Pollution Prediction Using IoT Systems

EGBOH DANIEL CHUKWUNONSO
University of Bradford

Abstract- ***Aquaculture has emerged as a critical sector in addressing global food security and the increasing demand for seafood. Effective management of aquaculture ponds is essential to ensure optimal growth and health of aquatic organisms. Temperature monitoring plays a vital role in understanding the pond's thermal dynamics, which directly impact the well-being of aquatic species. Wireless sensor networks (WSNs) have attracted a lot of attention recently. as a viable solution for real-time data collection in various domains, including aquaculture. This paper presents a study on data fusion techniques based on temperature monitoring of aquaculture ponds using WSNs. The primary objective is to develop a robust and accurate approach for acquiring and analysing temperature data from multiple sensors deployed in the pond environment. The proposed data fusion methodology combines data from different sensors to obtain a comprehensive and reliable representation of the pond's temperature profile. In this coursework, we will be carrying out three parts of the processing to include: a paper review, summarization and preparation of Data analysis, secondly, we shall carry out a time series analysis and prediction of the dataset, furthermore, we will we'll simulate real-time data from two distinct stations using the MQTT protocol, and we'll utilise Apache Flink to interpret real-time streams and complicated events. (CEP). The research focuses on the challenges associated with data collection, transmission, and fusion in an aquatic environment. The study investigates various WSN architectures, sensor placement strategies, and communication protocols suitable for aquaculture pond monitoring. Furthermore, it explores data fusion algorithms and techniques to integrate temperature readings from multiple sensors, considering factors such as sensor accuracy, spatial distribution, and temporal correlation.***

I. INTRODUCTION

Pollution is a growing concern worldwide, affecting both the environment and human health. Traditional methods of pollution monitoring and prediction have limitations in terms of coverage, timeliness, and accuracy (Himadri et al, 2017). However, with the advent of Internet of Things (IoT) systems, there is a significant opportunity to revolutionize pollution prediction and management. This technical introduction provides an overview of pollution prediction using IoT systems and highlights its potential benefits and challenges. IoT systems leverage interconnected devices and sensors to collect, transmit, and analyse vast amounts of data from various sources. When applied to pollution prediction, IoT systems offer a comprehensive and real-time approach to monitoring pollution levels, identifying pollution sources, and predicting future trends. By deploying a network of sensors in key locations, such as industrial areas, urban centres, or near emission sources, IoT systems enable continuous data collection on pollutants such as volatile organic compounds (VOCs), particulate matter (PM), nitrogen oxides (NOx), sulphur dioxide (SO2), and so on (Akshay et al, 2018).

The collected data from IoT-enabled sensors are transmitted to a centralized platform, where sophisticated algorithms and machine-learning techniques are applied to analyse and predict pollution patterns (Ayaskanta et al, 2018). These algorithms consider various factors such as meteorological data, traffic patterns, and historical pollution data to generate accurate predictions and insights. The predictions can be visualized on dashboards, allowing policymakers, environmental agencies, and the public to access real-time pollution information and make informed decisions. One of the significant advantages of pollution prediction using IoT systems is its ability to provide timely alerts and early warnings. By continuously monitoring pollution levels and

analysing data in real time, IoT systems can detect sudden spikes or exceedances in pollution concentrations, allowing immediate responses and preventive measures to be implemented. For example, alerts can be sent to individuals or organizations to take action, such as adjusting activities, implementing traffic management strategies, or activating air quality control systems (Gangwar et al, 2023). Moreover, IoT systems facilitate the integration of pollution prediction data with other environmental and health datasets. By combining pollution data with data on weather conditions, traffic flows, and health records, correlations and insights can be derived, leading to a deeper understanding of the relationships between pollution and various factors. This integrated approach enables policymakers and researchers to develop more effective pollution control strategies, implement targeted interventions, and assess the impacts of pollution on public health (Okokpujie et al, 2018).

