IOT Based Smart City: Weather, Traffic and Pollution Monitoring System

Swapnil Shah, Ketan Deshpande, Dr. R. C. Jaiswal

---***---

Abstract: *There is a room for an improvement when real time data monitoring is concerned. We are looking forward to give the necessary information of various aspects of a particular city like its weather, traffic and the level of pollution. No single platform gives us the information about these things on a single click. Furthermore, time is a crucial aspect of urban lifestyle. This paper will provide a prototype which will provide real time traffic scenario in different parts of a city in order to avoid your hectic journey also with the help of sensors and raspberry pi , real time data will be fetched and displayed on a web page. These values will be real time values of actual weather conditions which you can't get on internet (generally those are predictions).*

1. Introduction:

Internet has become an inevitable part of urban life. Using internet was a big deal twenty years before but today even a child can use smartphones and laptops easily. Popularity of 3G, 4G technologies, their affordable rates and ease of their use ultimately have resulted in large number of Internet users today. As internet is used for interaction with other users, through various platforms such as Facebook, twitter, mails and what not, we come across the concept called **Internet of Things**. The term itself is self-explanatory There are different means by which various electronic instruments can communicate with each other. This can be used for ample of applications. Eventually perfect implementation of such system which can communicate with each other can be used to create an efficient, error free system due to human interactions. There are limitations to the extent up to which a man can work on the other hand with proper maintenance and handling of machines, there is theoretically no limit on the productivity of a machine. It is estimated that there will be around 20.5 billion devices connected to internet by the end of 2017. The principle of internet of things can be applied in healthcare, industry, education so on and so forth.

2. Idea and implementation:

As mentioned earlier, this paper illustrates on a prototype which will give weather, pollution and traffic conditions of different parts of a city on a single click.

Sensors: Humidity ,Temperature ,Raindrop ,Carbon detection (pollution), Noise sensor
Processors: Raspberry Pi

HTML, CSS for the front end of the project for developing the site which will play role as application layer that is the actual communication between user and the interface.
AWS Server: For backend of the project.

2.1 Hardware Requirements:

The hardware of this monitoring system mainly consists of three sensors and raspberry pi. The sensors are

1 DHT11(Temperature and Humidity sensor)
2 Raindrop sensor
3 MAX4466(Noise sensor)

2.2 Software and Other Requirements:-

- **Amazon Web Services:**

Amazon web services is a cloud platform which is secure which offers various tools for implementation of compute power ,database storages, services like content delivery and other functionalities to help businesses thrive. Millions of customers are using these web services. One such service is AWS IoT. This service is being used for the prototype.

AWS IoT:

AWS IoT has prime functionality of creating bidirectional communications with devices which are connected over internet (sensors, actuators, embedded devices, or smart appliances) to the AWS cloud. This gives us privilege of collecting telemetry data from multiple devices and store and analyze the data . Appropriate applications also can be created so as to eneble users to control these devices from their phones.

AWS IoT consists of following components :

1. Device gateway: This enables devices to securely and efficiently communicate with AWS IoT.

2. Message Broker: This provides a secure mechanism for devices and AWS IoT applications to publish and subscribe message from each other. In this case MQTT protocol can be used in two ways. First is to use MQTT directly and second is to use MQTT over WebSocket to publish and subscribe. HTTP REST interface can be used to publish.
3. Rules Engine: This provides integration of messages to other services like Amazon S3, Amazon DynamoDB, Amazon Lambda.
4.Security and Identity service
5. Thing Registry: It is also referred as device registry which organizes the resources with each thing.
6. Thing Shadow : Also can be reffered as device shadow. It is a JASON document which I used for storing and retrieving the current state of information for a thing (It may be device or app etc.)

Protocols : a)MQTT b)HTTP c)MQTT over

Web socket: This is available on TCP port 443. This allows message to pass through most firewalls and web proxies.

- **Python:** We have used python programming language to take serial data from raspberry pi. In python there are modules for mqtt, websocket programming. We have just imported them and used them.So python is being used to connect to aws thing, implement mqtt, websocket and to publish the message over cloud and subscribe it.

- **HTML and CSS**: We are using basic web designing language HTML (Hypertext markup language) for fundamental implementation of web page. It forms a general structure of web page and will be helpful for basic layout.As we need to make user interface more user friendly and attractive we need to go for CSS . CSS provides some unique features

2.3 DynamoDB database :

It is noSQL databases which have better scalability and speed. There is no need for software pitching, hardware provisioning, replication and set up and configuration as well as cluster scaling . Any amount of data can be stored and retrieved from database. DynamoDB automatically adjusts the data through various servers for handling of throughput and storage requirements. A partition key is used as a primary key in order to sort data in order. Read operation and write operation on database is carried out with several steps.

