**Industry Project**

**Report**

**On**

**IoT DATA ANALYTICS**

**Developed By:- Guided By:-**

Archan VyasProf. Ravindra Patel (Internal)

Mr.Nikunj Thakkar (External)

**Submitted to**

**Faculty of Engineering and Technology**

**Institute of Computer Technology**

**Ganpat University**

****

**Year – 2024**

# 

# **ACKNOWLEDGEMENT**

Industry Internship project is a golden opportunity for learning and self-development. I consider myself very lucky and honored to have so many wonderful people lead me through in completion of this project. First and foremost, I would like to thank Dr. Rohit Patel, Principal, ICT, and Prof. Dharmesh Darji , Head, ICT who gave us an opportunity to undertake this project. My grateful thanks to Prof. Ravi Patel & Mr.Nikunj Thakkar (Internal & External Guides) for their guidance in project work IoT Data Analytics, who despite being extraordinarily busy with academics, took time out to hear, guide and keep us on the correct path. We do not know where would have been without his/her help. CSE department monitored our progress and arranged all facilities to make life easier. We choose this moment to acknowledge their contribution gratefully.

**ARCHAN VYAS (Enrollment No:21162122007)**

# **ABSTRACT**

This project focuses on developing a robust system utilizing Raspberry Pi and DHT11 sensor technology for real-time data collection of temperature and humidity. The collected data is seamlessly uploaded to MongoDB in the cloud and visualized using Power BI, enabling stakeholders to monitor environmental conditions effectively. Through rigorous data cleaning, sentiment analysis, and machine learning modeling, the system offers insights into user sentiment and application performance. Future enhancements include advanced analytics integration, predictive maintenance,and scalability optimizations, positioning the system as a versatile solution for IoT-based environmental monitoring in diversedomains.

Our project focuses on establishing a secure and efficient pipeline for transmitting sensor data from a DHT11 sensor connected to a Raspberry Pi to an AWS S3 bucket using MQTT protocol. We integrate AWS IoT Core for device management, AWS S3 for data storage, and Python scripting for data processing and transmission. By implementing end-to-end encryption and proper authentication mechanisms, we ensure the confidentiality and integrity of the transmitted data. This solution offers scalability, reliability, and seamless integration with AWS services, laying the foundation for advanced IoT applications and data-driven decision-making.

