Monitoring pollution level in a smart city

By

Kshitij Deshmukh
(19MCEC08)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Ahmedabad 382481

Monitoring pollution level in a smart city

Minor Project

Submitted in fulfillment of the requirements

For the degree of

Master of Technology in Computer Science and Engineering

By

Kshitij Deshmukh
(19MCEC08)

Guided By
Dr. Vijay Ukani
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Ahmedabad 382481

CERTIFICATE

This is to certify that the Minor Project entitled "Monitoring Pollution level in smart city" submitted by Kshitij Deshmukh, towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering of Nirma University is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Dr. Vijay Ukani
Associate Professor
Department of Computer Science & Engg.,
Institute of Technology,
Nirma University,
Ahmedabad

Dr. Madhuri Bhavsar
Dept. of Computer Science & Engineering,
Institute of Technology,
Nirma University,
Ahmedabad

ACKNOWLEDGEMENT

I would like to extend my thanks to the people who supported and cooperated during my Minor Project phase.

First, I would like to express my deep and sincere gratitude towards my project guide Dr. Vijay Ukani, Associate Professor, Computer Science and Engineering Department, Institute of Technology, Nirma University, for providing an opportunity to work on development of this project.

I would also like to thank all my friends for the support and guidance during the development of this project.

ABSTRACT

IoT has evolved the day to day objects to smart objects. These objects can be then controlled by humans when they are connected via internet. Its practical implementation came into existence few years back. Since the advent of IoT, it has tackled many challenges such as management, tracking, observation etc. It is not only used in industrial sector but contributed in fields such as smart city, smart home, smart cars and many more. One of the main concerns in present times is environmental degradation especially the quality of air we breathe in. There has been extensive study on developing monitoring applications to get hold of air quality outdoors with help of sensors that are deployed in the outside environment. The project focuses on monitoring the pollution level in any given area at any time of day and the sensors would interact and co-ordinate with the nearby sensors which would then be analyzed and processed.

CONTENTS

List of Figures

List of Tables

Chapter 1 – Introduction

1.1 Air Pollution Monitoring

One of the smart solutions to monitor the air quality in a smart city was by using Internet of Things. The sensors would sense and get the readings from the atmosphere in digital or analog form that would be converted to a suitable reading after calibration of the sensors. Different sensors are available to detect different gases present in the atmosphere like NO2, CO2, CH4, NH3 and also take temperature and humidity readings. Also, gas sensors can be calibrated to detect specific gases only. For instance, MQ-135 sensor when calibrated for CO2 can detect only the presence of CO2 in atmosphere. After these readings are taken from the sensor, these are sent to cloud for further processing. Sensors have limited processing capabilities. So, with the help of a gateway, like Raspberry pi 3, which has enough computational power is sufficient and network communication protocols like 2G, 3G, 4G-LTE, Narrowband-Internet of Things (NB-IoT), SigFox, Long Range Wide Area Network (LoRaWAN) can be used. The data is processed in cloud which is then sent back to the consumer end in form of information on their application interface.

Government bodies define air quality standards differently. Countries have different standards from others. The air quality standards are defined as color coding scheme called Air Quality Index as per [3]. The value ranges from 0 to 500.When AQI is in range of 0 to 50 air quality is symbolized by green color and is regarded as good quality whereas when it exceeds 150 it is considered to be harmful. Figure 1.1.1 shows one of the air quality standards adopted.

AQI Category, Pollutants and Health Breakpoints								
AQI Category (Range)	PM_{10} 24-hr	$PM_{2.5}$ 24-hr	NO_2 24-hr	O_3 8-hr	CO 8-hr (mg/m^3)	SO_2 24-hr	NH_3 24-hr	Pb 24-hr
Good (0-50)	0-50	0-30	0-40	0-50	0-1.0	0-40	0-200	0-0.5
Satisfactory (51-100)	51-100	31-60	41-80	51-100	1.1-2.0	41-80	201-400	0.5 – 1.0
Moderately polluted (101-200)	101-250	61-90	81-180	101-168	2.1- 10	81-380	401-800	1.1-2.0
Poor (201-300)	251-350	91-120	181-280	169-208	10-17	381-800	801-1200	2.1-3.0
Very poor (301-400)	351-430	121-250	281-400	209-748*	17-34	801-1600	1200-1800	3.1-3.5
Severe (401-500)	430 +	250+	400+	748+*	34+	1600+	1800+	3.5+