3. Implementation and Execution.

We connected all the sensors to controller . The power requirement of all sensors provided by RaspberryPi itself. After that writing a code for particular sensor we got the data on the RaspberryPi

Now, we get data in the raspberry pi. We have to upload this data on the cloud. As mentioned earlier we are using AWS (Amazon Web Services) for that. In AWS we are going to use their IOT service. We have to handle that data from the sensor and for that we are going to use non structures database like DynamoDB.
So, the flow of our project is as follows,

3.1 CREATE AWS IoT thing

We have created account on AWS. We have selected AWS IOT service on AWS.

In AWS IOT we have created 'Thing' which is as if raspberry pi on the cloud. We downloaded all the private, key's certificates of the thing. They are very important in order to interact with the thing, to connect to it and all. Also this certificates are not supposed to be shared at all.

- Next step is creating the policy, this policy signifies which functionalities you are going to have on your thing. Then, attach certificates to the policy and policy to the thing.
- You are done with establishing thing. Now, we have to think of like what should be done when any message is sent to this AWS thing.

3.2 Create an IAM Role for AWS IoT

In order for IoT to interact with other AWS services, you need to create an IAM role so that it has the right permissions. Copy the following code into a file and call it trust-policy.json.

Then to create the IAM role, run the create-role command giving it the Assume Role policy document that you just created:$ aws iam create-role --role-name iot-actions-role --assume-role-policy-document file://path-to-file/trust-policy.json

Make sure you save the ARN from the output of this command as you'll need it to create a rule later on. (rules enable you to access other AWS services through IoT)

3.3 Grant Permission to Role

Later on we're going to go through a couple of examples for using the AWS IoT service, one to post data to a DynamoDB table and then one to invoke a Lambda function. To do this we have to create a policy and then attach it to the role we created in the previous section. Copy the following code into a new file and name it whatever you like. We've called it iam-policy.json:

Run the create-policy and link to the file path you just created:

```
$ aws iam create-policy --policy-name iot-actions-policy --
policy-document file://IAM-policy-document-file-path
```

Make a note of the ARN that is returned in the command line and then run the attach-policy-role with this command:

```
$ aws iam attach-role-policy --role-name iot-actions-role --
policy-arn "policy-ARN"
```

You should now be able to interact with DynamoDB and Lambda!

3.4 Creating a Table and Rule to insert message into DynamoDB

Create a table in the DynamoDB console

Create a rule to trigger on a topic of your choice and insert an item into DynamoDB. Add the following to a file and call itdynamoDB-rule.json. Copy the arn from the iot-actions-role we created earlier for the 'roleArn':

Create a topic rule using the create-topic-rule command with the path to the DynamoDB rule from the previous step:

```
$ aws iot create-topic-rule --rule-name saveToDynamoDB -
-topic-rule-payload file://path-to-file/dynamoDB-rule.json
```

Now, we just need to create the rule to evoke lambda function and configure it.

With this we done with setting AWS thing, dynamoDB and lambda function.

3.5 Implementing MQTT with web socket programming

We are using python for Websocket connection with the server first using the socket library in it.For mqtt we have to use paho.mqtt.client in python and by giving proper certificate and details of the AWS thing we can publish reading of the sensor on the cloud.We can also see that reading in the table in AWS dynamodb console.

3.6 Extraction of data from dynamoDB

We are focusing on extracting the data from dynamoDB that is provided by sensors and which is real time data to our web page. This should be very efficient as the data is real time and need to be displayed instantaneously. We have programmed certain functions which would perform this operation. The updated web page will be a structure like stack in which the new data will be overwritten on new data and will be displayed.

3.7 Implementation by Dynamodb:

The above implementation was by AWS IOT. We can also implement this all by just using the dynamodb database of AWS. This implementation is as follows,

We have all our sensor and raspberry pi with us and we have to design our system. Let's divide our whole system in 3 parts.

1] The hardware interfacing.
2] The database.
3] Front end design.
Let us take a look of it one by one

3.7.1 The hardware interfacing:

- We have to interface all the sensor to raspberry pi. Actually raspberry doesn't have analog pins so we took all digital sensors only.
- To interface the sensor first thing we have to do is, choosing the mode of pin assignment. By the GPIO.setmode(GPIO.BOARD), we chose the BOARD mode. It means that we can use all pin numbers on board as it is directly in the program.
- we have to assign pins as input as we have to take data from the sensors. This can be done by GPIO.setup(11, GPIO.IN) where pin number 11 in the input now.
- For the DHT11, we have used library directly so we have to import in in the code. It's functions

directly returns temperature and humidity.

- Rain sensor directly gives output as 1 if rain is there on that particular input pin and 0 if not.
- So it was all about the interfacing the sensors and we will get all data from them on the raspberry pi.

