON/OFF the irrigation system based on the soil moisture reading from the plant under consideration.

4. The elements of the IoT system [10%]

4.1.Things

This subsection describes the things in the project and their functionality in minor details.

- **Soil –** The soil moisture sensor would monitor the soil to record the moisture levels which satisfies BR 1.
- **Plants –** The water sprinklers would have a camera setup that would monitor the type of plant in the user's home which satisfies **BR 15**.
- **Rooms –** Environment station, with all the different sensors, would monitor the rooms for temperature, humidity, air quality, UV index, light intensity which satisfies **BR 1** There would be smart switches deployed in the room that would control the states of the switches and ambient lights in the room.
- **Door and Windows –** Intrusion sensor and motion sensor would be used to monitor the doors and windows of the house which satisfies **BR 14**.
- **Switch –** The switches would make use of the relays to switch on/off a device. The smart switches would simply make use of multiple relays to switch on/off a device like the irrigation system, the smart switches would be given three triggering mechanisms. The first would be an alert signal from any of the sensors in the environment which would trigger the switch being turned on/off, and the remaining two are discussed in the mobile app.
- **Mobile Device –** GPS would be used to monitor the location of the user and make predictions about the users' movement. The mobile device would be used to give an overview of all the connected things in the environment and also be used to manage and monitor other things in the environment. One of the ways it can manage other things in the environment would be through manual control of the devices by sending an HTTP signal to the server, which in turn sends an MQTT signal/command to the ESP8266 to turn on or off certain devices in the environment – Smart Switches or irrigation system. Another set of information that would be transferred between the mobile app and the server would be the Red, Blue, and Green values to be set in the Ambient Lights. Finally, there could also a schedule in terms of how frequently certain devices are to be turned on/off. For example, the irrigation system might be configured to turn on every day at 9 am, and turn off at 9:30 am. This can be done using the mobile application. Another, functionality that would be a part of the mobile app would be to allow integration with the in-house operating system's voice assistant – Siri in case of iOS and google assistant in case of Android. For any other operating system, Alexa can be used since, it is an open source assistant provided by Amazon, thus satisfying **BR 11**.

4.2.People

This subsection outlines the people directly involved in interacting with the project. It also outlines the roles of the people involved. This is not an exhaustive list of the people involved in the project, but the ones that would be directly interacting with the system either to use it on a daily basis (Users), or to solve any problem associated with the devices or services being used a part of the system as whole (AWS Administrator and Local Administrator).

- **Owner/User –** The first type of people would be the direct consumers of the information being generated in the house – the owners of the house and the system. The owner would monitor and manage the states of the things in the environment, and take any action in case of an anomaly.
- **Administrator –** The administrator would be responsible for maintenance of the devices installed in the environment and take actions if the devices are malfunctioning or not functioning. They would also be responsible for managing this information for a group of users across houses.
- **AWS administrator –** The AWS [3] administrator would be responsible for registration and de-registration of any services that the user subscribes through either manually, or automatically through the use of the mobile application or other devices.

4.3.Processes

This section describes in detail few processes that would be a part of the project. It describes the process for registration of a device with the environment under consideration for a particular home. It also provides the series of steps that would be performed whenever there is a need to transfer information between the things in the project and the AWS environment. Finally, we also discuss a few more processes that would be a part of the system but are not described in detail.

4.3.1. Registration of device

It would help with the **BR 1, 2, 7, 8, 9,** and **10** for each of the devices in the project as this would be the starting point for each of the aforementioned BRs. Also, it would make use of **TR 4**, since a mobile application or web application would be needed for interaction with the device.

I. Switch on the device
   A. Once on, the device would look for a wireless network to connect to.
   B. Once connected, the device would be available on the mobile device, the web application, or the command line interface for the people involved in the project to interact with, for the registration process.

II. Get device information
   A. Scan the barcode of the device using a smartphone application, to get the information such as the Unique identifier (UID) and the Name of the thing (device).
   B. In case of command line interface registration of the devices, the unique ID from the device itself can be copied into the CLI for registration.
   C. This information would be saved on the mobile application/ web application, to keep a track of the registered devices and then this information would be sent to AWS for registration[2] with the IoT core service using the AWS IoT management service.

III. Generate a unique certificate
   A. Generate a unique certificate for the thing on the AWS platform using the mobile application/ web application.

IV. Attach the policy for the devices to the generated certificate
   A. These policies would allow communicating with other AWS services in the environment.

V. Register thing with AWS
   A. Use the private key of the thing along with the device certificate to register the thing with AWS environment.

VI. Configure the device in the AWS environment
   A. Use the publish and subscribe parameters from the devices to generate as many numbers of events in the AWS environment.
   B. If the thing is configured to only publish values (for example the environment monitor), as many publishing topics would be generated as there are sensors publishing the values.
   C. If the thing is configured to be both publish and subscribe, respective topics should be created in the AWS environment.

[2] https://docs.aws.amazon.com/application-discovery/index.html#lang/en_us

4.3.2. Transfer of information between things and AWS

*I. The thing would collect data from the environment as required and described in **BR 1**, and **BR 2**.*
   A. This data could in digital or analog form. Such as temperature, humidity, soil moisture, etc.
   B. If the data is in analog format, it is converted into digital format.
   C. This process would be repeated for all the sensors mounted on the thing.
   D. If multiple sensors are connected to the same pins, a multiplexer or a demultiplexer would be used to switch between the sensors connected to avoid interference amongst sensors.

II. The thing would subscribe to events.
   A. These events can be manually triggered by the user, for example in the case of switches.
   B. These events can be automatically triggered by lambda function from other things in the environment.
   C. For example, the soil moisture sensor sending a lambda event to the irrigation system in case of reduced moisture levels in the soil.

III. Publish the data to the publishing topic.
   A. Publishing topic would be a unique address to which the things would be sending the MQTT messages too.
   B. Based on the publishing topic for the particular thing, the AWS message broker would route the formatted data from the publishing client to the subscribing client.

IV. AWS IoT Core responds to the event by creating a lambda function.
   A. Once the subscribing client has received the messages, it would trigger off a lambda function.
   B. Using the lambda function; the received data can be processed to carry out the following functionalities.
      1. Storing the received data in DynamoDB which satisfies **BR 12**.
      2. Based on the threshold values for the processed data, another lambda would be triggered to react with notifications (SMS, Emails, Push notifications) or triggering actuators in the environment.

4.3.3. Other processes

- Generating graphs from the data collected from the sensors as mentioned in **BR 4**.
- Classification of plants from the image capture as mentioned in **BR 15**.
- Generation of behavioral analytic model based on the use of devices like smart switches, smart irrigation system, and ambient lights as mentioned in **BR 9**.
- Creation of redundant copy of data as described in **BR 11**.

4.4. Data

This subsections outlines the type of data that would be collected in the automation environment.

- The data being generated by the sensors as required in **BR 1**, **BR 2**, and **BR 14**, are as follows :
  - Temperature
  - Humidity
  - Air Quality
    - CO2, CO, LPG, Propane, and hydrogen
  - Light intensity
  - Air pressure
  - Soil moisture
  - UV radiation
  - Motion
  - Magnetic sensor status

- The data from the actuators as required in BR 2 are as follows
  - Status of the switches
  - Status of the irrigation system
  - Status of fire alarm
  - Red, Blue, Green values form the Ambient lights

4.5. Sensors and actuators

This subsection outlines the sensors and actuators that would be a part of the project. The first subsection describes the list of actuators that would be needed, while the second subsection outlines the list of sensors that would be needed as a part of the Business and Technical requirements. This is not a comprehensive list and more sensors and actuators might be needed for implementation purposes.

4.5.1. Actuators

As required by **BR 7**, **BR 9**, and **BR 15** the following list of actuators are identified

- **Micro-controller (ESP8266) –** The ESP8266 would be the brains of the individual sensors that would have the compute capability to wirelessly transfer information to the cloud server. This would also have the power supply connected to it; either in the form of a battery or any other source of wall outlet (5V) [4].
- **Valves –** The valves would be part of the irrigation system which would be used to control the flow of water to the plants.
- **Relays –** The relays would be a part of the smart switches, and irrigation system which would be used to switch ON/OFF the devices in which it is embedded.
- **Temperature Controller –** The temperature controller would be a part of the smart thermostat system which would be responsible for setting the temperature of the environment.

4.5.2. Sensors

This subsection lists and describes in some detail the sensors that would be needed as a part of the project. This is not an exhaustive list and more/fewer sensors might be needed as a part of the implementation when considering each house.