Figure 1.1.1: Air Quality Index standard

This IoT solution will monitor the pollution level in a smart city by using IoT devices and sensors as it is important to tackle the problem of harmful emissions and control it. Multiple stations(sensors) will be deployed across the cities. These stations will sense the harmful pollutants present in the air and forward this data to the edge devices present. With the help of semantic web technologies, we can interpret and integrate the data coming from these various devices. We describe these domains of knowledge and concepts, called as ontology, to give meaning to data. Data interpretation and analytics is also one of the goals to achieve in such an environment. After the devices have collected the data and analyzed it, the devices will give the feedback to client through cloud which can be routing the client to less polluted way, sending critical information like the harmful pollutants and emission rise observed at any particular place. The general structure of air pollution monitoring system is shown in figure 1.1.2.

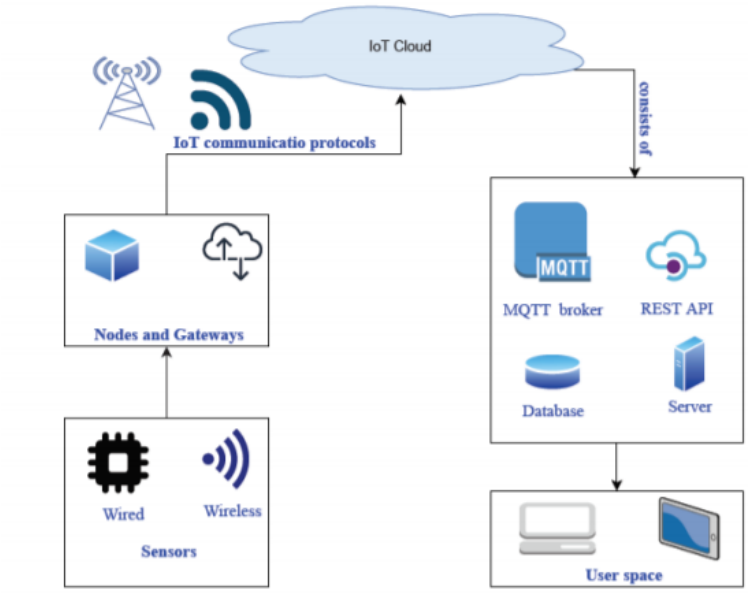

Figure 1.1.2: General structure of air pollution monitoring system

1.2 Semantic Web

The term Semantic Web was coined by Tim Berners Lee. Today, the web consists of huge repositories of data. The objective is to help people find answer to their questions based on knowledge/data available on the web. Semantic Web involves relationship among data. The purpose was to move from web of documents to web of data and start connecting the data available on the web. The framework of semantic web is as shown in Figure 1.2.1. Semantic web identifies data on web using Uniform Resource identifiers (URIs) [16].

URI is simple scheme to refer any data available on the web. Individual things/data/resource on the web can be identified uniquely with help of an identifier. It is acts as a namespace in XML language. XML language is used because it is easy to connect languages together.

Resource Description Framework (RDF) is a data model for representing any resource available on the web. This enables the machine to exchange information easily, i.e. it is machine readable. The data is structured in form of graphs. It consists of triplet, namely subject, predicate and object. It allows data to be defined in a way that is understood by machines. Benefit of using RDF is that, it is able to realize XML for easy data exchange.

