Raspberry Pi-Based Data Ingestion to Cloud Warehouses with ETL Automations

Obinna Hillary Ibekwe(315376)

Faculty of Computer Science

Schmalkalden University of Applied Sciences

February 21, 2025

1 Abstract

This project explores the design and implementation of a Raspberry Pi-based system for efficient data ingestion from a DHT22 sensor to cloud data warehouses, utilizing Extract, Transform, Load (ETL) automation. The system is aimed at real-time data processing for applications like environmental monitoring, where temperature and humidity readings from the sensor are collected, processed, and seamlessly integrated into AWS DynamoDB for storage. The ETL automation tools handle tasks such as data extraction from the sensor, transformation into the required format, and loading into DynamoDB. Additionally, the system features a web interface that displays real-time sensor data and provides a user-friendly view of the collected information. The project highlights the use of the Raspberry Pi as a cost-effective, scalable edge computing device for IoT applications, enabling minimal manual intervention through automated ETL processes. Real-time data ingestion is achieved with AWS IoT Core, while the web interface allows easy monitoring of the stored data. The results demonstrate the effectiveness of the proposed system in ensuring efficient data ingestion, scalability, and easy visualization of sensor data, making it suitable for academic, industrial, and research applications in environmental monitoring and beyond.

2 Keywords

Raspberry Pi, Data Ingestion, ETL Automation, DHT22 Sensor, Cloud Warehouses, AWS DynamoDB, Edge Computing, IoT Applications, Environmental Monitoring, Real-time Data Processing, Data Transformation, Data Storage, AWS IoT Core, Sensor Data Visualization, Scalability, Web Interface

3 Scientific Background

The rapid growth of IoT has generated large amounts of data that require efficient collection, processing, and storage. Data ingestion into cloud warehouses is key for managing this data, especially with IoT devices. Raspberry Pi, a low-cost microcomputer, is increasingly used for edge computing, processing data near the source before sending it to the cloud.

ETL (Extract, Transform, Load) automation streamlines data pipelines, converting raw IoT data into a structured format for integration with cloud warehouses like Amazon Redshift, Snowflake, or DynamoDB. This reduces manual work and enhances scalability for real-time applications.

Cloud data warehouses enable scalable storage and advanced analytics for IoT, supporting real-time and batch processing. AWS IoT Core and DynamoDB Streams facilitate seamless communication between IoT devices and the cloud, ensuring real-time data ingestion.

The Raspberry Pi's ability to interface with sensors, combined with its low cost and energy efficiency, makes it ideal for edge computing. Python's extensive libraries, like Boto3 for AWS, provide robust tools for ETL and cloud communication.

This project demonstrates how a Raspberry Pi-based system optimizes data ingestion workflows, integrating cloud services and automation tools to offer scalable, real-time data processing solutions for various applications.

3.1 Evolution Of Raspberry Pi-Based Data Ingestion to Cloud Warehouse

The idea of utilizing Raspberry Pi for data ingestion into cloud data warehouses with automated ETL processes has progressed alongside developments in IoT technology, edge computing, and cloud services. This progression can be seen in several important stages:

1. **Early IoT and Edge Computing Solutions**
    - **Initial Data Handling Challenges:** In the early stages, IoT systems primarily concentrated on gathering data locally or sending raw data directly to centralized servers. These methods encountered issues such as limited bandwidth, high latency, and poor resource efficiency.
    - **Emergence of Edge Devices:** The emergence of affordable and adaptable microcomputers, such as the Raspberry Pi, offered a cost-effective platform for carrying out basic computational tasks near the data source.
2. **Integration of ETL Pipelines**
    - **Manual Data Processing:** Initially, data extraction, cleaning, and structuring processes were manual or relied on simple scripts.
    - **Automation Introduction:** With the rise of ETL (Extract, Transform, Load) frameworks, automation was introduced, allowing for streamlined and repeatable data preparation workflows. Tools such as Apache Airflow and AWS Glue further supported this by offering strong scheduling and transformation capabilities.
    - **Raspberry Pi as an ETL Node:** Raspberry Pi, with its ability to interface with various sensors and run Python, started being utilized as an edge device for performing ETL tasks, thereby reducing the need to send raw data to the cloud.
3. **Growth of Cloud-Based Data Warehouses**
    - **Cloud Revolution:** Platforms such as Amazon Redshift, Snowflake, and DynamoDB transformed data storage by providing scalability, cost-effectiveness, and advanced analytics capabilities.
    - **IoT-Specific Features:** Services like AWS IoT Core and DynamoDB Streams were developed to meet the specific demands of IoT applications, enabling real-time data ingestion and processing.
4. **Modern Integration and Real-Time Processing**
    - **Hybrid Workflows:** Modern systems integrate both real-time and batch processing to optimize performance. Raspberry Pi is employed for initial data filtering and summarization, while cloud warehouses manage large-scale storage and advanced analytics.
    - **Enhanced Automation:** With the advent of tools like AWS Lambda and serverless computing, ETL pipelines now operate with lower latency and reduced operational overhead.
    - **improved Sensor Interfacing:** Modern sensors like the DHT22 are seamlessly integrated with Raspberry Pi, enabling efficient environmental data collection that is processed and transmitted to the cloud in near real-time.
5. **Current Trends and Future Directions**
    - **AI-Driven Data Processing:** Machine learning models running on Raspberry Pi improve ETL pipelines by allowing intelligent data filtering and anomaly detection prior to transmission.
    - **IoT Ecosystem Expansion:** The continued growth of IoT networks and improved cloud capabilities are driving the demand for scalable, energy-efficient, and low-latency data ingestion systems.
    - **Energy Efficiency Focus:** As energy concerns increase, the Raspberry Pi's low power consumption makes it an appealing option for IoT and edge computing applications.

This evolutionary journey demonstrates how Raspberry Pi-based ETL solutions have become essential to efficient data ingestion workflows, connecting IoT devices and cloud data warehouses for real-time, scalable, and automated data processing.

3.2 Related Work

According to P Rayavel et al [5] they proposed a system that aims to provide a personal, cost-efficient cloud storage solution by using hard disks controlled by a Raspberry Pi. Unlike private cloud services, this approach keeps data under the user's control, enhancing security and eliminating reliance on external providers. The system is compact, portable, and accessible from anywhere, ensuring data security through user-implemented measures. This setup offers a reliable, affordable alternative to conventional cloud storage services.

S. Emima Princy and K. Gerard Joe Nigel implemented a cloud server for real time data storage using Raspberry Pi. [4] A Raspberry Pi can be configured as a private cloud server for storing real-time signal data. It serves as a cost-effective microprocessor that leverages cloud platforms from specific vendors. Analog signals from environmental sensors are digitized using microcontrollers like Arduino or ADCs and transmitted to the Raspberry Pi. This setup enables the Raspberry Pi to act as a storage device for real-time applications.

JIR Molano et al [2] discusses the Internet of Things (IoT), its principles, and the technologies enabling communication between people and objects. It highlights various applications and emphasizes IoT's significance. Additionally, it describes a monitoring prototype using a Raspberry Pi, cloud storage, and a mobile device, demonstrating practical IoT implementation.

Dhvani Shah et al [6] explored how biometrics can utilize cloud computing's scalability, flexibility, and cost-efficiency to improve performance and reduce system requirements. A low-cost biometric system is implemented using Raspberry Pi as a remote enrollment node and integrates cryptographic algorithms like RSA and enhanced AES-256 for secure data transmission to the cloud. The system enables biometric authentication via Azure cloud services, combining IoT and biometrics to pioneer new research in secure, cloud-based IoT biometrics.