**INDEX**

**TITLE PAGE NO.**

**CHAPTER:1 INTRODUCTION 06-07**

**CHAPTER:2 PROJECT SCOPE 08-09**

**CHAPTER:3 SOFTWARE AND HARDWARE REQUIREMENT 10-13**

**CHAPTER:4 PROCESS MODEL 14-15**

**CHAPTER:5 PROJECT PLAN 16-19**

**5.1 LIST OF MAJOR ACTIVITIES 16**

**5.2 ESTIMATED TIME DURATION IN DAYS 18**

**CHAPTER:6 IMPLEMENTATION DETAILS 20-26**

**6.1 FLOWCHART OF IMPLEMENTATION 21**

**6.1.1 DATA COLLECTION 21**

**6.1.2 UNDERSTANDING DATA 22**

**6.1.3 DATA VISUALIZATION 22**

**6.1.4 SETTING UP RASPBERRYPI 22**

**6.1.5 CREATING AWS RESOURCES 22**

**6.1.6 SETTING UP AWS IOT CORE 22**

**6.1.7 SETTING UP MQTT 23**

**6.1.8 PYTHON CODING ON RASPBERRYPI 23**

**6.1.9 UPLOADING DATA TO S3 23**

**6.1.10 SECURITY CONSIDERATIONS 23**

**6.2 TRADITIONAL WAY 24**

**6.2.1 SET UP THE RASPBERRY PI&DHT11 SENSOR 24**

**6.2.2 SETTING UP MONGODB 25**

**6.2.3 STORING SENSOR DATA IN MONGODB 25**

**6.2.4 DATA VISUALIZATION WITH POWER BI 25**

**6.2.5 AUTOMATION AND MONITORING 25**

**6.2.6 DOCUMENTATION AND REPORTING 26**

**6.2.7 TESTING AND DEPLOYMENT 26**

**6.2.8 FUTURE ENHANCEMENTS 26**

**CHAPTER:7 CONCLUSION AND FUTURE WORK 27-29**

**CHAPTER:8 CONCLUSION 30-31**

# **CHAPTER: 1 INTRODUCTION**

**CHAPTER 1 INTRODUCTION**

The rapid advancements in IoT (Internet of Things) technology have revolutionized data acquisition and monitoring across various domains. This project focuses on integrating a DHT11 sensor with a Raspberry Pi to collect environmental data, specifically temperature and humidity readings. The sensor data will then be transmitted to AWS (Amazon Web Services) S3 using MQTT (Message Queuing Telemetry Transport) protocol for efficient storage and management. From S3, the data will be further processed using AWS Glue and loaded into AWS Redshift, enabling scalable and performant data warehousing. Finally, leveraging AWS Redshift's capabilities, the processed data will be visualized in PowerBI, facilitating insightful data analysis and decision-making.

**OBJECTIVES:-**

The primary objective of this project is to demonstrate a seamless end-to-end IoT data pipeline using cloud-native services. Specifically, the project aims to:

Establish a reliable data collection mechanism from a DHT11 sensor connected to a Raspberry Pi.

Implement secure and efficient data transmission using MQTT over AWS IoT Core.

Store sensor data in AWS S3 for scalable and durable cloud storage.

Utilize AWS Glue for data preparation and transformation tasks before loading into AWS Redshift.

Leverage AWS Redshift's data warehousing capabilities to enable complex querying and analysis of sensor data.

Visualize the processed data in PowerBI for intuitive and actionable insights.

This project not only showcases the integration of hardware (Raspberry Pi and DHT11 sensor) with cloud services but also emphasizes the importance of data orchestration, transformation, and visualization in IoT applications. By achieving these objectives, the project aims to provide a comprehensive example of leveraging AWS services for IoT datamanagement and analysis.

**CHAPTER:2 PROJECT SCOPE**

**CHAPTER:2 PROJECT SCOPE**

The project's primary goal is to establish a robust IoT data pipeline using a Raspberry Pi and AWS cloud services, with specific objectives aimed at achieving the following deliverables:

* IoT Data Collection: Implement a reliable method to capture temperature and humidity readings from a DHT11 sensor connected to a Raspberry Pi.
* Data Transmission to AWS S3: Utilize MQTT protocol and AWS IoT Core to securely transmit sensor data to AWS S3 for storage and archival.
* Data Processing with AWS Glue: Employ AWS Glue for data transformation and preparation, ensuring data quality and format compatibility for downstream processing.
* Data Warehousing with AWS Redshift: Load transformed sensor data into AWS Redshift, enabling scalable and efficient data warehousing capabilities for analytics and querying.
* Data Visualization using PowerBI: Integrate AWS Redshift with PowerBI to create interactive dashboards and visualizations for real-time monitoring and analysis of IoT sensor data.

**Limitations and Constraints**

Despite the ambitious scope of the project, certain limitations and constraints will guide its development and implementation:

* Hardware Constraints: The project is limited by the capabilities and processing power of the Raspberry Pi and DHT11 sensor, which may impose restrictions on data collection frequency and volume.
* Network and Connectivity: Reliable internet connectivity is essential for MQTT-based data transmission to AWS S3 and subsequent AWS services. Connectivity issues or network limitations may impact data transmission and real-time processing.
* AWS Service Dependencies: The project relies heavily on AWS cloud services (IoT Core, S3, Glue, Redshift), requiring adherence to AWS service limits, configurations, and best practices for optimal performance and cost-efficiency.
* Security Considerations: Implementing robust security measures (e.g., encryption, access controls) is critical to protect IoT data throughout the data pipeline, considering potential vulnerabilities in IoT devices and cloud services.
* Resource Management: Efficient resource utilization and cost management are essential due to the potential scalability and cost implications of AWS services, particularly AWS Redshift for data warehousing.