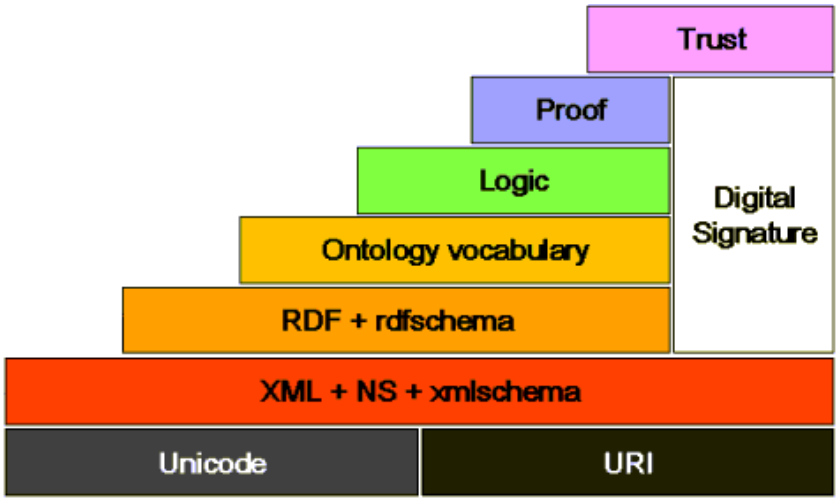

Figure 1.2.1: Semantic Web framework

1.3 Ontology

Ontology is formal, explicit specification of a shared conceptualization. It defines domains and relationships between domains. It helps in building logic and reasoning. It works on top of RDF and explains meaning of RDF based data. Ontology is used to make computers understand the terms and reason on their own based on RDF triplets. Web Ontology Language (OWL) is semantic web language which is used to depict the relationship between data. These documents can be published in the World Wide Web [16].

1.4 Objective

The objective of this project is to monitor air quality levels and then notify the user accordingly. If the air pollution within a given area is found out to be high, the user is notified with this information and thus avoids the particular route.

1.5 Scope of work

To mitigate the health problem caused due to bad to air quality, the solution proposed can be used to interpret and find out areas by creating pollution maps. These pollution maps can be created using predictive analysis based on the data obtained by the sensors. User can then avoid the polluted areas and can use the alternate route.

Chapter 2 – Literature survey

The work discussed in papers related to the project are low cost, low power, reliable and case studies show that they work well in real time environment. Along with this, there has been evolution of communication protocols which are low power, long range and high performance. The project work also involved diving into semantic web world and understanding how data in the web can be understood by machines.

In [1], NodeMCU (Microcontroller Unit) which is ESP8266 was used as node to which the sensors would pass the data. Temperature and humidity sensor (DHT sensor) and GPS sensor were used to obtain location of the module. Gas sensors to detect gases like NO2 and CO2 were also used. These sensors then continuously send the data to MCU which then uploads the data to web after every time interval. After the data is obtained at server, Google sheets were used as database and then Air Quality Index (AQI), which was based on US standards, readings were calculated. These readings were then presented to user in form of a smart phone application through which the user can easily interpret the information regarding air quality. In [11], the authors provide a real-time air quality monitoring system by using the approach of implementing IoT along with cloud computing. A PM sensor was used to monitor particulate matter along with other gas sensors and temperature humidity sensors. An Arduino Uno board was used to which the sensors were connected and would send data to Raspberry pi 2B. A Wi-Fi adapter was used for this. MQTT (Message Queueing Telemetry Transport) publish-subscribe model was incorporated as communication protocol between sensors and clients. The sensors being the publisher and user being the subscribers that were subscribed to the data from the cloud. IBM Bluemix cloud services were used to display data which would be then viewed by the users.

The work proposed in [3] used Arduino module to send the sensors data collectively to a smartphone via Bluetooth. Then the data was sent to the cloud by the smartphone. The data then could be accessed by other users. Also, multiple linear regression model was used for prediction in air quality levels over a period of time. In [9] also, the particulate sensors, gas sensors like MQ-2, temperature and humidity sensors were used which would send data to raspberry pi that acted

as a gateway between these sensors and cloud platform called as ThingSpeak which is real time cloud platform which could monitor each of the sensor readings. An ADC (Analog to Digital Converter) was required to convert MQ-2 signals to digital form. After the data was uploaded on ThingSpeak platform, the data would be then available to the user on the android application which included Google API called as Firebase that handled the security, back end, authentication, storage etc.

In [13], LPWA (Low power wide area) network is used to cover wide span of area in an urban environment. The data sensed by the sensors is sent over to the IoT cloud. At sensor level the four main components involved in this framework are controller module which is basically an MCU(STM32F103RC) that consists of all control functions and operates at high frequency of 72MHz with flash and SRAM memories, sensor module consisting of sensors to detect PM, gases in atmosphere, LPWA module and a battery module. These are then connected to an access point node through which data is then sent to cloud where it is stored and analyzed. Users can then use website or mobile application to find about air quality levels.

In [6], the authors proposed to use NB-IoT as a communication medium because of its benefits such as large capacity, coverage, low cost and low power consumption. The air quality system in consists of modules like computer module which is a 32-bit microcontroller (STM32F103RCT6), power module, sensor module (PTQS1005) and wireless communication module which is NB-IoT. When the NB-IoT network is not responsive, the system switches to GPRS (General Packet Radio Service) module. If both networks fail, then the data is stored in SPI flash. When any one of the systems gets back up then the data will be restored back to the cloud.