However, implementing pollution prediction using IoT systems also poses several challenges. Ensuring data accuracy and reliability is critical, as sensor calibration, data transmission, and environmental factors can introduce errors. Data security and privacy are additional concerns, as the collection and analysis of sensitive data require robust security measures to protect against unauthorized access and misuse (Barthwal et al, 2018). Furthermore, the scalability and cost-effectiveness of IoT systems for widespread deployment should be considered. The large-scale deployment of sensors and the maintenance of IoT infrastructure require significant investments. Therefore, careful planning and optimization strategies are necessary to achieve cost-effective and scalable solutions. In conclusion, pollution prediction using IoT systems offers a transformative approach to addressing the challenges of pollution monitoring and management. By leveraging interconnected devices, continuous data collection, and advanced analytics, IoT systems enable real-time pollution monitoring, accurate prediction, and informed decision-making. While there are challenges to overcome, the potential benefits of IoT-based pollution prediction are substantial, including improved environmental management, enhanced public health outcomes, and the development of targeted pollution control strategies.

PART 2, TASK 1 (ADAPTATION OF GROUP 5 DATASET)

DATA DESCRIPTION AND ANALYSIS

The data in use is an open data source on pollution dataset from the EU project CityPulse from the following link http://iot.ee.surrey.ac.uk:8080/datasets/pollution/ repository which contains 17568 instances and 8 columns, this is an adaptation of the group 5 dataset.

DATA PREPROCESSING

The data was processed and cleaned to ensure that there are no null values or missing values in order to avoid any form of outliers and below are the summary statistics of the dataset showing the variables in the dataset and their distribution.

```
> #check for null values
> is.null(data)
[1] FALSE
>
> #check the summary statistics of the dataset
> summary(data)
     ozone       particullate_matter carbon_monoxide sulfure_dioxide
 Min.   : 15.0   Min.   : 15.0       Min.   : 15.0   Min.   : 15.0
 1st Qu.: 65.0   1st Qu.: 77.0       1st Qu.:104.0   1st Qu.: 58.0
 Median :106.0   Median :119.0       Median :147.0   Median : 96.0
 Mean   :114.9   Mean   :116.6       Mean   :139.9   Mean   : 99.8
 3rd Qu.:170.0   3rd Qu.:159.0       3rd Qu.:182.0   3rd Qu.:138.0
 Max.   :214.0   Max.   :214.0       Max.   :215.0   Max.   :215.0
 nitrogen_dioxide   longitude        latitude      timestamp
 Min.   : 15.0    Min.   :10.18   Min.   :56.14   Length:17568
 1st Qu.: 63.0    1st Qu.:10.18   1st Qu.:56.14   Class :character
 Median :110.0    Median :10.18   Median :56.14   Mode  :character
 Mean   :112.9    Mean   :10.18   Mean   :56.14
 3rd Qu.:165.0    3rd Qu.:10.18   3rd Qu.:56.14
 Max.   :215.0    Max.   :10.18   Max.   :56.14
```

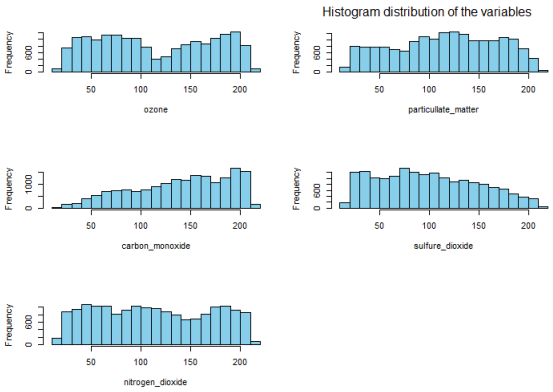
Also, the data type of the different variables was checked the necessary corrections were made, it was discovered that the Timestamp format is (chr) and was converted to the correct date time format (POSIXct)

```
> # check for data types of the variables
> str(data)
'data.frame':   17568 obs. of  8 variables:
 $ ozone              : int  40 36 40 42 44 41 42 42 42 47 ...
 $ particullate_matter: int  95 93 96 97 99 97 95 95 96 97 ...
 $ carbon_monoxide    : int  55 51 54 54 59 54 52 55 52 56 ...
 $ sulfure_dioxide    : int  80 82 85 83 82 77 73 70 69 74 ...
 $ nitrogen_dioxide   : int  62 58 62 64 61 59 60 55 59 60 ...
 $ longitude          : num  10.2 10.2 10.2 10.2 10.2 ...
 $ latitude           : num  56.1 56.1 56.1 56.1 56.1 ...
 $ timestamp          : chr  "01/08/2014 00:05" "01/08/2014 00:10" "01/08/2014 00:15" "01/08/2014 00:20" ...
>
```