Changqing Sun et al [7] developed a Modbus to MQTT gateway for Industrial IoT applications using a Raspberry Pi and RS485 add-on board. The gateway enables two-way communication between field devices and cloud applications by converting Modbus RTU data into MQTT protocol. It supports cloud integration (e.g., AWS, Alibaba Cloud) for analysis and features web-based configuration and over-the-air updates, providing a secure and efficient solution for IIoT systems while retaining existing operational technology.

Prabha et al developed a Cloud robotics system for collecting and storing environmental data (temperature, humidity, air quality, GPS) [3] on a private cloud, with real-time viewing via a web browser.

Donca et al. [1] discusses an environmental monitoring system using a Raspberry Pi cluster and BME680 sensor within a Kubernetes framework. It emphasizes security enhancements with HashiCorp Vault and OpenID Connect to address IoT vulnerabilities. Security tests show improved system performance and reduced vulnerabilities. The research highlights scalable and secure IoT solutions for various applications, including industrial and urban environments.

4 Selected Open Topic

Efficient data collection and storage are vital for IoT applications, especially those requiring real-time monitoring and analysis. This topic focuses on utilizing a Raspberry Pi to capture data from a DHT22 sensor and upload it to AWS DynamoDB using Python. The DHT22 sensor, known for its precision in measuring temperature and humidity, is widely used in IoT systems due to its reliability and simplicity.

The Raspberry Pi serves as the edge device, interfacing with the DHT22 sensor to collect environmental data. Python scripts are employed to read sensor values, preprocess the data, and securely transmit it to DynamoDB, a highly scalable NoSQL database. This setup enables real-time storage and retrieval of sensor data, ensuring seamless integration with cloud-based analytics and monitoring tools.

The project employs a structured approach that includes configuring the Raspberry Pi, integrating the DHT22 sensor, and using the AWS SDK for Python (Boto3) to interact with DynamoDB. The data is stored with a schema designed for efficient querying, using attributes such as sensor ID, timestamp, temperature, and humidity. The use of DynamoDB ensures low-latency performance and high availability, making it ideal for IoT applications.

To enhance usability and accessibility, the project incorporates a web-based data visualization dashboard. The dashboard retrieves data from DynamoDB and displays it in a user-friendly format. Framework such as flask is employed to create the web server. The web-based interface allows users to monitor data in real time and gain insights into environmental conditions.

By leveraging Python, Raspberry Pi and web technology, this system provides a cost-effective and flexible solution for environmental monitoring. The integration with DynamoDB offers

scalability and reliability, enabling the system to handle large volumes of data while maintaining performance. This project highlights the potential of combining edge computing with cloud storage to build robust IoT systems capable of real-time data ingestion and analysis.

5 Methodology

To achieve efficient data ingestion into cloud warehouses, this project employs a structured methodology integrating Raspberry Pi, ETL automation tools, cloud platforms, and a web-based user interface. The process is divided into the following steps:

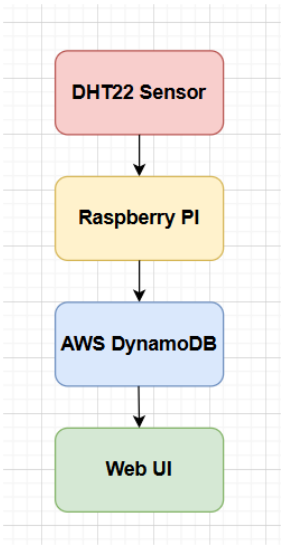

Figure 1: Flow Diagram

1. **Hardware and Software Configuration**

   - **Sensor Integration:** The DHT22 sensor is connected to the Raspberry Pi's GPIO pins, with necessary configurations for power, data, and ground.