**CHAPTER:3 SOFTWARE AND HARDWARE REQUIREMENTS**

**CHAPTER:3 SOFTWARE AND HARDWARE REQUIREMENTS**

**TRADTIONAL WAY**

The successful implementation of the project relies on the integration of the following hardware components:

1. **Raspberry Pi:** The core component of the system, serving as the computing platform to collect data from the DHT11 sensor, process it, and transmit it to the cloud. Raspberry Pi offers versatility, affordability, and low power consumption, making it an ideal choice for IoT applications.

2. **DHT11 Sensor:** A digital temperature and humidity sensor that interfaces with the Raspberry Pi to capture real-time environmental data. The sensor provides accurate readings with a relatively simple interface, making it suitable for monitoring applications.

3. **MicroSD Card:** Used to store the operating system (e.g., Raspbian) and software applications required for running the Raspberry Pi.

4. **Power Supply:** Provides the necessary power to operate the Raspberry Pi and peripherals. It is crucial to select a stable and reliable power source to ensure uninterrupted operation.

5. **Internet Connection:** Enables connectivity to the cloud platform for transmitting data collected by the Raspberry Pi. This can be achieved through Ethernet or Wi-Fi connectivity options available on the Raspberry Pi.

The software stack comprises the following essential components necessary for the development and operation of the system:

1. **Raspberry Pi Operating System:** The Raspberry Pi requires an operating system to function. Raspbian, a Debian-based Linux distribution optimized for Raspberry Pi, is commonly used due to its compatibility and support for various IoT applications.

2. **Python Programming Language:** Python serves as the primary programming language for developing software modules to interact with the DHT11 sensor, process data, and establish communication with the cloud platform. Python's simplicity, versatility, and extensive libraries make it well-suited for IoT development.

3. **MongoDB Database:** MongoDB, a NoSQL database, is employed for storing the collected environmental data in the cloud. MongoDB offers scalability, flexibility, and robust data management capabilities, making it an ideal choice for handling IoT-generated data.

4. **Power BI:** Microsoft Power BI is utilized for data visualization and analytics purposes. It enables the creation of interactive dashboards and reports to visualize the stored environmental data effectively.

5.**Cloud Platform:** Depending on the project requirements and preferences, a cloud platform such as AWS (Amazon Web Services), Microsoft Azure, or Google Cloud Platform may be utilized for hosting the MongoDB database and facilitating cloud-based data storage and management.

**CLOUD FRAMEWORK**

**Software Components**

The successful implementation of the IoT data pipeline project requires several key software components, primarily centered around AWS cloud services and supporting technologies:

**AWS Services:**

1. AWS IoT Core: This service facilitates communication between IoT devices (such as the Raspberry Pi) and the cloud. It manages device connections, handles message routing with MQTT, and supports device state management and security.

* Amazon S3 (Simple Storage Service): S3 is utilized for storing the raw sensor data transmitted from the Raspberry Pi. It provides scalable, durable, and highly available object storage with flexible data retrieval options.
* AWS Glue: AWS Glue is employed for data preparation and transformation tasks. It offers ETL (Extract, Transform, Load) capabilities to cleanse, normalize, and structure the sensor data before loading it into AWS Redshift.
* Amazon Redshift: Redshift serves as the data warehousing solution for storing and querying large volumes of structured data. It enables efficient data analysis and supports complex SQL queries for deriving insights from the IoT sensor data.

**2. MQTT Protocol:**

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol ideal for IoT applications. It ensures efficient, low-latency communication between the Raspberry Pi (acting as the MQTT publisher) and AWS IoT Core (acting as the MQTT broker).

**3. PowerBI for Visualization:**

PowerBI is chosen as the visualization tool to create interactive dashboards and reports based on the processed data stored in AWS Redshift. It provides powerful data visualization capabilities for real-time monitoring and analysis of IoT sensor metrics.