- **Temperature and Humidity Sensor (DHT22) –** The temperature and humidity sensor would help record the temperature and humidity of the environment (room/house) with an accuracy unto *±2%* for relative humidity and unto *±0.5 °C (±0.9 °F)*. The primary purpose of the DHT22 [5] sensor is to send updates to the user and other things (actuators) in the environment to take appropriate action based on the readings from the sensor. For example, if the temperature shoots above a threshold value, an alert can be sent to the user that the air conditioner has been turned on, while concurrently sending a signal to the air conditioner to turn on. Similarly, in case of humidity readings, a dehumidifier can be turned on.
- **Air Pressure Sensor (BMP180) –** Air Pressure Sensor can give relative height from the sea level, and this information along with the temperature reading either from BMP180 [6] sensor itself, or the DHT22 sensor, can help control the temperature of the environment more accurately in hilly regions to provide more energy efficiency. According to the datasheet, the BMP 180 sensor has an accuracy of *±0.12 hPa*.
- **Light Intensity Sensor (BH1750) –** Light Intensity sensor [7] would help record the amount luminosity in the environment. This would then be translated into other signals for other actuators in the environment primarily the switches for lights and window blinds. It has a very high resolution that gives values ranging from *1 lux to 65535 lux*.
- **Air Quality Sensor (MQ2) –** The MQ2 [8] sensor would record the concentration of different gases in the environment. The concentration of the gases is calculated on the basis of electrical conductivity. MQ2 Gas sensor has high sensitivity to LPG, Propane, and hydrogen, and also other combustible steam. When different gasses are present in the higher concentration, the conductivity of the sensor increase.
- **Motion Sensor (PIR) –** The motion sensor is a Passive Infrared sensor [9] that makes use of anomalies in IR generated to detect motion in the environment. This sensor would be able to detect motion, and then based on a certain set of predefined rules and conditions, ESP8266 would be used to send a signal other devices in the environment to generate an alarm.
- **Soil Moisture Sensor, (FC-28) –** Soil Moisture sensor makes use of conductance between two electrodes, to determine the percentage amount of water in the soil. This sensor makes use of an Analog pinto detect the difference in voltage level, which can then be used to translate into digital data [10].
- **UV Sensor (ML8511) –** ML8511 [11] works on a similar principle such as FC-28. However, it is to be noted that both FC-28 and the ML8511 use the Analog pin to communicate with the micro-controller. What this means, is that only one of them would be able to communicate with the micro-controller at any one time since ESP8266 has only one Analog pin, a Demultiplexer such as the IC 4051 would be needed to switch between the Analog pins between two or more sensors using it.
- **Magnetic sensor –** A magnetic sensor [12] would be used to monitor and sense the state of the door and the windows on which it is installed. If the magnetic sensor is in contact with the reader, then a “closed” message would be sent to the AWS environment, whereas if the sensor is not in contact with the reader, then an “open” message would be sent to AWS environment.
- **Camera –** The camera [13] would be used to classify the type of plant as required in **BR 15.** It would make use of the AWS Rekognition service to classify the plant type.

4.6. Community of things

This subsection outlines an arbitrary collection of things, as seen in Section 4.1, which are useful from a management perspective. There are several communities of things that are possible with the project, however, we limit ourselves to four.

- **Community of rooms –** This would be a collection of the rooms in the house, with sub-communities defined for different types of rooms such as the living room, bathroom, and study room.
- **Community of plants –** This would be a collection of the different plants that would be identified in the house of the user.
- **Community of doors and windows –** This would be a collection of the doors and windows in the house that are required to be monitored by **BR 14**.
- **Community of mobile devices –** this would be a collection of the mobile devices being used by the user to manage and monitor the state of the devices in the smart house.

4.7. Federation of communities

This subsection outlines the communication. Between Heterogeneous sets of communities as discussed in Section 4.6. There are several federations of communities possible in the project, but we limit ourselves to the three that are discussed below.

- *Community of rooms communicating with community of mobile devices*
  - The different sensors deployed in the rooms would send values to the mobile devices via the AWS environment as mentioned in **BR 1**.
  - The mobile device would also send signals to the devices such as the smart switches, smart thermostat, irrigation system etch to control the states of the devices via the AWS environment as expected in **BR 9**.
- *Community of doors and windows communicating with the community of mobile devices*
  - The magnetic sensor fitted on the doors and windows would send its status when they are changed to the mobile devices which are described in **BR 14**.
  - These status messages would be sent to the mobile devices either whenever the state changes or as described by the user.
- *Community of plants communicating with community of rooms*
  - The plants would communicate with the room, which would help identify what plants are available in which room which is described in **BR 15**.
  - This information would be necessary to identify which rooms have which plant

4.8. Conversations

This subsection outlines the different types of conversations that are possible between the sets and subsets of things, people, and data as discussed in Section 4.1, 4.2, and 4.4 respectively. A few of the many conversations possible are listed below.

- **Things to Data –** Rooms would be monitored for different data points including temperature, humidity, air pressure, the light intensity which would be collected by different sensors in the room. For example,, when the temperature sensor records some value from the room it would generate data that would then be sent to the AWS environment for further processing or storage.
- **Thing to thing –** The mobile device is used to manage the states of the devices in the environment. For example, when the mobile device is used by the user of the system to manage the state of switches in the environment, the user would toggle the switch on the mobile device via the application which would then be sent to the AWS environment for processing. This would then trigger a Lambda function which would then cause the switch in the house of the user to change states.
- **Thing to People –** Notification being sent by the device to the user. For example, when the [18] alarm is triggered, it would send an MQTT message to the AWS environment which would then trigger a Lambda function to trigger the AWS SNS to send a notification to the user of the system on the mobile device associated with the automation environment.
- **Data to Things –** Trigger generated based on events from the database to control the states of things in the environment. For example, soil moisture reading, triggering the activation of the water sprinklers.
- **Data to People –** Visualization of the past information of the data being stored in the database. The data being collected by the sensors, when sent to the AWS environment would trigger several Lambda functions, one of which would be responsible for the visualization of the data being collected for the user of the system to understand.
- **Data to data –** When data is used to extract data from other databases. for example, when we use the flower name from the classification model to get the water requirements from another database.
- **People to Data –** People manually changing the status of a particular device in the environment. For example, when the user is using the mobile device to manage the state of a particular device in the environment, the user is also adding data points for the states of the devices that he is managing.

4.9.Enabling technologies

- **Visualization –** Visualization of data would be an enabling technology in the project, as it would give the user a glimpse of all the information that the system is trying to accumulate with all the sensors in the smart home environment. The visualization mainly can comprise a time series display of the readings from the sensors. Open source tools such as influxDB [14] (Time Series Database) and Grafana [15] (Open source Visualization tool) and Amazons QuickSight. For more details regarding virtualization used in the project please see Section 11.

- **Sensing Technologies –** The sensors as described in **Section 4.5.2** would be a part of the enabling technologies.

5. Architectural questions [5%]

5.1.How to identify things?

In this subsection we describe two ways that things can be identified – RFID and Barcode. The RFID method would be used by professionals setting up the environment for the user, whereas the method of barcode would be used by the user to set up the automation environment themselves as required by **BR 3**.

- *RFID*
  - Using tags for each of the devices we can uniquely identify a thing.
  - Program the tag to set a unique identifier, and attach the tag to the device under consideration.
  - When the need for identifying the thing arises, an RFID reader can be used to read the tags to uniquely identify a particular device, by reading the hexadecimal value from the tag and comparing the hex value, to the values stored in the database of the reader.
- *Barcode*
  - Barcodes are used to help the user identify the tags of the device faster in a more convenient manner
  - The user can scan the barcode using their smartphone or a barcode reader to identify the device and connect the device to AWS through the mobile/web application.

5.2.How to discover things?

Here, we briefly outline the AWS service that would be used for the discovery of things. This allows for setting up of a scalable environment as needed by **BR 3**.

AWS IoT Management Service[3] would allow bulk registration of things

- Whenever thing is to be discovered in the environment a simple API call can be made to the AWS servers with the following parameters – region name, zip code, street address, apartment number, floor, room, thing type, and ID.
- If any parameter(s) is/are not provided, then the rest of the parameters would be used to discover the thing.
  - For example, if the floor, room, thing type, and id are not provided, the API call would return the set of things in a particular apartment.
  - A JSON file for registration of several devices with attributes defined like Wattage, type, model, name, ID, etc. would be returned.
  - This JSON file would also help ensure the **BR 8** to control the deployment speed of the over-the-air updates to the consumer devices already in the field.

[3] https://aws.amazon.com/iot-device-management/

5.3.How to connect to things?

- Our project would be using Publish-Subscribe methodology for things to publish values to AWS message broker from the publishing clients, and then the subscribing clients at the AWS end would listen to the events from the things in the smart home environment.
- There would be certain things that would be subscribed to events from the AWS end, such as the Irrigation system or the smart switches that would be listening to the events from the AWS Lambda service (essentially the MQTT messages) that would be to switch them ON/OFF.
- So things in the environment would be connected through the AWS message broker, followed by a layer of lambda function which would be responsible for reacting to the messages sent from one thing in the environment to another thing in the environment.

5.4.How to connect things to people?

Here, we describe two ways of connecting things to people. One would be the way that the user would be interacting with the system. This would be to check the status of the devices, and/or monitor the states of the devices deployed in the house. The second would be for the administrators to interact with the environment for debugging problems encountered by the users, or serve important security patches, or any other managerial functions that the administrator would have.

