Assignment 2

SUMMARY REPORT

TRAN DUC ANH DANG

103995439

Executive Summary

Main Findings and Summary:

This report presents an Internet of Things (IoT)-based environmental monitoring system that uses the Adafruit_Sensor library on an Arduino to control DHT11 and LM35 sensors to measure temperature and humidity precisely. A set of advanced software tools supports the data flow of the system, from collection by the Arduino to storage and analysis on a Raspberry Pi. While the frontend is based on React and makes use of Material-UI components as well as the data visualisation capabilities of Chart.js and react-chartjs-2, the backend makes use of Flask and its CORS extension. Python scripts are used by actuators to react to environmental stimuli, showcasing the system's capacity for automated tasks like turning on fans in response to temperature thresholds. A feature-rich interface allows users to work with the monitoring data, changing parameters and looking at individual data points. This report captures the system's flexibility and potential for scaling, paving the way for advanced smart environment applications.

Table of Contents

Introduction

Purpose and Scope:

The current project aims to develop an advanced real-time environmental monitoring system that tracks temperature and humidity. This program is driven by the growing need in both home and industrial settings for monitoring systems that are responsive and intelligent. Using adaptable web apps and Internet of Things (IoT) technologies, this system aims to deliver continuous and accurate environmental data to support environmental control and informed decision making.

This project's scope includes designing and integrating a system that is dedicated to environmental monitoring and consists of both hardware and software components. The hardware setup makes use of the well respected DHT11 sensor to measure humidity and ambient temperature with accuracy, as well as the precision oriented LM35 sensor for temperature readings. An Arduino microcontroller, which connects with various sensors to gather environmental data, is at the centre of the data collecting process.

An LED light that acts as a visual signal is one of the hardware setup's standout features. The LED is set up in the system to turn on when temperature readings go beyond a certain threshold, which gives rise to an instantaneous visual alarm for noteworthy changes in the surroundings. By providing a simple and efficient way to indicate when crucial temperature levels are discovered, this threshold based LED signalling increases the usefulness of the system.

Regarding software, the project utilises a fullstack methodology. Python is used in the construction of the backend to manage server side logic and data processing, and the React framework, which is well known for its effectiveness in displaying dynamic user interfaces, is used in the frontend. With its dual faceted software configuration, the system is guaranteed to be both functional and user friendly. Real-time data visualization is provided through charts and tables, enabling prompt understanding of the environmental conditions.

This system serves as a prototype for larger scale implementations that might support more complicated environmental monitoring and control systems, in addition to providing a steady stream of sensor data for instantaneous display. The project's goal is to show that it is feasible to combine contemporary web technologies with conventional sensor gear to provide an easy to use, scalable environmental monitoring system.

Methodology

Research Design:

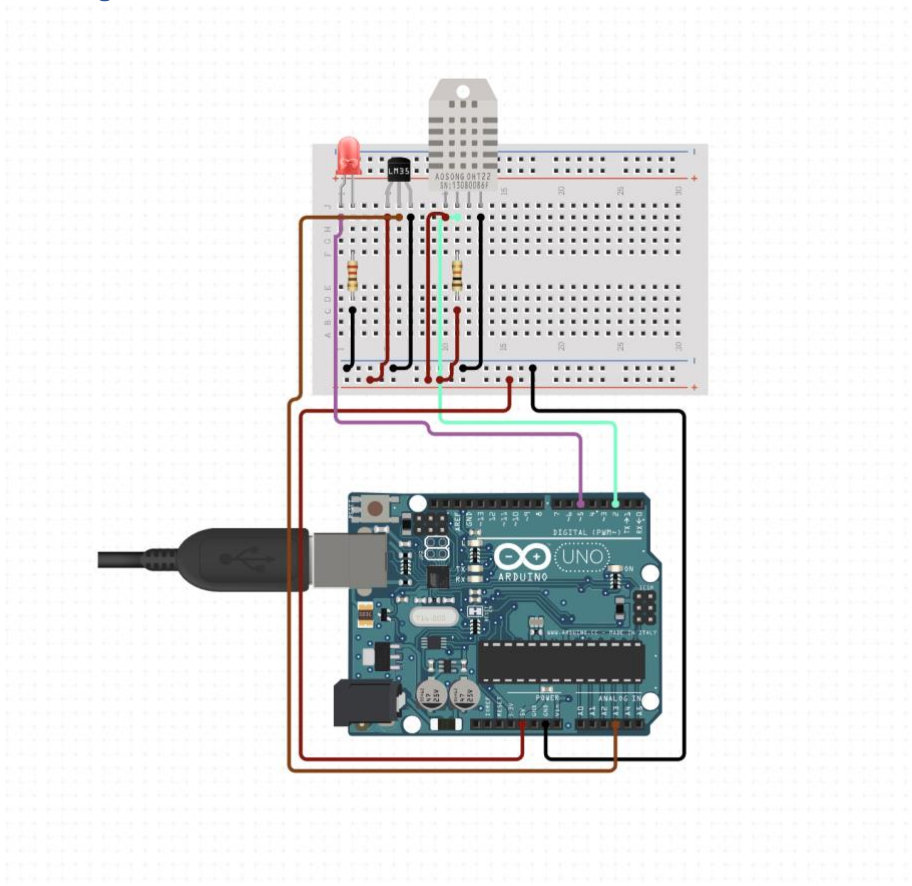


Figure 1 - Arduino Design System

The real-time environmental monitoring system, which makes use of a combination of hardware sensors and a fullstack software solution, is the focal point of the project's methodology. The hardware consists of an Arduino setup with a microprocessor that connects with three sensors: an LM35 sensor for precise temperature measurements, a DHT11 sensor for humidity and ambient temperature readings, and an LED light to provide threshold-based temperature alarms.

Data Collection:

Data from the sensors is scheduled to be periodically collected by the Arduino microcontroller. It takes measurements of the temperature and humidity, formats them, and sends them via serial connection to the backend server. The frontend and server can communicate securely and seamlessly thanks to the Python Flask application that serves as the backend and is configured to handle CORS requests.

Functions to initialise the database, make tables, and enter sensor readings into the MySQL database are all included in the backend Python script. It runs on a different thread and records the measurements into the database at predetermined intervals while continuously listening for fresh data from the Arduino over a serial connection.

System Architecture:

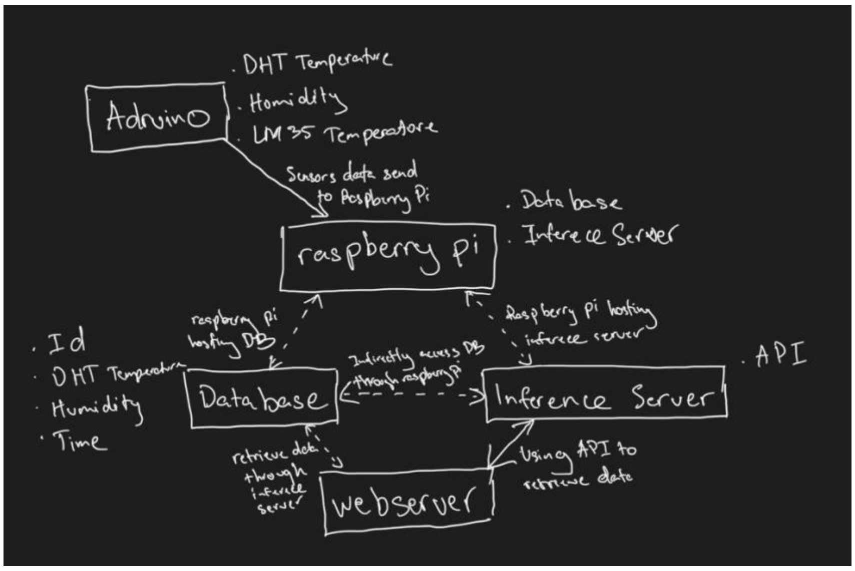


Figure 2 - System Architecture

The environmental monitoring system's architecture represents a carefully orchestrated interplay between sophisticated hardware components and a multilayered software stack, crafted to ensure the reliable acquisition, storage, and dissemination of environmental data.