**Hardware Components**

The hardware components selected for this project form the foundation of the IoT data collection system:

**1. Raspberry Pi:**

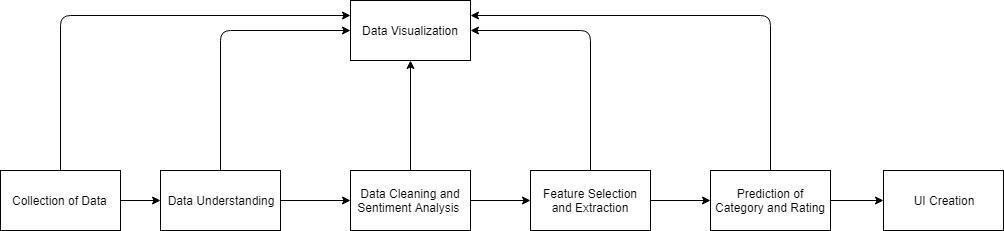
The Raspberry Pi serves as the edge computing device responsible for interfacing with the DHT11 sensor and transmitting the collected data to AWS S3 via MQTT. Its compact size, low cost, and GPIO (General-Purpose Input/Output) capabilities make it an ideal choice for IoT applications.

**2. DHT11 Sensor:**

The DHT11 sensor is used to capture environmental parameters such as temperature and humidity. It connects directly to the Raspberry Pi's GPIO pins and provides digital output, simplifying integration with the IoT data collection system.

**CHAPTER:4 PROCESS MODEL**

**CHAPTER:4 PROCESS MODEL**



**CHAPTER:5 PROJECT PLAN**

**CHAPTER:5 PROJECT PLAN**

**5.1 LIST OF MAJOR ACTIVITIES**

**Planning and Design:**

**Objective:** Define project scope, requirements, and architecture.

**Tasks:**

* Conduct requirements gathering and analysis.
* Define system architecture and components (Raspberry Pi setup, AWS services).
* Develop data flow diagrams and sequence diagrams.
* Create a detailed project plan outlining tasks, timelines, and dependencies.

**Implementation:**

**Objective:** Build and configure the IoT data pipeline as per the design.

**Tasks:**

* Set up Raspberry Pi with DHT11 sensor and necessary libraries.
* Establish connectivity to AWS IoT Core and configure MQTT communication.
* Implement Python scripts for data collection and transmission to AWS S3.
* Create AWS resources (S3 bucket, Glue job, Redshift cluster) and configure data processing pipeline.
* Develop ETL (Extract, Transform, Load) jobs using AWS Glue to prepare sensor data for storage in Redshift.
* Integrate AWS Redshift with PowerBI for data visualization.

**Testing and Validation:**

**Objective:** Ensure functionality, reliability, and performance of the IoT data pipeline.

**Tasks:**

* Conduct unit testing of individual components (Raspberry Pi data collection, MQTT communication).
* Perform integration testing to validate end-to-end data flow from sensor to Redshift.
* Validate data accuracy, integrity, and transformation quality using sample datasets.
* Test PowerBI dashboards for real-time visualization and responsiveness.

**Deployment:**

**Objective:** Prepare the system for production deployment.

**Tasks:**

* Document deployment procedures and configurations.
* Set up monitoring and alerting for AWS services (CloudWatch, AWS Glue job monitoring).
* Create user manuals and documentation for system operation and maintenance.
* Plan and execute deployment to production environment following best practices.

**5.2 Estimated Time Duration in Days (Gantt Chart)**

The project timeline is estimated based on the activities outlined above and may vary depending on resource availability and project complexity. Below is a high-level Gantt chart representing the estimated time duration for each project phase:

Planning and Design: 10 days

* Requirements gathering and analysis: 3 days
* System architecture and design: 4 days
* Project plan development: 3 days