Furthermore, Longitude and Latitude columns were not needed since they had the same values and were dropped hence a new data type summary is provided below:

```
'data.frame':   17568 obs. of  6 variables:
 $ ozone              : int  40 36 40 42 44 41 42 42 42 47 ...
 $ particullate_matter: int  95 93 96 97 99 97 95 95 96 97 ...
 $ carbon_monoxide    : int  55 51 54 54 59 54 52 55 52 56 ...
 $ sulfure_dioxide    : int  80 82 85 83 82 77 73 70 69 74 ...
 $ nitrogen_dioxide   : int  62 58 62 64 61 59 60 55 59 60 ...
 $ timestamp          : POSIXct, format: "2014-08-01 00:05:00" "2014-08-01 00:10:00" ...
```

Also, the univariate analysis of each variable was analysed using a histogram plot which further shows that the data is evenly distributed in virtually all the variables except in carbon monoxide which is left-skewed and sulphur dioxide which is right skewed as shown below.



Furthermore, a box plot analysis was carried out to check for outliers, however, it was discovered that the dataset contains no outliers as shown in the figure below having no outliers.

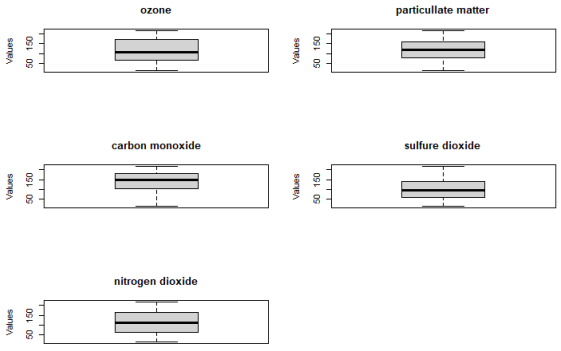


Figure 4: Box plot of the Dataset

Also, a line plot was further used to analyse the data showing their changes over time. It can be observed that the observations were in their peak state at some point as well as at their lowest point at some point in the month as shown below.

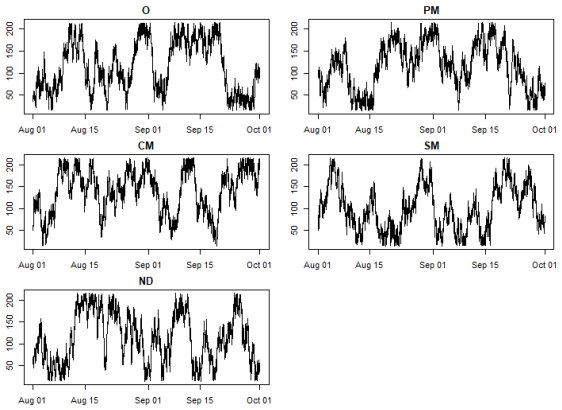


Figure 5: Line plot of the Dataset.

We also carried out a scattered plot analysis of the dataset variables, however, the analysis shows that there is no correlation between the variable of the dataset as shown below.

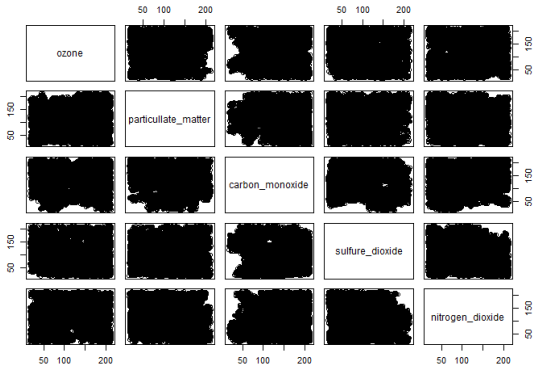


Figure 6: Scattered plot of the correlation of the variables

The Ozone Variable was chosen for further analysis to show the seasonal plots, however, there was no change observed between the hourly, daily and weekly plots as shown below.