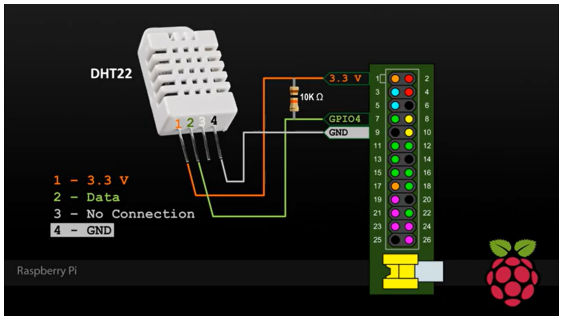

Figure 2: Sensor Integration

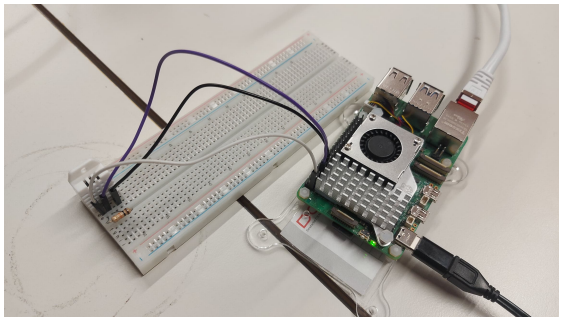

Figure 3: Hardware Integration

   - **System Preparation:** The Raspberry Pi is configured with Python and libraries such as Adafruit_DHT for sensor communication and Boto3 for AWS DynamoDB interaction

2. **Data Acquisition**

   - A Python script reads real-time temperature and humidity data from the DHT22 sensor at predefined intervals.
   - Collected data is validated to eliminate erroneous readings caused by sensor noise or hardware issues.

3. **Data Transmission**

   - **Schema Design:** A DynamoDB table is created with a schema including ID (partition key), Timestamp (sort key), Temperature, and Humidity.

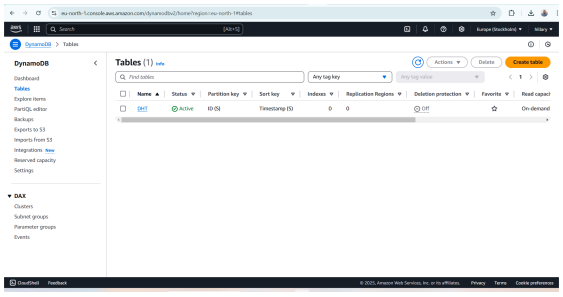

Figure 4: DynamoDB table

   - **Data Upload:** The Python script formats the sensor data into JSON objects and uploads it to DynamoDB using the AWS SDK (Boto3).
   - **Batch Processing:** For efficiency, data is uploaded in batches if multiple readings are collected within a short timeframe.

4. **Error Handling**

- Network interruptions and AWS API failures are managed using retry logic and error-handling mechanisms in the Python script.
- Logs are generated to track upload status, errors, and data inconsistencies.

5. **Testing and Validation**

   - System functionality is tested under various conditions, including fluctuating sensor readings and network connectivity issues.
   - The accuracy of the stored data is compared against manual readings from the DHT22 sensor.

6. **User Interface (UI)**

   The system incorporates a Flask-based web user interface (UI) hosted on the Raspberry Pi, enabling real-time monitoring and visualization of sensor data. This web UI provides users with an accessible and intuitive platform for interacting with the system.

   - **Framework and Implementation**
     The Flask web framework was used to build the web server, leveraging its lightweight architecture and ease of integration with Python-based applications. The web interface serves two key functionalities:
     - **Homepage (/):** Displays the latest sensor readings (temperature, humidity, and timestamp) in a user-friendly format. This page is rendered using HTML templates, making it visually accessible for users.
     - **API Endpoint:** Provides real-time sensor data in JSON format for programmatic access. This endpoint supports integration with other systems or applications requiring live environmental data.
   - **Real-Time Data Updates**
     The UI is powered by a background thread that continuously updates sensor readings in real time. The thread collects data from the DHT22 sensor and stores it in a global variable. The web interface retrieves this data dynamically, ensuring users always view the most up-to-date information.
   - **Accessibility**
     The Flask application is configured to run on the Raspberry Pi's local IP address, making it accessible from any device on the same network. Users can access the interface through a web browser by navigating to the Raspberry Pi's IP and port
   - **Scalability and Extensibility**
     The modular design of the web UI allows for easy customization and expansion. Features such as data visualization (e.g., graphs for temperature and humidity trends) or user authentication can be added without disrupting the core functionality.