Implementation: 20 days

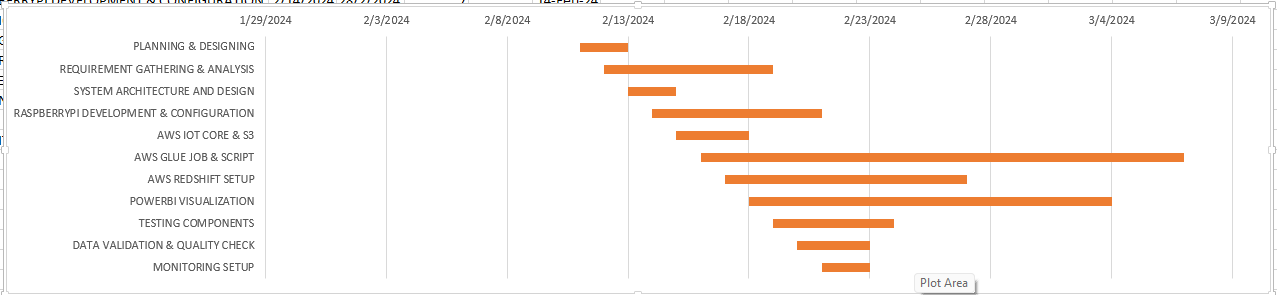
* Raspberry Pi setup and configuration: 4 days
* AWS IoT Core and S3 configuration: 5 days
* Python scripting and AWS Glue job development: 6 days
* AWS Redshift setup and integration with PowerBI: 5 days

Testing and Validation: 10 days

* Unit testing of components: 3 days
* Integration testing of IoT data pipeline: 4 days
* Data validation and quality checks: 3 days