In [7], monitoring system comprises of GPS (Global positioning system) that uses the location of the sensors so that user finds out in their application interface the precise location where pollution level is more. Google maps navigation API was used to determine latitude and longitude co-ordinates and map out the region in maps where there was pollution level. Further the sensors were connected to Arduino board with ESP8266 module for Wi-Fi. The data read would be then sent to Ubidots cloud service to store and process data.

In [12], LoRa transmitter was used to communicate data over LoRaWAN communication protocol. LoRa is a suitable network for IoT environment as small

data can be sent over a wide area with low power consumption so it is best fit for batter powered devices. The sensors send this data to a gateway that sends the data to ThingSpeak platform. The user can then monitor the sensor values.

In [10], the methodology was almost the same as in [12] except for the way the data is transmitted. LoRaWAN network infrastructure with the help of a LoRaWAN module was deployed on a city-wide scale to monitor the Air Quality capabilities of the sensors at different locations in the city. For data connectivity, remote access ways like OpenTunnel and SSH were used to connect IoT devices to cloud server directly and a Linux VPS running on Microsoft Azure cloud platform respectively. In [8], the authors developed a system to monitor PM content in the environment. The objective of the prototype was to monitor the PM 2.5 and PM 10 pollutants concentration in the environment by deploying an IoT model which consisted of LoPy ESP32 micro-controller in Micropython language and PM sensor which would send this data over gateway via Wi Fi. Google suite was used at the cloud for storage, processing, web-application service.

In [5], NodeMCU ESP8266 was taken as node to which sensors were attached. These sensors would publish data to a broker using MQTT protocol. The MQTT broker is Mosquitto broker from eclipse foundation. Nodered is flow based programming tool that wires together hardware devices, APIs and online services. Nodered is subscribed to the data sent by the broker. The data received is then displayed in form of text. If the value of data exceeds a certain limit, LINE notification service will be activated which is basically a warning message to the users.

Chapter 3 – Methodology used

Here, we briefly discuss the steps taken to implement the solution. The procedure is as follows:

1) Collect analog readings from the sensors.
2) Send the readings to the server for processing. The server gets the readings at regular time intervals from the sensor node.
3) The user can observe the readings on the web page.
4) Store the readings in XML file.
5) Annotate semantically the XML file to generate RDF file by executing the java application.
6) Run a SPARQL query against the RDF to return the value and deduce meaning out of it.

Chapter 4 – Implementation

4.1 – Air pollution monitoring setup

The tools required for the setup are listed below:

1) ESP8266 – Low cost microchip with Wi-Fi module and processing power. It has 16 GPIO pins.

2) MQ-135 - MQ-135 is an electrochemical gas sensor which is used for detection of gases like C02, NH4, ethanol. It has analog and digital output. The sensor however requires pre-heating period of at least 24 hrs and has to be calibrated to detect a gas.

Sensors	Detects gases	Paper
MQ-2	Methane, Butane, LPG, smoke	[4],[2],[11]
MQ-4	Methane,CNG Gas	[8]
MQ-7	Carbon Monoxide	[18],[6]
MQ-9	Carbon Monoxide, flammable gasses	[13]
MQ-135	Air Quality (Benzene, Alcohol, smoke)	[13],[18]
MQ-136	Hydrogen Sulfide gas	[18]

Table 4.1.1: Gas sensors that can be used in air pollution monitoring

Micro-controllers	Capabilities	Paper
Arduino uno R3	Flash Memory 32 KB (ATmega328P) of which 0.5 KB used by bootloader SRAM KB (ATmega328P) EEPROM 1 KB (ATmega328P) Clock Speed 16 MHz	[4]
Arduino mega	Flash Memory 256 KB of which 8 KB used by bootloader SRAM 8 KB EEPROM 4 KB Clock Speed 16 MHz	[5]
NodeMCU	Flash Memory: 4 MB SRAM: 64 KB Clock Speed:80 Mhz Wi-Fi: IEEE 802.11 b/g/n	[5],[1],[8],[6]
Raspberry pi	Broadcom BCM2837 Arm7 Quad Core Processor powered Single Board Computer running at 900MHz 1GB RAM ,40pin extended GPIO, 4 x USB 2 ports,4 pole Stereo output and Composite video port,Full size HDMI	[2],[11]
STM32F103RC Microcontroller Unit (MCU)	Cortex-M3 core,max CPU speed of 72 MHz 16 Kbytes to 1 Mbyte of Flash motor control peripherals USB full-speed interface	[20]