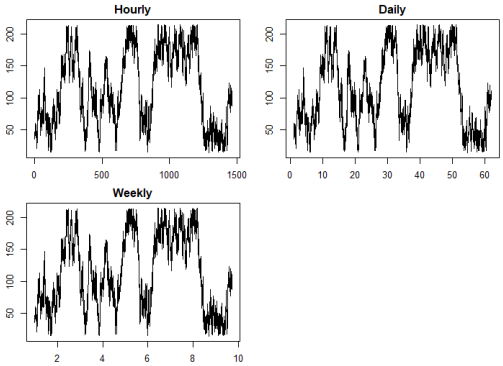


Figure 7: Seasonal Plot of the variables

Finally, a time series for each month was separately plotted for more insight into the dataset using the Ozone variable and the changes over time are shown below.

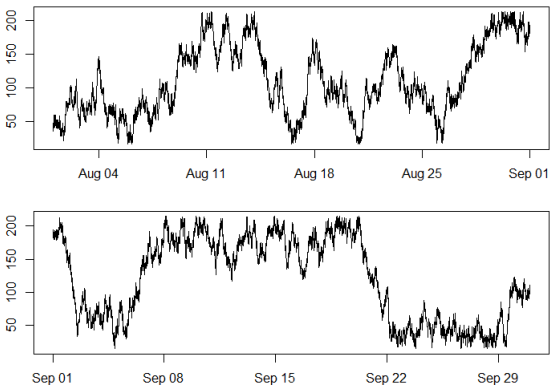


Figure 8: Time series Plot for the various month

PART 2, TASK 2 Time Series Analysis and Prediction
Using the dataset allocated to group 5, The figure below shows a snippet of the data frame using the suggested data points for time stamp and carbon monoxide which goes to show that the data frame consists of 17568 instances and 2 columns.

A data.frame: 17568 × 2

timestamp	carbon_monoxide
<chr>	<int>
2014-08-01 00:05:00	89
2014-08-01 00:10:00	93
2014-08-01 00:15:00	91
2014-08-01 00:20:00	88
2014-08-01 00:25:00	88
2014-08-01 00:30:00	90
2014-08-01 00:35:00	86
2014-08-01 00:40:00	86
2014-08-01 00:45:00	86
2014-08-01 00:50:00	90
2014-08-01 00:55:00	91
2014-08-01 01:00:00	86

Figure 2.1: Data frame of the Dataset

DATASET PREPROCESSING
The dataset has been cleaned in part 1 of the work hence there are no missing or null values, this was achieved by ensuring that rows with missing values were removed. Therefore, the data was split into 70% for training and 30% for testing data. the dataset is split into a train set and a test set, we can effectively train, validate, and evaluate machine learning models, ensuring they perform well on unseen data and avoid overfitting. Below is a snippet for the split of the data and the data size.

```
dataSize <- floor(0.7* nrow(df))
dataSize
train <- df[1:size, ]
test <- df[(size + 1):nrow(df), ]
```

```
12297
```

ARIMA-BASED TIME SERIES MODEL

An Arima-based time series model with a train data frame was created using the dataset and a snippet of the various points and forecast results are shown below as the complete work is attached in the code.

```
 ARIMA(2,1,2) with drift         : Inf
 ARIMA(0,1,0) with drift         : 81354.48
 ARIMA(1,1,0) with drift         : 81351.31
 ARIMA(0,1,1) with drift         : 81351.24
 ARIMA(0,1,0)                    : 81352.48
 ARIMA(1,1,1) with drift         : 81353.28
 ARIMA(0,1,2) with drift         : 81352.55
 ARIMA(1,1,2) with drift         : Inf
 ARIMA(0,1,1)                    : 81349.24
 ARIMA(1,1,1)                    : 81351.27
 ARIMA(0,1,2)                    : 81350.54
 ARIMA(1,1,0)                    : 81349.31
 ARIMA(1,1,2)                    : Inf

 Best model: ARIMA(0,1,1)

Series: train
ARIMA(0,1,1)

Coefficients:
          ma1
      -0.0195
s.e.   0.0085

sigma^2 = 19.12:  log likelihood = -40672.62
AIC=81349.24   AICc=81349.24   BIC=81364.34

Training set error measures:
                      ME     RMSE      MAE         MPE     MAPE     MASE
Training set -0.0004997832 4.372156 2.853864 -0.09634068 2.568526 1.001857
                    ACF1
Training set 0.0001328281
```