- *Via the mobile Application/ Web Application*
  - User interaction with their home environment would happen through a mobile/web application. The mobile/web application would connect to the user's environment using a generated unique identifier to his/her environment. This would be set up at the time the devices are set up at the home environment as described in **BR 9**, and **BR 10**.
  - The user can access the readings of the devices such as temperature, humidity, air quality, moisture level etc through the mobile/web application as needed by **BR 4**.
  - The user can change the statuses of devices such as smart switches, the brightness of the light, the temperature of the room etc, through the mobile/web application at any place in time as called for in **BR 6**.
  - The user will receive notifications about the configured events that were detected from the environment and any changes were made in the environment by our system following a detected event, as described in **BR 14**.
- *Administrator interaction*
  - Administrator interaction, as needed in **BR 8**, would be enabled once the AWS command line interface environment has been set up on the laptop/ desktop of the administrator.
  - This would first require the AWS administrator, to create a role specific to the administrator based on the location of the smart-home environment under consideration and then associate the respective policies to his account.
  - The AWS administrator would then have to create a set of Access Key ID and secret access key for the local administrator while giving him CLI access to the smart home environments for a particular location.

- The local administrator would then use the secret access key and access key ID to set up the AWS environment up on his laptop/ desktop which would then enable him to connect with the things in the smart home environment under consideration.
- All of these steps cumulatively would help meet **BR 8**.

5.5.How to forward traffic from things?

Since the project is using MQTT messages for the devices in the environment, we are relying on the broadcast methodology and using the publish/subscribe models. In this model, a thing can be a publisher, subscriber, or both. A publisher would be sending messages to a specific topic, where only subscribers who have subscribed to this specific topic would receive messages from the publisher. MQTT is chosen because it is a lightweight protocol, flexible and scalable which satisfies **BR 3**.

For a mobile device, we can't use MQTT since is it is more like a client-server connection since the mobile device does not need to send data to other things all the time other than when the user actively using the mobile device to change the status of a device. Therefore, we'll use restful API calls to interact with the mobile device with our server. REST uses standard HTTP requests which are used popular and easy to learn, this will help developer implement the application easier. Also due to the separation of the client and server, our application becomes more portable between different type of mobile devices. This would be beneficial since our users could use varieties of mobile devices as stated in **BR 9**.

5.6.How to compute?

Here, we describe the two sets of computation that would take place in the project based on the use case of the device. The first set of computation would primarily happen at the edge, whereas the next set of computation would take place in the cloud.

- *Edge*
  - In this case, the computation would primarily take place
  - For example, in case of things with analog sensors, the translation of the analog voltage signals to the corresponding digital values can be computed at the edge itself, so as to reduce the latency of the transmission of only essential information to the AWS environment.
- *Cloud*
  - The next set of computation can happen at the cloud, where more computational power is needed for the processing of the received data.
  - Generation of the threshold values for different things and sensors depending on several attributes such as – location, time of the year, time of the day, and other weather-related information.
  - For example, in one of the applications where we use the image recognition service from AWS, a P3.2xlarge instance would help with the computing needs of the application. [16]
  - 8 vCPUs and 16 GB in GPU memory should ensure scalable classification of plant types [17].
  - The camera would send the image information over to AWS Rekognition and then use the computational power at the P3.2xlarge instance to recognize the plant or crop in the picture, and then use lambda function to not only store the plant information but also retrieve the water usage per week for the plant under consideration.

5.7.How to control things?

- *Using Lambda function*
  - Lambda function would essentially be used as triggering mechanisms to send out MQTT messages to the respective things which can also be a response to an MQTT message triggered by another thing in the environment.
- *Using a mobile device*
  - HTTP messages from the mobile and the web application to the AWS service triggering those lambda events, which would then trigger an MQTT message.
- Considering one of example in the smart home system – Irrigation system and the environment station.
  - The Environment station would send out the readings from the soil moisture sensor to the AWS message broker every 5 minutes, using the publishing topic that is specific to the soil moisture sensor.
    - This message being sent by the publishing client would be in form of MQTT messages being sent to AWS message broker.
  - The subscribing client on the AWS end would read the message and then would trigger lambda function which would trigger two functions simultaneously.
    - The lambda function would be responsible for storing the value received from the environment station and storing the value in the dynamoDB.
    - The lambda function would also be responsible for checking the value against the threshold value that has been defined in the database for the water level requirement for the particular plant that would be registered in another DynamoDB.
    - Once the value is checked and compared, if the value just received is less than the threshold value, the lambda function would trigger another lambda function.
  - This newly triggered lambda function would be responsible for sending an MQTT message from the publishing client on the AWS end via the message broker to the subscribing client on the users end such as the irrigation system. This message would help control the state of the thing in users environment.
  - The Lambda function would be responsible for sending an HTTP message to the mobile devices to update the state of the thing under consideration (irrigation system), from ON to OFF or vice versa.
  - A similar series of steps can be taken while the user is manually controlling the state of things in the environment such as the smart switches which would first involve sending HTTP messages from the mobile device to the AWS end, followed by triggering of lambda function to triggering corresponding events.

5.8.What are the building blocks of the architecture?

- *Things*
  - Discussed in Section 4.1

- *Data*
  - Discussed in Section 4.4

- *People*
  - Discussed in Section 4.2

- *Process*
  - Discussed in Section 4.3

- *Naming conventions and rules*
  - We can use the location of the device in the house along with the type and the identifier to uniquely identify a particular thing.
    - For example – *Regionname-Zipcode-SteertAdd-Apt-Floor-room-Thingtype-id*
    - Publish and subscribe topic can be the same as the name above.
  - This naming convention is optimal as it would help uniquely identify each of the things in the region. As required in **BR 3**, this also allows for scalability.

6. Alternate design methodology [5%]

In this section, we *try* to describe the entire project using the IDA architecture. In the lowest layer ***Sensor Layer*** *(Blue Layer)* of Figure 1. we have the different types of sensors which are a part of the **BR 1**, and **BR 14**. The sensors are responsible for measuring various aspects of the house in which they would be deployed. These sensors would not only be responsible to monitor the environment but also, collect and process the data collected at the edge and process it before sending it over to AWS for processing and storage. These sensors would also be equipped with connectivity such as WiFi using ESP8266, and other micro-controllers to allow for communication with the local network and to the AWS servers via the internet.

The next layer in the diagram is the ***Gateway and Networks*** *(Green Layer)*. This layer is primarily responsible for the communication aspect of the project with the outside world, that is, AWS.