Table 4.1.2: Node devices and gateways that can be used with their capabilities

The software tools required for the setup are as follows:

1) Eclipse IDE – Eclipse is an open source IDE which is used for computer programming, creating Java applications, and other web projects.
2) Arduino IDE – An open source software to write and compile code written in C/C++ language in the Arduino module.
3) Jena framework and SPARQL – For semantic processing, apache jena is a framework developed for Java. SPARQL is a semantic query language.
4) Machine-to-Machine Measurement (M3) application template – It is framework which is based on semantic web. It defines the meaning of the sensor values in such a way that, it can be easily understood by machines. It is used to build IoT applications that consist of different domains (Cross-domain support). Here, we generate a template for "Environment" domain with "Air pollutant sensor" as the sensor that provides with data [14].

In this project, we made use of two ESP8266 devices in which one of the nodes acts as a server and access point and another for the MQ-135 gas sensor which acts as our edge node in the network. For deployment scenarios, we should make use of industrial Micro Controller Units (MCUs) instead of ESP8266. The edge node ESP8266 would be connected to another ESP8266 which works as server and displays the readings at regular time interval of 10 seconds. Here we make use of Wi-Fi communication network. Other communication mediums that can be used are SigFox, Long Range Wide Area Network (LoRaWAN), 2G/3G/4G-LTE. The processing is done locally on the user machine. The server then is able to forward the sensor readings for further processing.

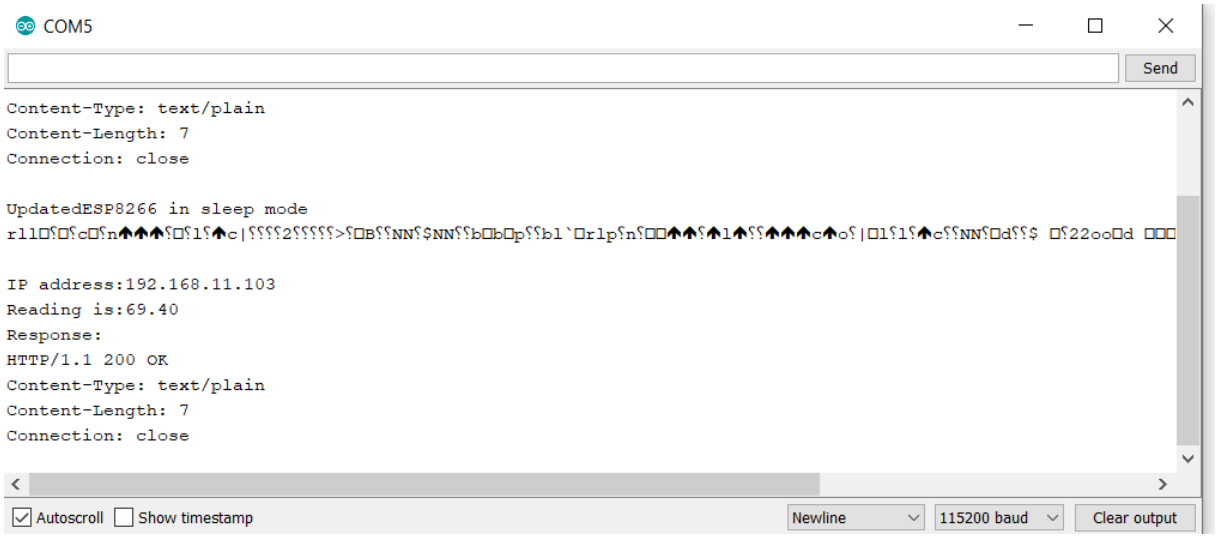

Figure 4.1.1: Reading sent by the edge node

There are certain steps to take care for MQ-135 sensor calibration. As it is not a multi-gas real-time detector, we can calibrate it only for one gas at a time. Resistance value of MQ-135 is difference to various kinds and concentration of gases. The following steps are required towards calibration for CO2:

1) Connect the sensor to the circuit and leave it powered on for 24hrs (preheat time).
2) So, to calibrate the detector it was observed that a load resistance (RL) has to be applied to the board in series (preferably 20K ohm according to datasheet).