Figure 2.1 Arima-Based Time Series Model

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
12298	39.97597	35.8362262	44.11571	33.6447802	46.30715
12299	39.97654	34.1919925	45.76109	31.1298361	48.82325
12300	39.97653	32.9193387	47.03372	29.1834873	50.76957
12301	39.97653	31.8434867	48.10957	27.5381135	52.41495
12302	39.97653	30.8941925	49.05887	26.0862934	53.86677
12303	39.97653	30.0351360	49.91792	24.7724798	55.18058
12304	39.97653	29.2446255	50.70843	23.5634983	56.38956
12305	39.97653	28.5084772	51.44458	22.4376567	57.51540
12306	39.97653	27.8168139	52.13625	21.3798490	58.57321
12307	39.97653	27.1624301	52.79063	20.3790553	59.57401
12308	39.97653	26.5398780	53.41318	19.4269441	60.52612
12309	39.97653	25.9449201	54.00814	18.5170344	61.43603
12310	39.97653	25.3741830	54.57888	17.6441674	62.30889
12311	39.97653	24.8249295	55.12813	16.8041567	63.14890

Figure 2.2: Model of time series based on Arima Data Frame.

MEAN ABSOLUTE PERCENTAGE ERROR

Mean Absolute Percentage Error (MAPE) is a statistical measure used to quantify the accuracy of a forecasting model or prediction by evaluating the typical proportion of the anticipated values that deviate from the matching actual values. It frequently appears in the field of time series analysis and forecasting.

The formula to calculate MAPE is as follows:

MAPE = (1 / n) * Σ(|(Actual - Predicted) / Actual|) * 100

Where:

- MAPE is the Mean Absolute Percentage Error.
- n is the total number of observations or data points.
- Σ represents the summation operator.
- Actual refers to the actual observed values.
- Predicted refers to the predicted values.

MAPE expresses the average relative deviation between the numbers as projected and as they were as a percentage. It provides a standardized way to assess the forecasting accuracy, regardless of the magnitude of the data. A lower MAPE indicates higher accuracy, while a higher MAPE suggests greater forecasting error. The mape from our forecast as shown below of 0.680259102603479 indicates a higher accuracy in our analysis conducted.

```
Training set error measures:
                      ME     RMSE      MAE       MPE     MAPE     MASE
Training set -0.0004997832 4.372156 2.853864 -0.09634068 2.568526 1.001857
                    ACF1
Training set 0.0001328281
[1] "MAPE:  0.680259102603479"
```

Figure 2.3 Snippet of output showing the MAPE of the forecast.

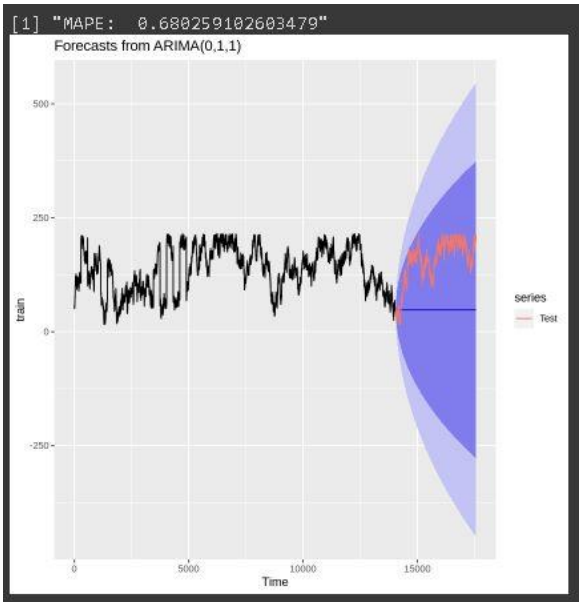


figure 2.3 Arima forecast plot showing mape value

ARIMA MODEL PLOT OF THE FORECAST

An Arima model plot was further carried out to show the result of the analysis graphically.