   The web UI simplifies user interaction with the system by eliminating the need for direct command-line operations, offering a practical and user-friendly platform for real-time data visualization and monitoring.

6 Results

The implementation of the system yielded the following outcomes:

1. **Real-Time Data Ingestion**

   - Data from the DHT22 sensor was successfully transmitted to DynamoDB at regular intervals with minimal latency.

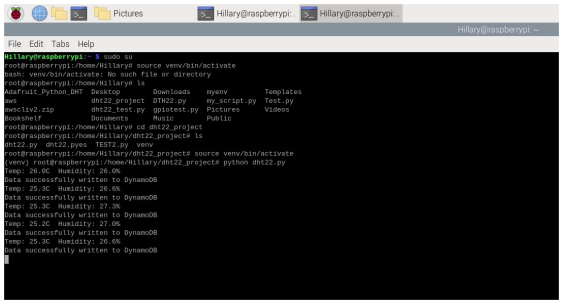

Figure 5: Real-Time Data Ingestion

   - Real-time updates were visible in the DynamoDB table, ensuring accurate and up-to-date environmental monitoring.

2. **Scalability and Reliability**

   - The system effectively handled continuous data uploads without performance degradation, highlighting DynamoDB's capacity to scale with increased data volume.

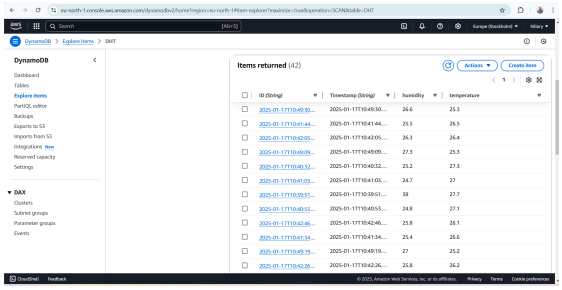

Figure 6: Real-Time Data Ingestion

- Error-handling mechanisms ensured minimal data loss during network interruptions, maintaining data consistency and reliability.

3. **Web-Based User Interface**

   - A web-based dashboard was developed to visualize the sensor data stored in DynamoDB, displaying real-time temperature and humidity readings along with timestamps for each data entry. The data updates every 5 seconds to provide the most current information.

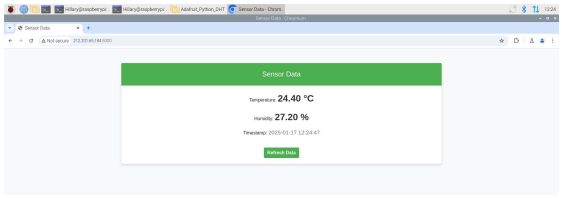

Figure 7: Web-Based User Interface

- Responsive design ensured compatibility across devices, including desktops, tablets, and mobile phones, enhancing accessibility and usability.

4. **Data Accuracy and Usability**

   - Sensor readings were validated for accuracy, with consistent performance observed during extended testing periods.

7 Conclusion/Future work

Overall, the project successfully developed a Raspberry Pi-based system for real-time data collection through ETL automation. By combining a DHT22 sensor, AWS DynamoDB, and a Flask web interface, the system effectively handles data collection, transformation, storage, and visualization for environmental monitoring purposes.

The results show the system's scalability, reliability, and automation. The real-time data ingestion pipeline, powered by AWS IoT Core and DynamoDB, ensures smooth cloud integration, while the web dashboard improves user experience by displaying live sensor data in an easy-to-understand format. The system also incorporates error-handling features to maintain data integrity, even during network interruptions.