3) Then put the sensor in outside air (Temp: 20±2C and Humidity: 65% ±5% according to datasheet).
4) Calculate the calibration value as:
   float rzero = gasSensor.getRZero();
5) The determined value of Rzero is then defined in mq135.h file.
6) Read the ppm value for C02 gas only as:
   float ppm = gasSensor.getPPM();

4.2 – Defining the data using RDF

First, we download the already available rules, ontology and RDF that is part of Semantic Web of Things (SWoT) generator. For designing a SWoT application that operates in cross-domain environment, developers require the necessary dataset to work upon. So, by providing sensor and domain, we can generate template for the corresponding application. The template consists of the particular domain ontologies, datasets, Rules and SPARQL queries. The rules, ontologies and datasets are interoperable among each other. In this case, we searched for "Environment domain" and sensor as "Air pollutant sensor" which gives us the already defined files that are created by the communities and organizations. It is called as Linked Open data. M3 ontology defines in a uniform way the various sensors, measurements, domains and units, to combine cross-domain knowledge and a new concept called Linked Open Rules to share and reuse in a uniform way semantic domain rule [15]. The following are part of building our semantic model:

1) environment.owl - The environment ontology is defined here.
2) environment-dataset.rdf - The environment dataset is defined here.
3) m3SparqlGeneric.sparql - Query executed to get the necessary meaning out of data.
4) m3.owl - The m3 ontology that describes sensor data in interoperable manner to ease reasoning and interlinking of domains.
5) LinkedOpenEnvironmentRules.txt - It consists of rules that defines the final deduction based on sensor measurements.
6) rulem3Converter.txt – It consists of set of rules that converts sensor data according to M3 language.

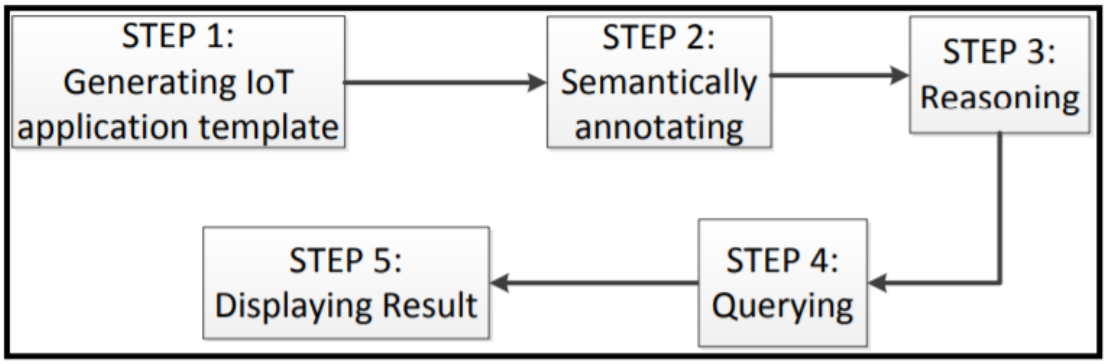

Figure 4.2.1: M3 framework

Here, an IoT based Java application is created by which we can create meaning between data using RDF and OWL. The steps in which the final outcome is achieved is as follows:

1) Generate IoT application template with sensor as "Air pollutant sensor" and domain as "Environment" as shown in below figure from [design cross domain].