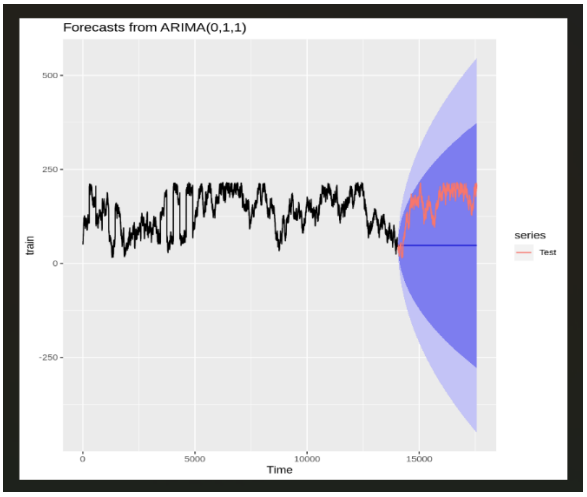


Figure 2.4: Arima model Forecast Plot

If an ARIMA model yields a Mean Absolute Percentage Error (MAPE) value of 0.680259102603479, it implies that the model's predictions are typically about 5% off the actual results of 0.68026 or 68.026%.

MAPE is a measure of forecasting accuracy, where lower values indicate higher accuracy. In this case, a MAPE value of 0.68026 suggests that the ARIMA model's forecasts have a comparatively high average percentage difference based on the real values.

When interpreting the MAPE, it's important to consider the specific context and domain of the problem. Depending on the industry or application, a MAPE of 0.68026 may or may not be considered acceptable. For instance, in some fields where precise predictions are critical, such as finance or demand forecasting, a MAPE of 0.68026 might be considered relatively high. On the other hand, in domains where predictions have inherent uncertainty or high variability, a MAPE of 0.68026 could be acceptable or even satisfactory.

It is also worth noting that MAPE is just one of many metrics used to evaluate forecasting models. It is always recommended to consider multiple evaluation metrics, assess the model's performance across different time periods or datasets, and compare it with alternative models or benchmarks to comprehensively understand its accuracy.

SUMMARY/CONCLUSION

Time series analysis is a statistical method used to analyse and predict data that is collected over time. It is commonly applied in various fields such as Health, finance, economics, weather forecasting, and sales forecasting. One popular approach for time series analysis is the ARIMA (AutoRegressive Integrated Moving Average) model. The ARIMA model combines three key components: autoregression (AR), differencing (I), and moving average (MA). These components capture different aspects of the time series data, allowing for the analysis and prediction of trends, seasonality, and other patterns. hence in this section, we have been able to utilise the given dataset allocated to my group, divided them into training and testing set, and created an Arima-based time series model with a train data frame. Furthermore, this model was applied to test the data frame for the prediction of pollution. Also, the mean absolute percentage error (MAPE) of the model was analysed and discovered and given as 0.680259102603479 which can be said to indicate a higher accuracy of the predicted values. Finally, the plot of the Arima forecast model was drawn to give a vivid and quick understanding of the analysis.

TASK 3 – SIMULATING REAL-TIME STREAM FROM THE TWO STATIONS BY APPLYING MQTT PROTOCOL

In other to achieve the task stated above, first, a couple of software and extensions were downloaded to aid the processing involved in the simulation of real-time streams from the stations. Hence the Java development kit 20.0.1 was downloaded and added to part as shown below.

Figure 3.1 Screenshot of Java software downloaded.

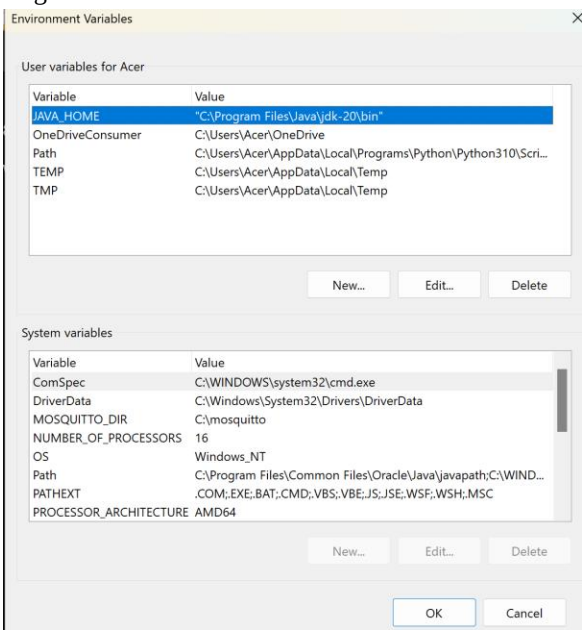


Figure 3.2 Screenshot of the Java software added to part.