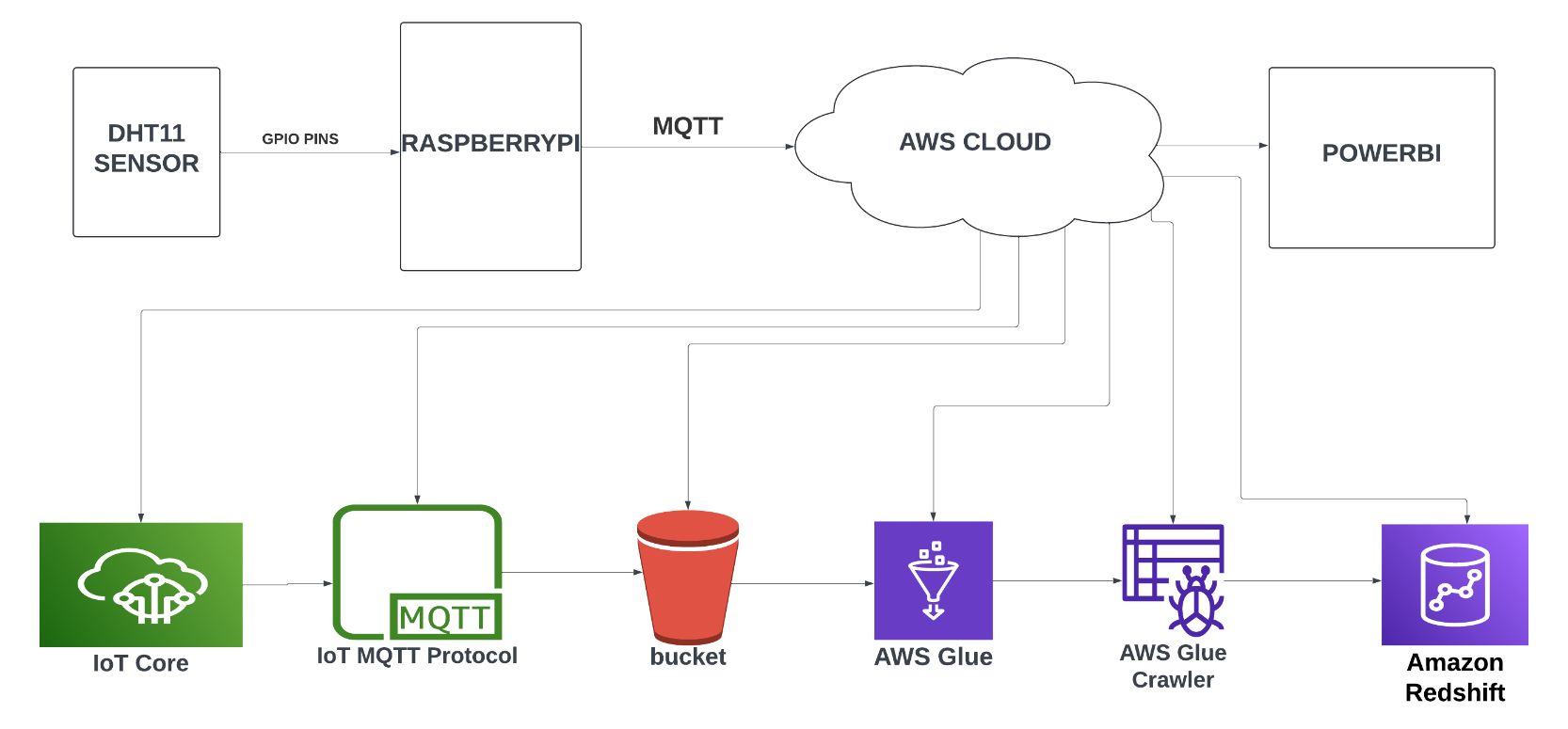
Deployment: 5 days

* Documentation and user manual preparation: 2 days
* Production deployment and monitoring setup: 3 days



**CHAPTER:6 IMPLEMENTATION DETAILS**

**CHAPTER:6 IMPLEMENTATION DETAILS**

****

**6.1 IOT FRAMEWORK WITH CLOUD**

This chapter outlines the detailed implementation steps for setting up the IoT data pipeline, encompassing data collection, understanding, visualization, hardware setup, AWS resource creation, MQTT setup, Python coding, data uploading, and security considerations.

**6.1.1 Data Collection**

Setup of DHT11 Sensor with Raspberry Pi:

* Connect the DHT11 sensor to the Raspberry Pi's GPIO pins.
* Install necessary libraries (e.g., Adafruit\_DHT) to interface with the sensor.
* Validate sensor connectivity and test data acquisition.

Data Acquisition and Formatting:

Develop Python scripts to read sensor data (temperature, humidity) from the DHT11 sensor.

Format the sensor data into a suitable message payload for MQTT transmission.

**6.1.2 Understanding Data**

Interpretation of Sensor Data:

* Implement logic to interpret raw sensor readings (e.g., temperature in Celsius/Fahrenheit, humidity percentage).
* Handle edge cases and outlier data to ensure data integrity.

Data Integrity and Quality Checks:

* Perform data validation to identify and filter out erroneous readings.
* Implement data preprocessing techniques (e.g., smoothing, averaging) to enhance data quality.

**6.1.3 Data Visualization**

Importance of Visualization in IoT Data:

* Explain the significance of visualizing IoT sensor data for actionable insights and decision-making.
* Highlight the role of real-time dashboards in monitoring environmental parameters.

Integration with PowerBI for Real-time Dashboards:

* Integrate AWS Redshift (data warehouse) with PowerBI for interactive and dynamic data visualization.
* Design and develop PowerBI dashboards to display sensor data trends and anomalies.

**6.1.4 Setting up Raspberry Pi**

Installation of Raspbian OS:

* Install the Raspbian operating system (OS) on the Raspberry Pi.
* Configure network settings and SSH access for remote management.

Configuration of Python Environment:

* Set up a Python development environment on the Raspberry Pi.
* Install required Python packages for sensor data processing and MQTT communication.

**6.1.5 Creating AWS Resources**

Setup of AWS Account and Services:

* Create an AWS account and configure billing settings.
* Provision necessary AWS services including S3 (Simple Storage Service), Glue, and Redshift.

Creation of S3 Bucket, Glue Job, Redshift Cluster:

* Create an S3 bucket to store sensor data.
* Configure AWS Glue jobs for data transformation and ETL tasks.
* Set up an AWS Redshift cluster for data warehousing and analytics.

**6.1.6 Setting up AWS IoT Core**

Configuration of IoT Core:

* Configure AWS IoT Core to manage IoT devices and message broker.
* Define IoT Core policies and rules for secure communication.

Device Registration and Policy Management:

* Register the Raspberry Pi as an IoT device within AWS IoT Core.
* Implement device policies to control access and permissions.

**6.1.7 Setting up MQTT**

MQTT Protocol Setup for Communication:

* Implement MQTT client on the Raspberry Pi using Python.
* Establish secure MQTT communication with AWS IoT Core.

Security Considerations for MQTT:

* Configure TLS (Transport Layer Security) for encrypted communication.
* Implement authentication mechanisms (e.g., client certificates) for MQTT security.