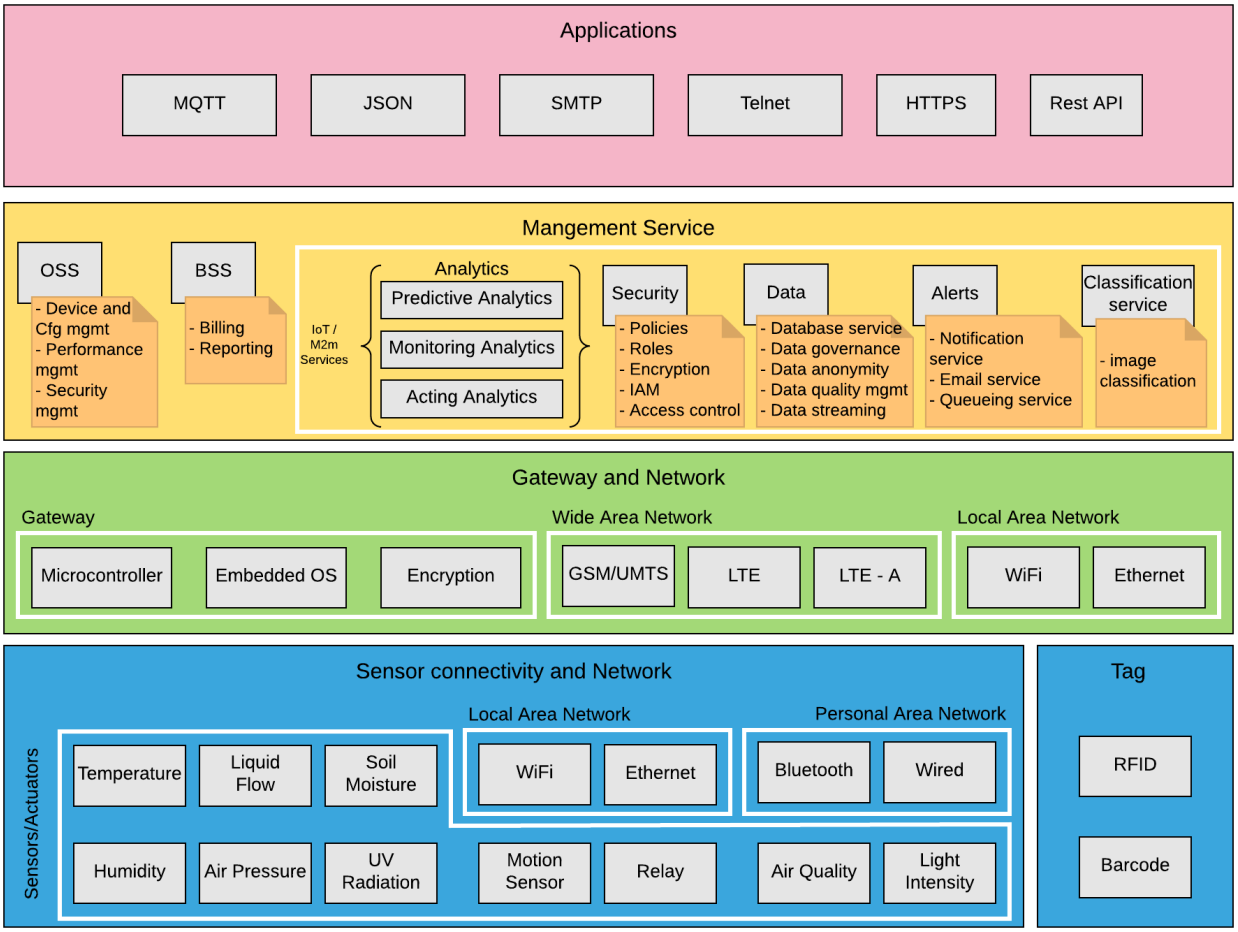

Figure 1: IDA Architecture

The micro-controller in our project (ESP8266) would be the gateway that would be responsible for enabling the sensors to connect to the local WiFi network to allow for the transfer of information. The LTE connectivity is a part of the project because of **BR 7**.

The ***Management Service Layer*** provides the basis for access to the services that would be a core part of the project as provided by AWS. It would comprise of all the services that are required as stated in **BR 4**, **BR 5**, **BR 8**, **BR 10**, **BR 11**, **BR14**, **BR 15**. This mainly comprises of the services associated with Analytics, Machine Learning, Security, Data, and Alerts. Finally, the ***Application Layer*** covers the *layer 5* application protocols that would be needed to achieve some of the business requirements stated for the project.

6.1.Limitations of IDA Architecture

Even though the IDA architecture, aims at providing comprehensive information about the project, it is not able to fulfill the requirement. The issues with IDA architecture is that it does not help understand the entire architecture of the project. The architecture fails at providing information about the modules within the layers that would be communicating with each other to pass information to other layers in the model. Also, while building the model, the *layer 5* ***application layer*** does not have enough information to clearly depict the relationship between the layers below associated with the application itself.

If the audience of this document were the developers of the project, they would complain that a lot of the information provided is ambiguous, because there are no specifications associated with each of the services provided. Also, there is not enough information associated with the protocols or implementation, which makes it difficult for any audience to understand the document enough to complete the implementation of the project.

For these reasons, we would now discuss the IoT-A architecture which would provide much more descriptive models to understand the project in a more complete fashion.

7. IoT-A System Architecture [20%]

7.1.Domain Model

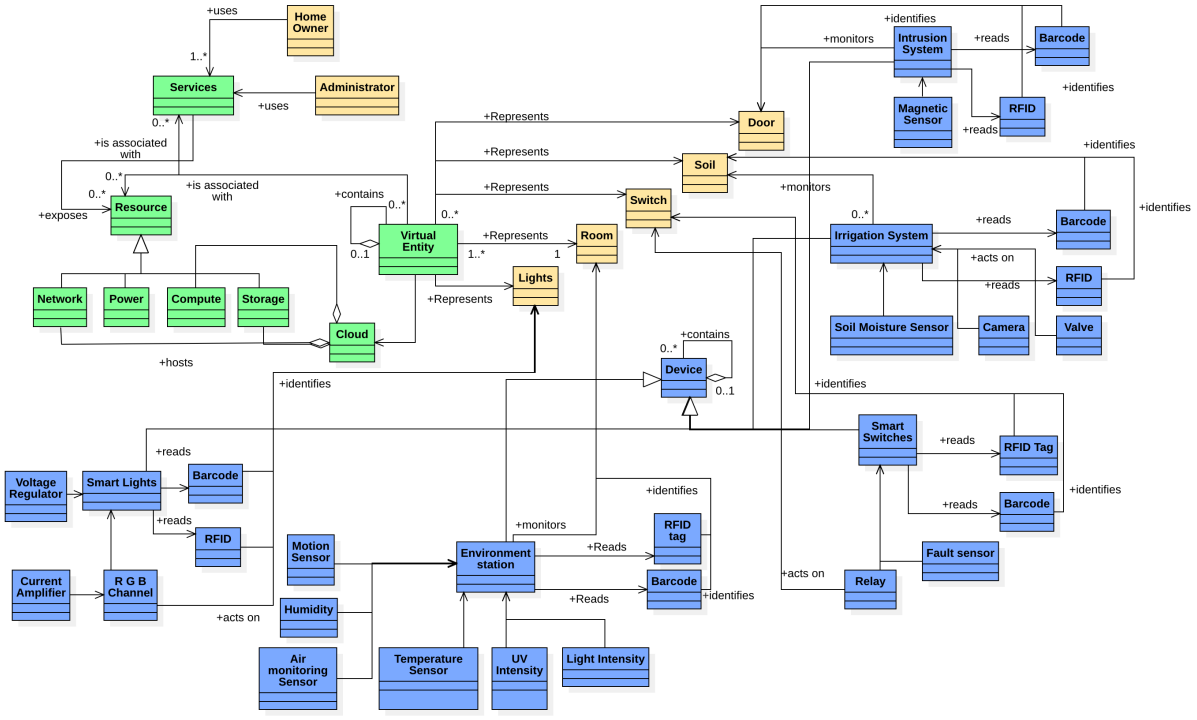

Figure 2: IoT-A Domain Model

In this model, we outline the physical entities and the users of the that are colored in yellow in Figure 2. We also show the Virtual entities that comprise of the cloud services such as the compute, network, and storage (Green Color). Each of the physical entities – Lights, Room, Switch, Soil, and Door have the associated sensors and actuators (Blue Color) that are responsible for monitoring the state of the respective physical entity and also responsible for managing the state of the devices in the environment using the actuators – relays, valve etc. Finally, we use two types of identification mechanisms as described in Section 5.1, that is, RFID and Barcode. This was done primarily to allow for both Users and administrators to register new devices using the steps described in Section 4.3.

Taking an example of one of the physical entity to describe the model – Switch. For this physical entity, the RFID Tags, and the barcode would be used to identify the thing for registration purposes. The fault sensor would be used to monitor the status of the smart switches to keep into account whether the device is operating or not. Also, this physical entity has an associated actuator, that is, the relay that would act on the switch to which it is connected.

Similarly, for other devices connected in the system, there are associated sensors and actuators that would help them operate as required in the Business Requirements.

7.2.Information Model

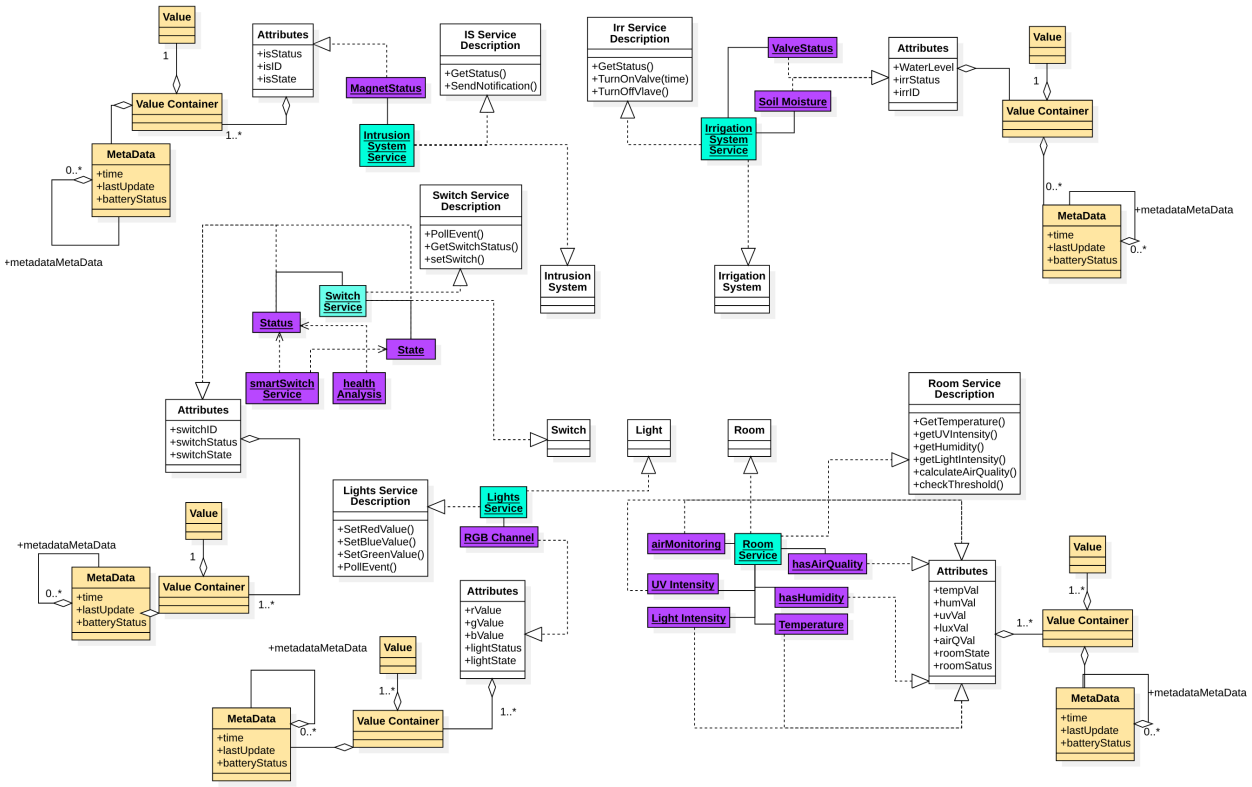

Figure 3: IoT-A information model

In this model, we describe the associated information generated and monitored by each of the entity in the system, with the set of functions each of the devices are calling, that is, we describe the structure of the system by specifying the Attributes (White) , Services (Purple), Service Description (White) and associated metadata (Yellow).