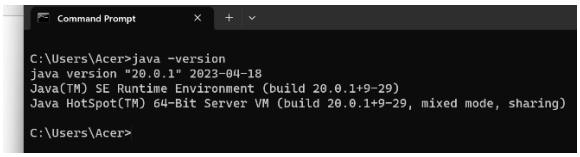


Figure 3.3 Screenshot of the presence of Java software on the system.

The MQTT broker provided by Eclipse was downloaded and the environment was set up for the programming as shown below.

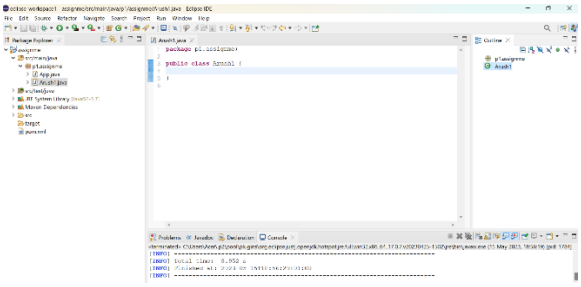


Figure 3.4: Screenshot of Eclipse Environment set-up. After the set-up was completed, an MQTT publisher program was written for both Arhus1 and 2 with Arhus1 publishing two data points from the group5 dataset while Arhus2 publishes two data points of timestamp and carbon monoxide from the pollution data as its payload every 10 seconds.

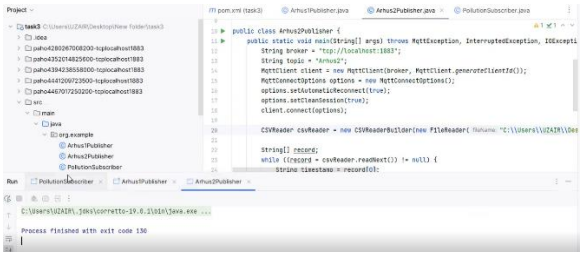


Figure 3.5 Snapshot of Arhus2publisher displayed

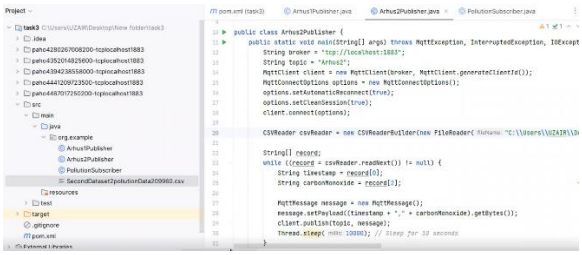


Figure 3.6 Screenshot of Arhus1publisher displayed.

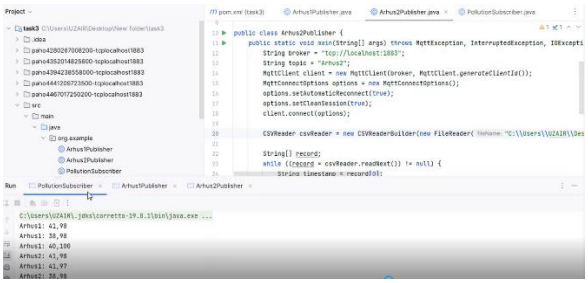


Figure 3.7 Screenshot of the publisher and subscriber working perfectly.