**6.1.8 Python Coding on Raspberry Pi**

Development of Python Scripts for Data Transmission:

* Write Python scripts to publish sensor data to AWS IoT Core using MQTT.
* Implement error handling and retry mechanisms for reliable data transmission.

**6.1.9 Uploading Data to AWS S3**

Implementation of Data Upload Mechanism:

* Develop Python scripts to upload formatted sensor data to AWS S3.
* Handle data streams and batching to optimize data upload performance.

**6.1.10 Security Considerations**

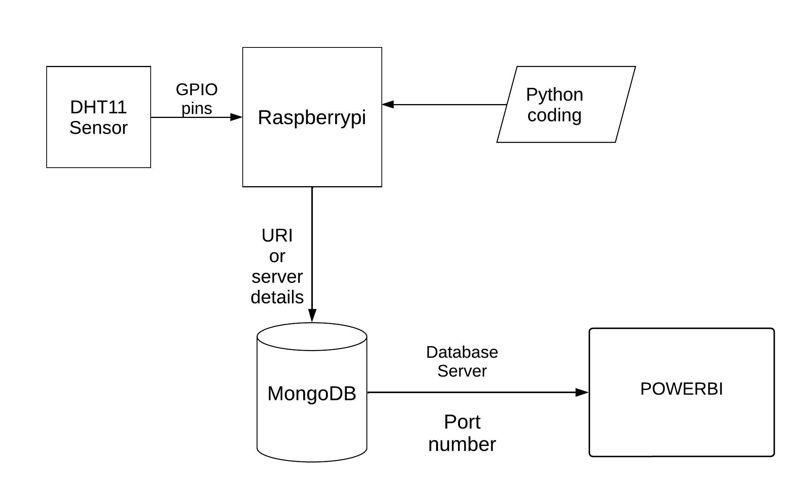
Data Encryption and Access Control:

* Implement data encryption at rest and in transit using AWS services.
* Configure access controls and IAM (Identity and Access Management) policies for secure data handling.

AWS IAM Policies for Secure Data Handling:

* Define IAM roles and policies to restrict access to AWS resources based on least privilege principles.
* Implement logging and monitoring to detect and respond to security incidents.

**TRADITIONAL WAY**

****

**6.2 IOT FRAMEWORK WITHOUT CLOUD**

**IMPLEMENTATION DETAILS(TRADITIONAL WAY):-**

**6.2.1. Setting Up the Raspberry Pi and DHT11 Sensor**

Hardware Setup:

* Connect the DHT11 sensor to the Raspberry Pi GPIO pins.
* Ensure the Raspberry Pi is properly powered and connected to the internet.

Software Installation:

* Install necessary libraries on the Raspberry Pi

Write Python Script:

* Create a Python script to read data from the DHT11 sensor periodically.
* Use the Adafruit\_DHT library to interact with the sensor.
* Implement error handling and data parsing in your script.

**6.2.2. Setting Up MongoDB**

Installation:

* Install MongoDB on the Raspberry Pi or on a separate server
* Database and Collection Creation:
* Create a MongoDB database and collection to store sensor data.

**6.2.3. Storing Sensor Data in MongoDB**

Integration with Python Script:

* Modify the Python script to establish a connection to MongoDB.
* Store the sensor data (e.g., temperature and humidity readings) in the MongoDB collection.

**6.2.4. Data Visualization with Power BI**

MongoDB Connection Setup:

* Launch Power BI Desktop on your computer.
* Click on "Get Data" and select MongoDB.
* Enter the MongoDB connection details (host, port, database, collection).

Data Transformation and Visualization:

* Load the data into Power BI.
* Transform the data as needed (e.g., convert data types).
* Create visualizations (e.g., line charts, gauges) based on the sensor data.

**6.2.5. Automation and Monitoring**

Automation with Cron Job:

* Set up a cron job on the Raspberry Pi to run the Python script at regular intervals (e.g., every 5 minutes).
* Monitoring and Error Handling:
* Implement logging and error handling in the Python script to monitor sensor readings and data insertion into MongoDB.
* Set up alerts or notifications for any critical issues.