For each of the entity, we associate the attributes they would be responsible. For example, Irrigation System has the attributes for measuring the water/moisture level in the soil, and the irrigation system status. Which would have the corresponding metadata about when the measurement was taken, and when was the last update made to the database. The battery status information would help identify if the administrator needs to be informed about any faults with the system or to the User about whether they need to replace the batteries of the device.

Finally, each of the devices also has the associated functions that can call and perform. Taking the same example fo the irrigation system, we have to turn valve On and Off functions which would be responsible for reacting to the soil moisture thresholds as received from the AWS environment.

7.3.Physical Entity View

7.3.1.Room

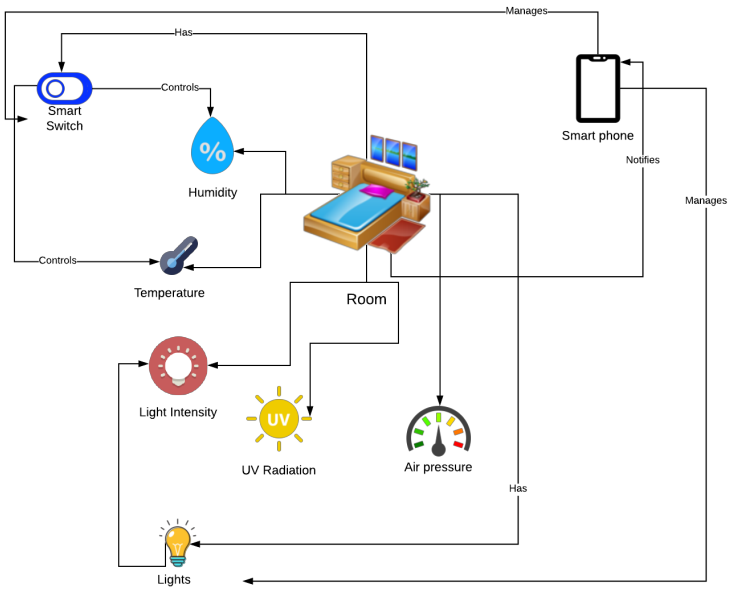

Figure 4: Physical Entity view for a Room

- **Sensors:** We have multiple different sensors to monitor the environment of the room. The measurements of these sensors are temperature, humidity, air quality, UV level, and light brightness.
- **Actuators:** We have switches that can change the states of different devices that control the environment of the rooms. The switches changed when their associated measurements go below or beyond its threshold. For example, if the temperature goes below 65, that would trigger a switch to change the thermostats setting to increase the temperature of the room.
- **Phone:** The phone is used to help the user interact with the room environment. The room would notify the user through the phone when there is a significant change in the conditions of the room either pre-defined or as defined by the user. Also, the user can set the temperature or light brightness of the room through the phone.

7.3.2.Door

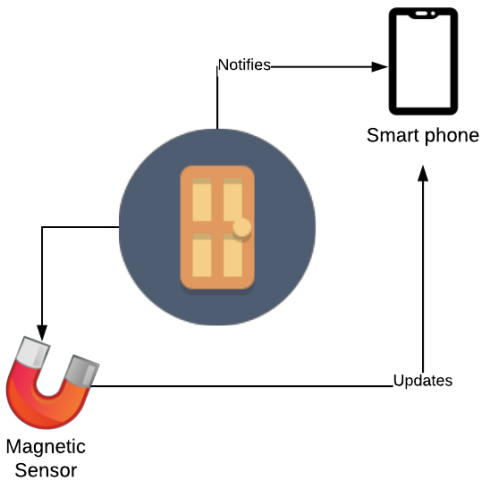

Figure 5: Physical Entity view for a Door

- ***Sensor:*** The magnetic sensor is used to track if somebody has entered the room.
- ***Phone:*** The phone is used to notify the user when somebody entered the room based on the preferences set by the user.

7.3.3.Plant

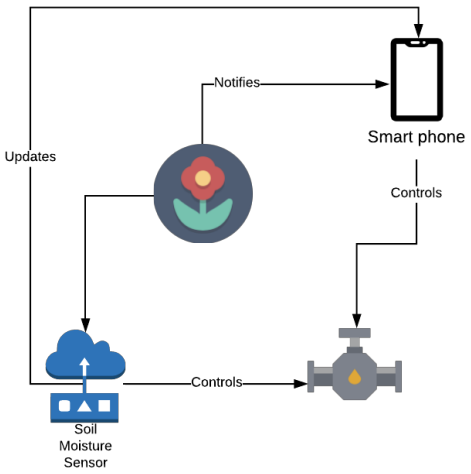

Figure 6: Physical Entity view for a Plant

- ***Sensors:*** soil moisture sensor is placed at the plants of in the room to monitor the water level of these plants.
- ***Actuators:*** valve/water sprinklers are used to fill up the water of a plant when its level water falls below the thresholds.
- ***Phone:*** Similar to the room, the phone is used to receive notifications about soil level or fill water for the plants

7.4.IoT context view

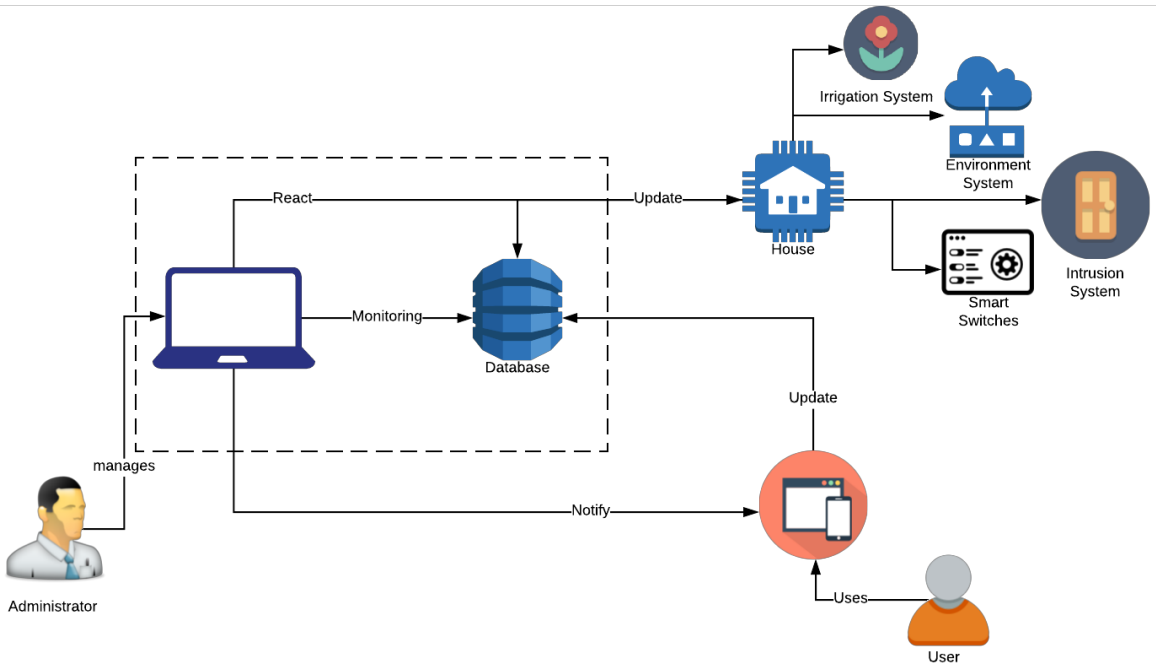

Figure 7: COSMOS' IoT context view

- The dashed rectangle represents our IoT system. Elements that interact with the system are user's house, user, and local administrator.
  - ***User:*** User interacts the system through the phone with a mobile application that can communicate the IoT System. The user can use the phone to either receive notification or update the house environment
  - ***Local Administration:*** Local Administration manage his responsible houses though CLI commands with his computer
  - ***House environment:*** The house environment interact with the system through sensor and actuators that are placed in the house. Measurements from sensors are stored in the system database to be analyzed, then the reaction to these measurements are taken place using the actuators in the house. The state of these actuators are also stored in the system database

8. Feedback Model [5%]

8.1.Setting goals

Make sure all of the devices in the house are operating correctly. If they are not, the user needs to be informed. The local administrator may set this goal since it's his/her responsibility to keep the system running.