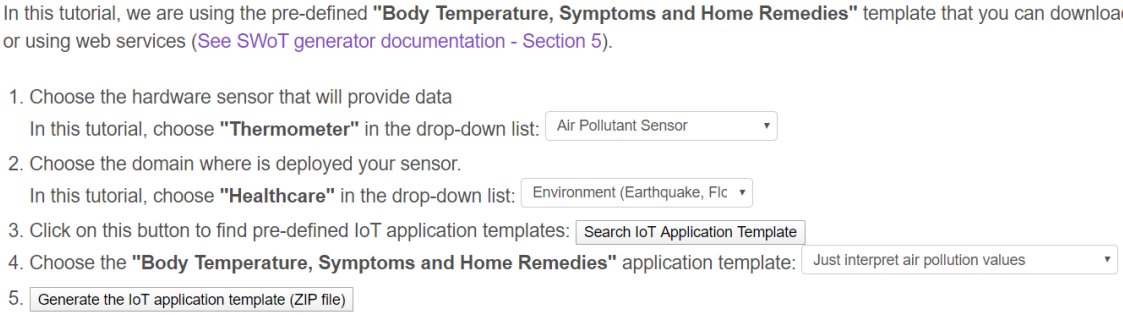

Figure 4.2.2: Generate template

2) Now, semantically annotate the data with RDF by loading rules for semantic annotation and M3 ontology.

```
// STEP: SEMANTIC ANNOTATION

SemanticAnnotator semanticAnnotator = new SemanticAnnotator();
System.out.println(semanticAnnotator.convertXMLSenMLIntoRDF(sensorMeasurements));

// WRITE SEMANTIC SENSOR DATA IN A FILE
String fileName = GENERATED_SEMANTIC_SENSOR_DATA;
FileWriter out = new FileWriter(fileName);
semanticAnnotator.model.write(out,"RDF/XML-ABBREV");
```

Figure 4.2.3: Semantic annotation of data stored in XML

Figure 4.2.4: Generated sensor data in RDF

3) Then, using the M3 framework we build RDF data that is compliant with M3.

4) Jena inference engine is basically a reasoning engine in the code that is used to derive the final result based on the sensor data, and rules defined for logic. It derives the meaning out of the data. It updates the RDF sensor data obtained in step 2.

```
// STEP: LOAD SEMANTIC SENSOR DATA
Model model = ModelFactory.createDefaultModel();
//GENERATED_SEMANTIC_SENSOR_DATA = input.toUri().toString();
ReadFile.enrichJenaModelOntologyDataset(model, GENERATED_SEMANTIC_SENSOR_DATA);


// GENERIC APPLICATION
GenericApplication generic_appli = new GenericApplication(model);
// M3_ONTOLOGYinput.toUri().toString()
// STEP: SPECIFIC DOMAIN ONTOLOGIES AND DATASETS
ReadFile.enrichJenaModelOntologyDataset(generic_appli.model, M3_ONTOLOGY);
ReadFile.enrichJenaModelOntologyDataset(generic_appli.model, ENVIRONMENT_ONTOLOGY);
ReadFile.enrichJenaModelOntologyDataset(generic_appli.model, ENVIRONMENT_DATASET);


// STEP: EXECUTING REASONING ENGINE
Model deduceMeaningfulInformationFromSensorData =
        generic_appli.executeReasoningEngine(LINKED_OPEN_RULES_ENVIRONMENT);
```

Figure 4.2.5: Executing Jena reasoning engine

5) Now, SPARQL engine is executed.

```
D:\Program Files\apache-jena-3.14.0\bat>sparql.bat --data=q2.rdf --query=q1.rq
11:19:11 WARN  riot                :: Lexical form '1231' not valid for datatype XSD dateTime
11:19:11 WARN  riot                :: Lexical form '1121' not valid for datatype XSD dateTime
11:19:11 WARN  riot                :: Lexical form '1523' not valid for datatype XSD dateTime
-----------------------------------
| name              | value | unit |
===================================
| "AirQuality zone1" | 80.0  | "PM" |
| "AirQuality zone2" | 100.0 | "PM" |
| "AirQuality zone3" | 50.0  | "PM" |
-----------------------------------

D:\Program Files\apache-jena-3.14.0\bat>sparql.bat --data=q2.rdf --query=q2.rq
11:19:16 WARN  riot                :: Lexical form '1231' not valid for datatype XSD dateTime
11:19:16 WARN  riot                :: Lexical form '1121' not valid for datatype XSD dateTime
11:19:16 WARN  riot                :: Lexical form '1523' not valid for datatype XSD dateTime
-----------------------------------
| name              | value | unit |
===================================
| "AirQuality zone1" | 80.0  | "PM" |
| "AirQuality zone2" | 100.0 | "PM" |
-----------------------------------
```

Figure 4.2.6: Query output returned when PM value is greater than 70

Chapter 5 – Conclusion

In order to deploy the system on a large scale, there are large number of low power communication protocols that can be used such as Long Range Wide Area Network (LoRaWAN), Sigfox, Narrowband-IoT. A real time monitoring system can be used to send the data by different sensors at intervals, this data gets processed and the meaning of the data is derived too. The processing that would occur at cloud. Also, we need to consider the security aspects and of the data. So, in a smart city, different sensors located at different locations would send the data to the cloud at different periods. The system needs to ensure that the user gets the accurate results from the application. This way the user can avoid the pollution zones and go along less polluted route and avoid the health problems caused due to harmful gases present in atmosphere.