**6.2.6. Documentation and Reporting**

Document the Entire Setup:

* Prepare a detailed report documenting each step of the implementation.
* Include configuration details, code snippets, and screenshots of data visualizations.
* Provide troubleshooting tips and recommendations for further improvements.

**6.2.7. Testing and Deployment**

Testing the Complete System:

* Conduct thorough testing of the integrated system.
* Ensure sensor data is consistently stored in MongoDB and visualized accurately in Power BI.

**6.2.8. Future Enhancements**

Explore Additional Features:

* Consider adding features such as real-time data streaming, historical data analysis, or integrating other sensors.
* Evaluate scalability options for handling larger datasets or multiple sensors.

**CHAPTER:7 CONCLUSION AND FUTURE WORK**

**CHAPTER:7 CONCLUSION AND FUTURE WORK**

**Conclusion**

In conclusion, the implementation details presented in this report outline a comprehensive approach to setting up an IoT data pipeline using a Raspberry Pi, DHT11 sensor, and AWS cloud services. By following the outlined steps, the project aims to achieve the following objectives:

* Successful Data Collection and Transmission: Establishing reliable data collection from the DHT11 sensor connected to the Raspberry Pi, and transmitting the collected data securely to AWS S3 using MQTT protocol and AWS IoT Core.
* Efficient Data Processing and Storage: Utilizing AWS Glue for data transformation and ETL tasks to prepare the sensor data for storage in AWS Redshift, a scalable data warehousing solution.
* Interactive Data Visualization: Integrating AWS Redshift with PowerBI for real-time dashboards and visualizations, enabling stakeholders to monitor and analyze IoT sensor data effectively.
* Ensuring Security and Reliability: Implementing security best practices, including data encryption, access control, and MQTT protocol security, to ensure the integrity and confidentiality of IoT data throughout the pipeline.

**Future Work**

While the initial implementation of the IoT data pipeline will serve as a foundation, there are several avenues for future work and enhancements to further extend the project's capabilities:

* Advanced Data Analytics: Explore advanced analytics techniques such as machine learning and anomaly detection algorithms to derive deeper insights from IoT sensor data.
* Scalability and Optimization: Optimize the data pipeline architecture for scalability to handle larger volumes of sensor data and optimize resource utilization for cost efficiency.
* Enhanced Monitoring and Alerting: Implement comprehensive monitoring and alerting mechanisms using AWS CloudWatch to proactively detect and respond to performance issues or anomalies in the IoT system.
* Integration with IoT Edge Services: Extend the project to incorporate edge computing capabilities by deploying edge devices and implementing edge analytics for real-time data processing closer to the data source.
* Integration with Additional AWS Services: Explore integration with other AWS services such as AWS Lambda for serverless computing, Amazon QuickSight for enhanced data visualization, or AWS IoT Analytics for advanced data processing.
* Exploration of IoT Use Cases: Apply the developed IoT data pipeline to various use cases beyond environmental monitoring, such as industrial IoT, smart agriculture, healthcare monitoring, or smart home applications.

By continuously iterating and expanding on the initial implementation, the project can evolve into a robust and versatile IoT solution capable of addressing diverse business and industry needs. Future work will focus on leveraging emerging technologies and best practices to enhance the efficiency, scalability, and intelligence of the IoT data pipeline.

**CHAPTER:8 REFRENCES**

**CHAPTER:8 REFRENCES**

* 1. [https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-](https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/) [play-store/](https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/)
  2. <https://towardsdatascience.com/the-tale-of-missing-values-in-python-c96beb0e8a9d>
  3. <https://www.listendata.com/2017/05/feature-selection-boruta-package.html>
  4. <https://www.analyticsvidhya.com/blog/2016/03/select-important-variables-boruta-package/>
  5. <https://www.machinelearningplus.com/machine-learning/feature-selection/>
  6. <https://repost.aws/knowledge-center/iot-core-publish-mqtt-messages-python>
  7. <https://www.youtube.com/watch?v=GqmNvcnXQOQ&list=LL&index=1&t=2073s>
  8. <https://www.youtube.com/watch?v=-ROGwBtKazc&list=LL&index=3>

**THANK YOU**