8.2.Monitoring

Monitor the status of all devices by having devices sends their status to the cloud periodically. The period could be 1 minute, 2 minutes, or 5 minutes depending on the type of the device since different devices have their own configuration on how often data needs to be collected by the system. Only the latest status of a device should be stored, so we don't have much cost for storage in monitoring this. However, we might have a high cost for traffic as devices send packets through the network frequently. For other devices which do not send out periodic messages to AWS, like smart switches, there would be hardware proxies independent of the device hardware which would be responsible for sending out a signal to AWS in case the hardware fails.

8.3.Processing the Measurements

The required process is to accurately store the status of active devices and recognize whether a device has been down based on its deemed period of sending status.

8.4.Taking actions

When a device failed to update its status to the server, the user and local administrator of the house needs to be informed through notification systems or SMS message, so appropriate action can be taken. For devices sending out periodic messages to the AWS broker, if there are missed measurements for more than three times, we can assume that the device is no longer functional and a notification is triggered from AWS to the administrator to take necessary actions.

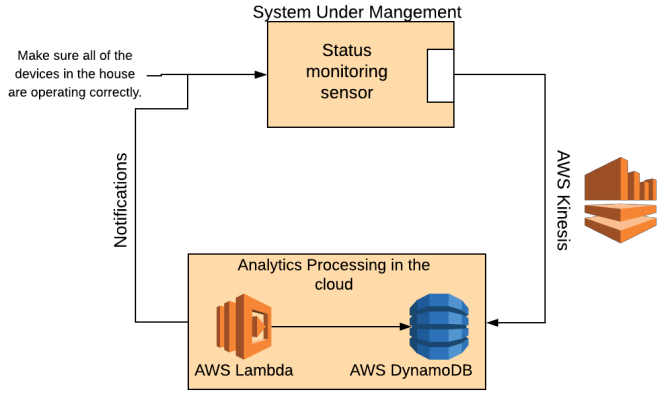

Figure 8: Feedback model

9. Analytics and computing [10%]

9.1.Monitoring Analytics

Taking into consideration **BR 1**, **BR 2**, **BR 4**, and **BR 14** one of the collective goal is to set up a visualization and monitoring service for the project. In this scenario, the System Under Management would be the environment station with all the different sensors that it encompasses. The set of sensors that would be a part of the environment station are Temperature and humidity sensor, air pressure sensor, air quality sensor, UV intensity sensor, and light intensity sensor. An example from the project for Environment station is shown in Figure 9.

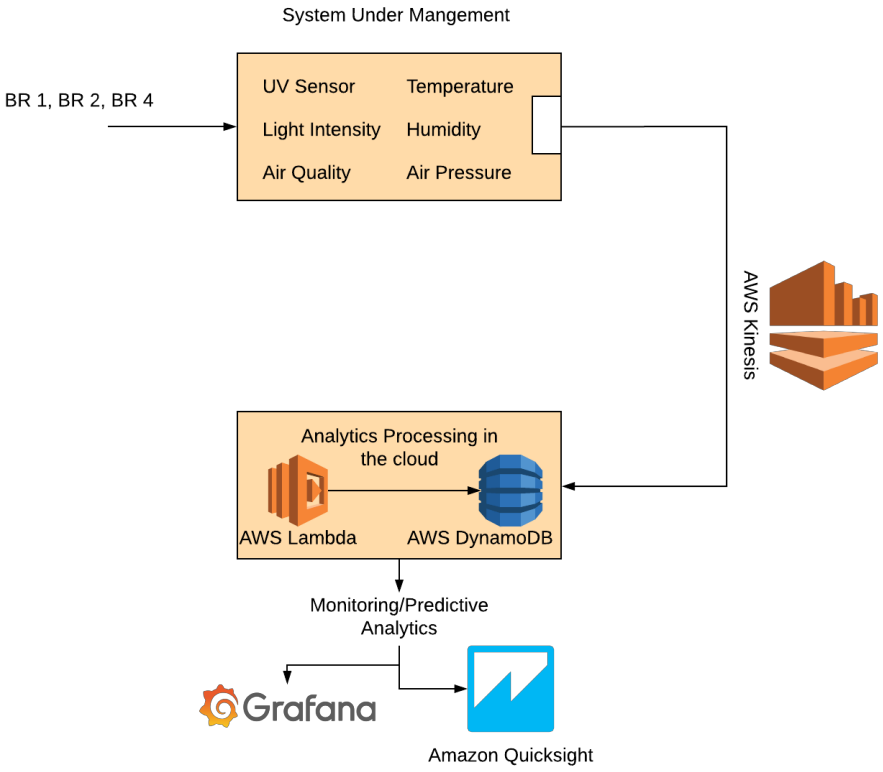

Figure 9: Monitoring analytics feedback model

In monitoring analytics, the data from the environment station would be passed via the gateway to the AWS Kinesis streams service which would be responsible for transporting the collected data to the AWS broker on the backend. This would then lead to the triggering of an AWS lambda function which would duplicate the data and send it to two separate kinesis streams. One path would be responsible for real-time analytics, whereas the other path would be responsible for storing of the data for predictive and query analytics, in either scenario the processed data would be passed on to either Amazon Quicksight or Grafana for visualization and other trend related predictions.

9.2.Predictive Analytics

Referring to Figure 9, from the monitoring analytics portion of the document, the process of transport of information from the devices to the AWS environment would be through the Kinesis service. The predictive processing would happen in the "processing section" of the model, where the lambda function would be used for storing the data in the Amazon DynamoDB from where the trend analysis would be processed.

An example of the predictive analysis would be to use the air quality sensors deployed in the house. If the network of sensors deployed in the house is taken into consideration for predicting whether a particular room has a higher concentration of a particular gas (Carbon Monoxide, Carbon Dioxide, Ammonia, Nitrogen Dioxide etc.) This would not include checking the current values against a threshold but predicting whether the rate at which a gas' concentration is increasing would it breach the threshold levels and if so predict this information for the user to see.

9.3.Acting Analytics

In the acting analytics portion of the project, the analytics being processed by AWS Lambda function would result in signals being sent back to the devices in the network of the home. These devices would be subscribed to the topics to which the lambda function publishes. The information being transferred would be transported via the Kinesis service again back to the devices in the network. The devices that would receive these signals would be the one with actuators integrated with them, for example, the irrigation system, smart switches, and smart lights.

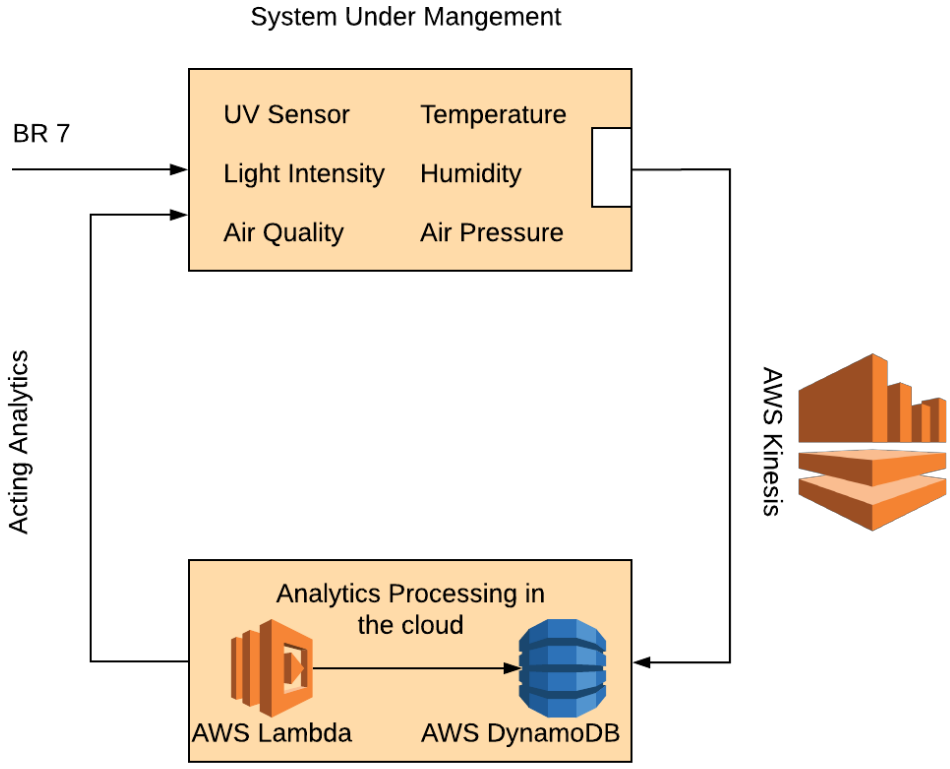

Figure 10: Acting Analytics feedback model

9.4.Other considerations for analytics

In Figure 11, we outline a simple AWS architecture for the project. This gives an example as to how the data would reach the cloud for processing. In this example, we take into consideration the Environment station which would be sending values to the AWS environment.