3.7.2 The database:

So far we got data on raspberry pi from the sensor. Now we want to use some kind of database which will be storing this data and we must be able to retrieve the latest entered value from the database. Here, in our project we have used dynamodb as our database. It is provided by AWS only. To use this dynamodb is very simple and now we will take a look at it.

Our backend will be like, for our each hardware module we will have different table having different names. Data will be uploaded from raspberry pi in that particular table continuously. On the web-portal

Uploading data in Dynamodb:

- We have used python to upload data from raspberry pi to Dynamodb table. We did this by using python-aws connector named 'boto'.
- To authorize the aws account we have to provide the Secrete key and Access key on this python code.
- To have synchronization i.e. by entering name of place user must get the only the latest data updated in that table, we have to give little thought to it.
- From the system time we calculated one value and we gave it as sortkey of our dynamodb table. Reading of all the sensor along with this key named as 'time_t' we entered in the table.
- That part was quite easy, we just need to select particular table and define one tuple and enter all the attribute and its value in that table. You will understand it by taking look of code.

Dynamodb table:

- The table will have different column for different sensor along with one primary key and sort key.
- As such we don't need primary key here but we need sort key to compare the time while retrieving the record value. In our case time key is named as 'time_t'.
- The attribute of the tables are delta(primary key), time_t (sort key), temperature (attribute), humidity (attribute), rain (attribute).
- This is the dynamodb table which is ready to store data.

Retrieving data from the table:

- Now we have our table ready on AWS server whenever user refreshes webpage the latest value will be retrieved from the database. Let's see how exactly this happens.
- Here we are using jsp as a web technology so that we can execute java code to get data from table and display is to user.
- There are connectors in java to access the dynamodb table. By making use of them we are getting proper table the one which is entered by the user and then by calculating that value of time_t from the system time we retrieve that one particular record having data of all sensors and we just display it.

Results:

- After designing this system, we will be able to upload data on server which we get on Raspberry Pi. The output for all the sensors is in directly in integer form.
- The accuracy we got for sensors was 85% initially. After using the cross validation set it increased up to 94-95%. Initally we used port 1883 to transfer data on the server but then we came to know about more secure port connection. So now we are using port 8883 for secured connection.
- Thus, our result is more efficient and can be used for further extension of prototype. We have experimented our system with different sensors and at the end we selected the sensors according to their accuracy. The data from the sensors is just in the number format which further pass to raspberry pi for further processing. The data feed into the system and then passed through several pre-processing steps before it can be recognized.

We are able to monitor the traffic conditions, weather status and pollution conditions on web page of different parts of city. This helped us not only to analyze and study the parameters but also proved how efficiently real time data can be displayed with significant error minimization.

Reference:

[1] J. Jin, J. Gubbi, S. Marusic and M. Palaniswami, "An Information Framework for Creating Smart City Through Internet of Things," in IEEE Internet of Things Journal, vol. 1, no. 2, pp. 112-121, April 2014.

[2] F. Zhu, Z. Li, S. Chen and G. Xiong, "Parallel Transportation Management and Control System and Its Applications in Building Smart Cities," in IEEE Transaction on Intelligent Transportation Systems, vol. 17, no. 6, pp. 1576-1585, June 2016.

[3] Aceves, E.; Larios, V. M. "Data Visualization for Georeferenced IoT Open Data Flows for a GDL smart City Pilot" IEEE Guadalajara GDL CCD White Papers, 2015.

[4] Mohammed Atiquzzaman "Special Issue on Internet of things:Smart things network and communication" Spring 2014.

[5] S. Misbahuddin, J. A. Zubairi, A. Saggaf, J. Basuni, S. A-Wadany and A. Al-Sofi, "IoT based dynamic road traffic management for smart cities," 2010 12th International Conference on High-capacity Optical Networks and Enabling/Emerging Technologies(HONET), Islamabad, 2015, p p. 1-5.

[6] http://www.arduino.org/?gclid=Cj0KEQjwvve_BRDmg9Kt9ufO15EBEiQAKoc6qvpeQ90AOeyTH2oEq83P6b5VTJ1HBOJP0dJuCjfVTlQaApN68P8HAQ

[7] https://www.raspberrypi.org/

[8] https://thenewboston.com/

[9] The Second Machine Age:Work,Progress and Properity in a Time of brilliant technologies. By Erik Brynjolfsson and Andrew Mcfee.

[10] Getting Stareted with IOT by Cuno Pfster.

[11] The Silent Intelligence by Daniel Kellmereit and Daniel Obodovski.

[12] IOT Disruptions:The Internet of Things-Innovation &jobs by Sudha Jamate.

[13] Meta Products:Building the Internet of things by Wimer Hazenberg.

[14] Designing connected products by Claire Rowland,Elizabeth Goodman,Martin Charlier.