References

[1] H. Aamer et al. "A Very Low Cost, Open, Wireless, Internet of Things (IoT) Air Quality Monitoring Platform". In: 2018 15th International Conference on Smart Cities: Improving Quality of Life Using ICT IoT (HONET-ICT). 2018, pp. 102–106.

[2] S. Abraham, J. Beard, and R. Manijacob. "Remote environmental monitoring using Internet of Things (IoT)". In: 2017 IEEE Global Humanitarian Technology Conference (GHTC). 2017, pp. 1–6.

[3] A. Barthwal and D. Acharya. "An Internet of Things System for Sensing, Analysis Forecasting Urban Air Quality". In: 2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT). 2018, pp. 1–6.

[4] A. Candia et al. "Solutions for SmartCities: proposal of a monitoring system of air quality based on a LoRaWAN network with low-cost sensors". In: 2018 Congreso Argentino de Ciencias de la Inform´atica y Desarrollos de Investigaci´on (CACIDI). 2018, pp. 1–6.

[5] S. Chanthakit and C. Rattanapoka. "MQTT Based Air Quality Monitoring System using Node MCU and Node-RED". In: 2018 Seventh ICT International Student Project Conference (ICT-ISPC). 2018, pp. 1–5.

[6] Y. Cheng et al. "Design of Air Quality Monitoring System Based on NB-IoT". In: 2019 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS). 2019, pp. 385–388.

[7] S. Dhingra et al. "Internet of Things Mobile–Air Pollution Monitoring System (IoT-Mobair)". In: IEEE Internet of Things Journal 6.3 (2019), pp. 5577–5584.

[8] O. O. Flores-Cortez, R. Adalberto Cortez, and V. I. Rosa. "A Lowcost IoT System for Environmental Pollution Monitoring in Developing Countries". In: 2019 MIXDES - 26th International Conference "Mixed Design of Integrated Circuits and Systems". 2019, pp. 386–389.

[9] Harsh Gupta et al. "An IoT Based Air Pollution Monitoring System for Smart Cities". In: Feb. 2019, pp. 173–177. doi: 10. 1109/ ICSETS. 2019.8744949.

[10] S. J. Johnston et al. "IoT deployment for city scale air quality monitoring with Low-Power Wide Area Networks". In: 2018 Global Internet of Things Summit (GIoTS). 2018, pp. 1–6.

[11] S. Kumar and A. Jasuja. “Air quality monitoring system based on IoT using Raspberry Pi”. In: 2017 International Conference on Computing, Communication and Automation (ICCCA). 2017, pp. 1341–1346.
[12] K. M. Simitha and M. S. Subodh Raj. “IoT and WSN Based Air Quality Monitoring and Energy Saving System in SmartCity Project”. In: 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT). Vol. 1. 2019, pp. 1431–1437.
[13] K. Zheng et al. “Design and Implementation of LPWA-Based Air Quality Monitoring System”. In: IEEE Access 4 (2016), pp. 3238–3245.
[14]Gyrard, Amelie. (2015). Designing cross-domain semantic Web of things applications.
[15] A. Gyrard, C. Bonnet and K. Boudaoud, "Enrich machine-to-machine data with semantic web technologies for cross-domain applications," 2014 IEEE World Forum on Internet of Things (WF-IoT), Seoul, 2014, pp. 559-564, doi: 10.1109/WF-IoT.2014.6803229.
[16] A. Gyrard, M. Serrano and G. A. Atemezing, "Semantic web methodologies, best practices and ontology engineering applied to Internet of Things," 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, 2015, pp. 412-417, doi: 10.1109/WF-IoT.2015.7389090.

Appendix – List of useful websites

1. https://www.st.com/en/microcontrollers-microprocessors/stm32f103.html
2. https://www.element14.com/community/docs/DOC-73827/l/raspberry-pi-2-model-b-1gb-technical-specifications
3. https://nodered.org/
4. https://docs.zerynth.com/latest/official/board.zerynth.nodemcu3/docs/index.html
5. https://pib.gov.in/newsite/PrintRelease.aspx?relid=110654
6. store.arduino.cc/usa/
7. https://www.mysensors.org/build/gas
8. https://www.w3.org/RDF/Metalog/docs/sw-easy
9. https://www.w3.org/2001/sw/wiki/OWL
10. https://www.w3.org/TR/rdf-sparql-query/
11. http://sensormeasurement.appspot.com/?p=end_to_end_scenario