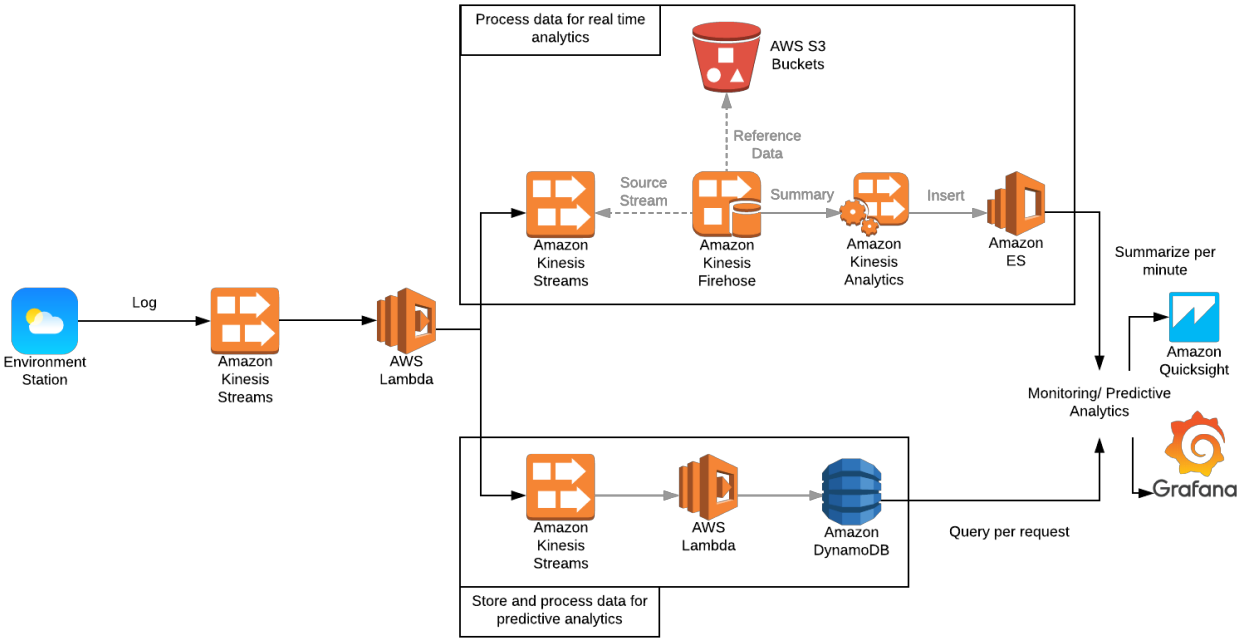

Figure 11: AWS Architecture for monitoring analytics

The Environment Station would be sending data to the AWS environment with the help of the AWS Kinesis Service. Once in the AWS environment, the AWS Lambda Function would be used to send the data across two different processes – processing data for real-time analytics, and for predictive analysis. For real-time analytics, the data would be stored in AWS S3 buckets, and also sent to Amazon Kinesis Analytics for monitoring analytics. Once the analytics is processed, the processed data would be sent to either Grafana or Amazon QuickSight. For predictive analytics, the data collected would be stored in Amazon DynamoDB for query-based requests. Once the required data is collected, it would be sent to an AWS Machine Learning service for further processing for predictive purposes.

9.5.Type of computing

We discussed this in Section 5.6

While choosing the cloud service that would be optimal for the project we had a look at several Service Level Agreements of AWS IoT Core[4], Google Cloud IoT Core[5], Azure IoT Hub[6] to make an informed decision. While considering the server downtimes, each of the cloud services provided an equal amount of credit refunds for similar percentage of downtime, the only difference was the maximum credit, which was offered by Google at 50% for less than 99% uptime.

We quote from [20] :

> ***Amazon Web Services (AWS) IoT:*** Pricing works on a fee per 1 million messages sent or received. There is also a free low throughput tier. Messages are processed in blocks of 512 bytes with each block representing one message up to a maximum of 128 KB. AWS IoT Core also connects to AWS' countless other cloud services, including AWS Lambda, Amazon Kinesis, Amazon S3, Amazon SageMaker, Amazon DynamoDB, Amazon CloudWatch, AWS CloudTrail, and Amazon QuickSight. This enables the building of IoT applications that can gather, process, analyze and act on data generated by connected devices, without DevOps teams needing to manage any infrastructure.
>
> ***Google Cloud IoT Core:*** Google IoT pricing is tiered as per the number of operations plus storage costs (and network costs if a consumer is located in a different region). Processing a typical message will involve three operations: 1 publish, 1 pull and 1 acknowledgement. Messages are split into 64 KB units that are each considered a single message for billing purposes.
>
> ***Azure IoT Hub:*** Pricing-wise, IoT Hub is grouped into four tiers, from a free tier up to the high throughput S3 tier, which is capable of supporting up to 300,000,000 messages per day. If more throughput is required, additional units can be added to each tier. Messages are sent in 4 KB blocks; for billing purposes, each block is counted as a message – up to a maximum 512 KB.

We were attracted at the wide range of the services offered by AWS' environment as compared to that provided by Google and Azure. Also, the pricing for IoT Core services[4] was very competitive and was found to be more cost effective than the other two.

[4] https://aws.amazon.com/iot-core/pricing/

[5] https://cloud.google.com/iot/sla

[6] https://azure.microsoft.com/en-us/support/legal/sla/iot-hub/v1_0/

10.The management plane [5%]

10.1.Management Tasks

This has already been discussed in Section 4.3.1 when discussing the registration of a device. This would be the task of the administrator for a particular area. Another management task, that we already discussed in Section 5.4, where we outline the task of enabling administrator interaction with the automation environment of a house. This would require the AWS administrator to give access to the local administrator to the services and Command Line Interface (CLI) access to the environment.

10.2.Management Solutions

The solutions to the above two management tasks regarding registration of a device, and enabling CLI for the administrator are discussed in the respective sections, where we outline the steps that have to be performed.

11.Virtualization [5%]

In the project, we chose to use the AWS compute services for hosting the servers, and also for machine learning purposes.

- For example, in one of the applications where we use the image recognition service from AWS, a P3.2xlarge instance would help with the computing needs of the application.
- 8 vCPUs and 16 GB in GPU memory should ensure scalable classification of plant types
- The camera would send the image information over to AWS Rekognition and then use the computational power at the P3.2xlarge instance to recognize the plant or crop in the picture, and then use lambda function to not only store the plant information but also retrieve the water usage per week for the plant under consideration.

11.1.Virtualization Goals

For other applications, each of the houses would be allotted an EC2 instance. As described in **BR 1**, **BR 3**, **BR 7**, **BR 10** and the associated Technical Requirements, that is, **TR 2**, and **TR 5**; the C5 instance should be optimal for the use cases required in the project. The C5 instance is a compute-optimized EC2 instance, which would be sufficient. The reasoning behind is that, for all the data processing, a C5.xlarge instance with 2 vCPUs, and 8 GiB of memory would be more than enough for handling the workload that a single household would require. Moreover, it would have a network bandwidth of 10 Gbps. Assuming, that each house has 10 sensors, sends out 17,000 messages every day (1 message every 5 seconds), in addition to the compute performance as required by the machine learning algorithm running on a P3.2xlarge instance, the network bandwidth would be able to handle the workload with ease. The main takeaway from the virtualization environment being set up in the project would be that each of the houses would be in a private subnet, communicating with a public subnet that would be a part of a small region. Figure 12 outlines the network architecture for a particular area (one particular street), which would be scaled up for each of the regions.

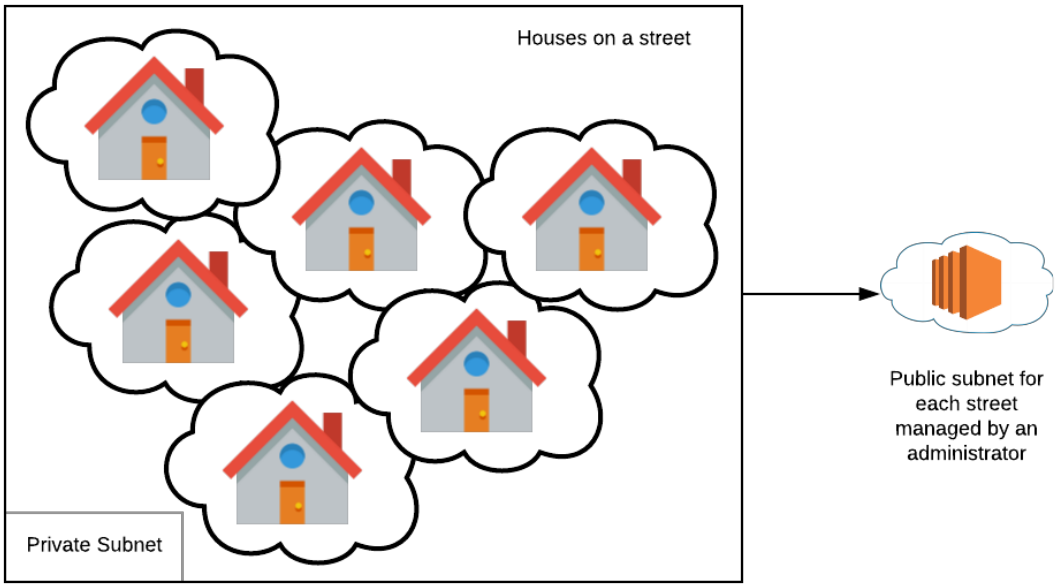

Figure 12: Network Architecture example for a street

11.2.Using Virtualization in the project

For allowing the private instances to connect to the internet to enable transport of the sensor readings to the AWS environment, we would be using the NAT instances connected to the public subnet. The alternate design consideration was the NAT gateway, but we instead chose NAT instances, because the latter is slowly being deprecated, and moreover because NAT instances allow for manual configuration of security patches and updates, as stated in **BR 8** and allows for transfer of data from the DynamoDB to S3 as required in **TR 5** without traversing the internet.