This project demonstrates the power of edge computing and cloud integration for IoT, reducing the need for manual intervention and streamlining data workflows. With its combination of affordable hardware, cloud scalability, and automation, the system proves to be a flexible and cost-effective solution for use in academic, industrial, and research settings.

Possible future enhancements include support for additional sensors, machine learning for predictive analysis, improved security, and multi-cloud compatibility. Overall, this project showcases an effective, scalable, and reliable approach to real-time environmental monitoring and IoT data processing.

8 Open Questions

1. How well does the system perform when handling large volumes of data from multiple sensors, and what optimizations can be applied to improve the performance of data ingestion and visualization for real-time applications?

2. What mechanisms can be implemented to ensure data integrity during network failures or interruptions, and how can the system detect and handle faulty sensor readings or hardware malfunctions?

3. How can the system ensure secure data transmission between the Raspberry Pi and DynamoDB, and what are the best practices for implementing authentication and authorization for the web-based visualization interface?

4. How does the cost of running this system compare to alternative cloud storage and visualization solutions, and are there ways to reduce the cost of using AWS services without compromising system performance?

5. How easily can this system integrate with other cloud platforms like Google BigQuery or Azure Cosmos DB, and can the data visualization interface be expanded to support advanced analytics tools or frameworks?

6. What strategies can be implemented for long-term data storage and archiving without overloading the DynamoDB table, and how can the system handle data lifecycle management, such as deleting old data or migrating it to cold storage?

7. What additional features can be added to the web-based visualization to improve usability, such as custom alerts, advanced filtering, or export options, and how can the interface be made more responsive and user-friendly for different devices?

8. How can real-time data updates be improved for better responsiveness in the web interface, and what are the limitations of the current setup in achieving near-instantaneous visualization for rapidly changing sensor data?

9. How does the DHT22 sensor's accuracy vary under different environmental conditions, and how can the system compensate for such variations? Additionally, can additional sensors or features be integrated to enhance environmental monitoring?

10. How can the system be adapted to support other types of sensors or IoT devices, and what are the challenges in scaling this solution for industrial or large-scale IoT deployments?

11. 11. How can machine learning or predictive analytics be integrated into the system to provide insights or forecasts based on the collected sensor data, and what challenges might arise in implementing such advanced features?

References

[1] I.-C. Donca, O. P. Stan, M. Misaros, A. Stan, and L. Miclea. Comprehensive security for iot devices with kubernetes and raspberry pi cluster. *Electronics*, 13(9):1613, 2024.

[2] J. I. R. Molano, D. Betancourt, and G. Gómez. Internet of things: A prototype architecture using a raspberry pi. In *Knowledge Management in Organizations: 10th International Conference, KMO 2015, Maribor, Slovenia, August 24-28, 2015, Proceedings 10*, pages 618–631. Springer, 2015.

[3] S. S. Prabha, A. J. P. Antony, M. J. Meena, and S. Pandian. Smart cloud robot using raspberry pi. In *2014 International Conference on Recent Trends in Information Technology*, pages 1–5. IEEE, 2014.

[4] S. E. Princy and K. G. J. Nigel. Implementation of cloud server for real time data storage using raspberry pi. In *2015 online international conference on green engineering and technologies (IC-GET)*, pages 1–4. IEEE, 2015.

[5] P. Rayavel, N. Anbarasi, B. Renukadevi, D. Maalini, et al. Raspberry pi based secured cloud data. In *Journal of Physics: Conference Series*, volume 1964, page 042101. IOP Publishing, 2021.

[6] D. Shah et al. Iot based biometrics implementation on raspberry pi. *Procedia Computer Science*, 79:328–336, 2016.

[7] C. Sun, K. Guo, Z. Xu, J. Ma, and D. Hu. Design and development of modbus/mqtt gateway for industrial iot cloud applications using raspberry pi. In *2019 Chinese automation congress (CAC)*, pages 2267–2271. IEEE, 2019.