SUMMARY/CONCLUSION

The MQTT (Message Queuing Telemetry Transport) protocol is a lightweight, publish-subscribe messaging protocol commonly used in IoT (Internet of Things) applications. It is designed to enable efficient and reliable communication between devices and applications in real-time or near-real-time scenarios. Simulating real-time streams from two stations using MQTT can provide a means to transmit data between these stations in a decentralized and scalable manner. MQTT Broker, A central MQTT broker acts as a messaging intermediary between the two stations. It receives messages published by the stations and routes them to the appropriate subscribers. The broker manages topics to which devices can subscribe and from which they can publish. In this task, two MQTT publisher program has been written which published two data points (Timestamp and Carbon monoxide) from the provided dataset as its payload every 10 seconds. An MQTT broker provided by Eclipse was downloaded and used in the course of the program. Also, an MQTT subscriber programme was further developed which was used to subscribe to the two topics from the publishers' program and finally, the output of the publisher and the subscriber was synchronised to work correctly as shown in the snippets above. Overall, utilizing the MQTT protocol enables the simulation of real-time streams from two stations in a scalable, efficient, and decentralized manner. It facilitates the seamless transmission of data between the stations and provides a reliable messaging framework for real-time communication in IoT and other applications.

TASK 4: USE OF APACHE FLINK FOR REAL-TIME STREAM PROCESSING AND COMPLEX EVENT PROCESSING (CEP)

Apache Flink is an open-source, distributed stream processing and batch processing framework designed to process large-scale data sets and real-time data streams. It provides a unified programming model and execution engine to enable high-throughput, low-latency, fault-tolerant data processing. Flink supports stream processing, allowing for continuous and real-time data processing. It provides the ability to process data as it arrives, enabling the handling of high-velocity data streams with low latency. Flink's stream processing capabilities include event-time processing, windowing, and stateful computations.

SUMMARY/CONCLUSION

Overall, Apache Flink is a powerful, distributed data processing framework that excels in handling real-time data streams and large-scale batch processing. Its unified programming model, fault tolerance, scalability, and ecosystem integrations make it suitable for a wide range of use cases, including stream analytics, machine learning, fraud detection, and data pipelines. In the course of the task, real-time streams from the two stations in the task completed earlier were ingested as input, afterwards, the complex event processing pattern was defined for low pollution using Apache flink pattern class.

REFERENCES

[1] Himadri Nath Saha, Supratim Auddy, Avimita Chatterjee, Subrata Pal, Shivesh Pandey, Rocky Singh, Rakhee Singh, Priyanshu Sharan, Swarnadeep Banerjee, Debmalya Ghosh, Ankita Maity. (2017) “Pollution Control Using Internet of Things (IoT)”. 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON)

[2] Akshay Kajale, Prathamesh Inamdar, Vivek Bagad, Ankit Mhaske, Ameya Joshi, Prof. Pooja Dhule. (2018) “Cloud and IoT Enabled Smart Air Pollution Monitoring System”, International Journal of Innovative Research in Science, Engineering and Technology, Vol. 7, Issue 4.

[3] Ayaskanta Mishra. (2018) “Air Pollution Monitoring System based on IoT: Forecasting and Predictive Modelling using Machine Learning”. International Conference on Applied Electromagnetics, Signal Processing and Communication (AESPC).

[4] Gangwar, A., Singh, S., Mishra, R. et al. (2023). The State-of-the-Art in Air Pollution Monitoring and Forecasting Systems Using IoT, Big Data, and Machine Learning. *Wireless Pers Commun* 130, 1699–1729 https://doi-org.brad.idm.oclc.org/10.1007/s11277-023-10351-1

[5] Okokpujie, K., Noma-Osaghae, E., Modupe, O., John, S., & Oluwatosin, O. (2018). A smart air pollution monitoring system. Int J Civ Eng Technol, 9(9), 799–809.

[6] Barthwal, A., & Acharya, D. (2018). An internet of things system for sensing, analysis & forecasting urban air quality. In The IEEE International Conference on Electronics, Computing and Communication Technologies (IEEE CONECCT). India: Bangalore.

[7] Elliott, G., Rothenberg, T. J., & Stock, J. H. (1996). Efficient tests for an autoregressive unit root. Econometrica, 64(4), 813–836. https://doi-org.brad.idm.oclc.org/10.2307/2171846.

[8] Kiruthika, R., & Umamakeswari, A. (2017). Low-cost pollution control and air quality monitoring system using Raspberry Pi for the Internet of Things. In 2017 international conference on Energy, communication, data analytics and soft computing (ICECDS), Chennai (pp. 2319–2326).

[9] Noorian, F., & Leong, P. H. W. (2017). On-time series forecasting error measures for finite horizon control. IEEE Transactions on Control Systems Technology, 25(2), 736–743.