A simplified version of the virtualized network and connections are drawn in Figure 13 which outlines how the houses in the private subnet would be connected to the NAT instance sitting in the public subnet. The house instances in the private subnet sends the traffic from the instances in the private subnet to the NAT instance in the public subnet. The NAT instance sends the traffic to the Internet gateway for the VPC. The traffic is attributed to the Elastic IP address of the NAT instance. The NAT instance specifies a high port number for the response; if a response comes back, the NAT instance sends it to an instance in the private subnet based on the port number for the response.[7] This would allow for the transfer of data from the instances to the DynamoDB to S3, while still keeping the house instances inaccessible from the internet as stated in **TR 5**.

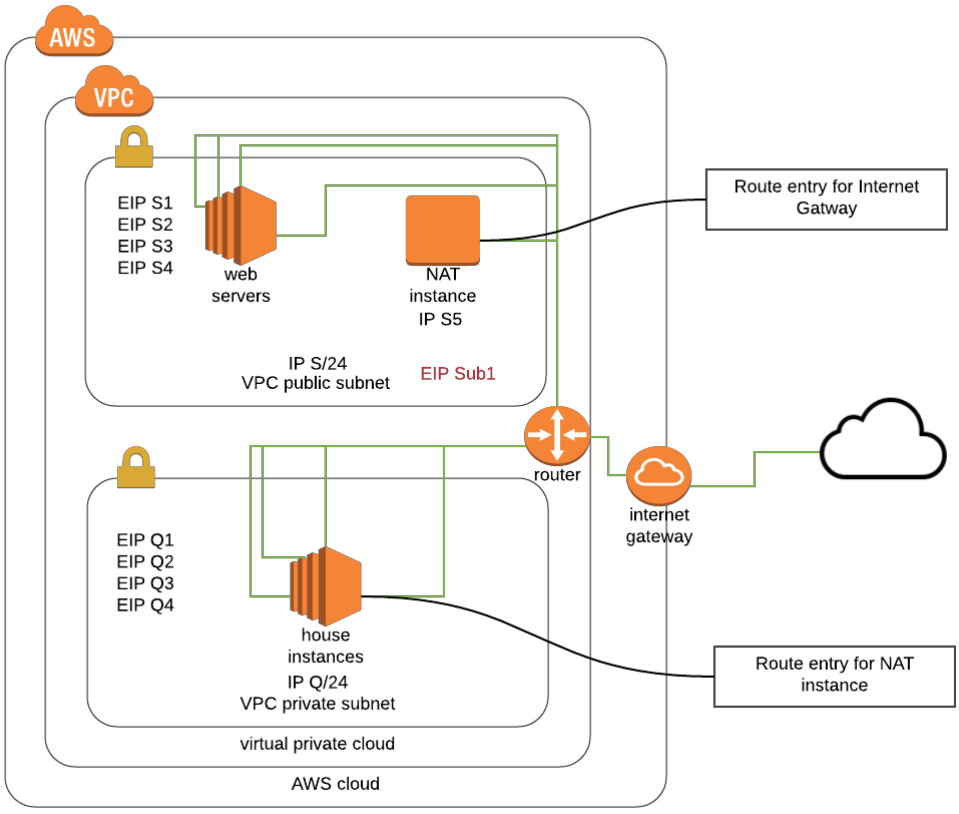

Figure 13: AWS architecture to show the virtualization of networks using NAT instances

7 https://docs.aws.amazon.com/vpc/latest/userguide/VPC_NAT_Instance.html

12.Conclusion

12.1.The lessons learned

As described in the Introduction Section, this project is a precursor to the implementation of a project done by one of the authors The lessons that were learned as a part of this project were understanding the architectural implementation of an IoT system. Getting exposed to different architectural methodologies such as the IDA Architecture, or the three-tier model, helped understanding how they would not appropriately describe the IoT system for a developer to get a complete idea for implementation of the system. This understanding was possible, because of the prior implementation was done by the author. However, when carefully studying the architecture, it became more and more apparent that a more detailed architecture such as the IoT-A architecture was needed to standardize the approach to a problem that was being tackled. This shift in the adoption of a standardized architecture helped to understand and take into consideration other aspects of the project such as the business and technical requirements. This, in turn, helped make the project more scalable and more secure using the virtualization aspect of the project as outlined in Section 11.2. Also, getting to know about the definitions of the elements of the IoT system helped better understanding the IoT system from the perspective of an architecture and designer.

Also, we learned about different AWS services that can be integrated with the project, that would help in consolidating the use of Machine Learning aspects into the project, which was a future scope in the project[8] that was described in Introduction Section. In this project, we were able to explore several services associated with AWS – Rekognition, and IoT Core services, and other core services such as Elastic Cloud Compute (EC2) and Simple Storage Service (S3).

12.2.Future work in programming class

In the future programming class, we would try to implement the machine learning aspect of the project. The idea would be to use a classification model implemented in the AWS Rekognition service to classify the type of plant that the user has in his house, then this information would be used to retrieve information from the DynamoDB about the water requirement. Then this information would be used to actuate the irrigation system in the environment which would be done using AWS Lambda function. Other aspects that could be implemented would be to integrate the cloud services into the existing project that is linked below[5].

[8] https://github.com/jubeenshah/COSMOS

References

(1) "COSMOS: ubiquitous automation solution" [Online]. Available: https://github.com/jubeenshah/COSMOS/blob/master/06_FinalDocumentation/13_FinalDocumentation/COSMOS.pdf
(2) "COSMOS: Concept and plan" [Online]. Available: http://www.jubeenshah.com/2017/10/
(3) "Amazon AWS" [Online]. Available: https://aws.amazon.com
(4) "ESP8266EX Datasheet" [Online]. Available: https://docs.aws.amazon.com/application-discovery/index.html#lang/en_us
(5) "DHT 22 Datasheet" [Online]. Available: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf
(6) "BMP 180 datasheet" [Online]. Available: https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf
(7) "Light sensor datasheet" [Online]. Available: https://www.mouser.com/ds/2/348/bh1750fvi-e-186247.pdf
(8) "MQ2 Sensor datasheet" [Online]. Available: https://www.pololu.com/file/0J309/MQ2.pdf
(9) "PIR Motion sensor" [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf
(10) "Soil Moisture sensor" [Online]. Available: https://www.mouser.com/ds/2/744/Seeed_101020008-1217463.pdf
(11) "ML8511 UV sensor datasheet" [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/LightImaging/ML8511_3-8-13.pdf
(12) "Magnetic Door sensor" [Online]. Available: http://www.itwatchdogs.com/datasheets/MS-1%20-Magnetic%20Door%20Switch%20datasheet%20(v1.06).pdf
(13) "ESP8266 Camera setup" [Online]. Available: http://www.arducam.com/downloads/ESP8266_UNO/ArduCAM_ESP8266_UNO_DS.pdf
(14) "InfluxDB for time series data" [Online]. Available: https://www.influxdata.com/products/
(15) "Grafana: Visualization of time series data" [Online]. Available: https://github.com/grafana/grafana
(16) "EC2 Instance types" [Online]. Available: https://aws.amazon.com/ec2/instance-types/
(17)Zhu L, Li Z B, Li C, Wu J, Yue J. High performance vegetable classification from images based on AlexNet deep learning model. Int J Agric & Biol Eng, 2018; 11(4): 217-223
(18) K. Muheden, E. Erdem and S. Vançin, "Design and implementation of the mobile fire alarm system using wireless sensor networks," 2016 IEEE 17th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, 2016, pp. 000243-000246. doi: 10.1109/CINTI.2016.7846411.http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7846411&isnumber=7846370
(19) "Architectural Principles of the Internet" [Online]. Available: https://www.ietf.org/rfc/rfc1958.txt
(20) "AWS IoT vs. Google IoT vs. Azure IoT" [Online]. Available: https://www.bizety.com/2018/08/28/aws-iot-vs-google-iot-vs-azure-iot/