Hardware Components and Data Flow:

The Arduino microcontroller, painstakingly set up to gather real-time readings from the DHT11 and LM35 sensors, is positioned at the beginning of the data trip. While the LM35 sensor is only used to provide accurate temperature readings, the DHT11 sensor is responsible for measuring both the surrounding air temperature and humidity. These sensory inputs are essential for getting a complete picture of the surrounding world.

After being collected, the data is sent from the Arduino to the Raspberry Pi, which serves as our architecture's central nervous system. Through the use of a Raspberry Pi as a conduit, sensor data is able to flow to a reliable database system, where each datum is methodically saved with a date and unique identification, guaranteeing the accuracy and traceability of the environmental measurements throughout time.

Software Infrastructure:

The Raspberry Pi hosts the central component of our software architecture, extending its capabilities beyond simple data transport. two servers consisting of an inference server and a web server Our database is protected by the web server, which also controls how stored data is retrieved upon request. The data is provided to the frontend interfaces via this server, where it appears as useful information that the end users may utilise.

Parallelly, the inference server delves into the analytical realm, potentially employing machine learning algorithms or statistical models to extrapolate predictive insights from the data, thus elevating the system's utility from reactive monitoring to proactive environmental management.

Interface and Accessibility:

A well defined API, which condenses the complexity of the backend activities into a smooth and intuitive data access experience, governs access to this abundance of information. By using this API, users may stay away from the underlying complexities of the system's internal operations while still querying the most recent environmental data, seeing historical patterns, and getting real-time changes.

This design provides a scalable and adaptable framework that can adjust to changing monitoring requirements. It also supports the system's ability to offer continuous and accurate environmental data. The design of the system clearly defines the functions of each component, ranging from data collecting to processing and visualisation. This allows the system to be applied in a variety of scenarios, ranging from complicated industrial settings to homes.

Data Analysis:

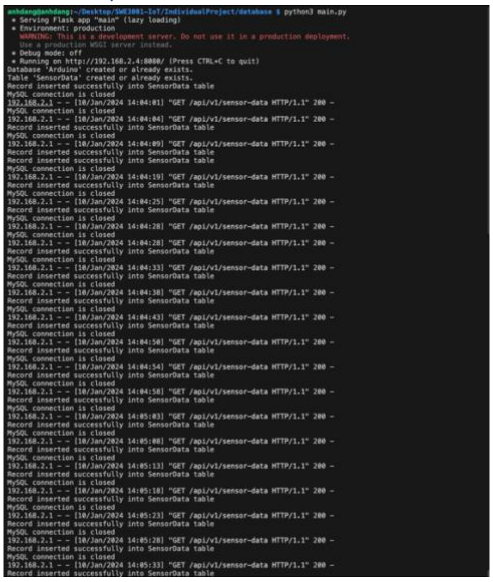
This system uses both frontend and backend processes to analyse data. The Python programmed backend server oversees keeping the sensor data in a MySQL database and providing the most recent readings via an API. This API is used by the React-developed frontend to retrieve the data, which is then used to compute average values and provide real-time visualisations.

The frontend application uses tools like Chart.js to graphically depict the data trends over time, Material-UI to design the user interface, and Axios for data retrieval. An in-depth look at the environmental conditions the system monitors is provided by the user-interactive frontend, which displays full tables of the sensor data, graphical representations, and statistical analysis such as mean temperature and humidity estimates.

By describing the methodical data collection, storage, analysis, and visualisation, this methodology offers a strong approach to environmental monitoring. It is based on an integrated hardware and software solution that is intended to be responsive and encourage user participation.

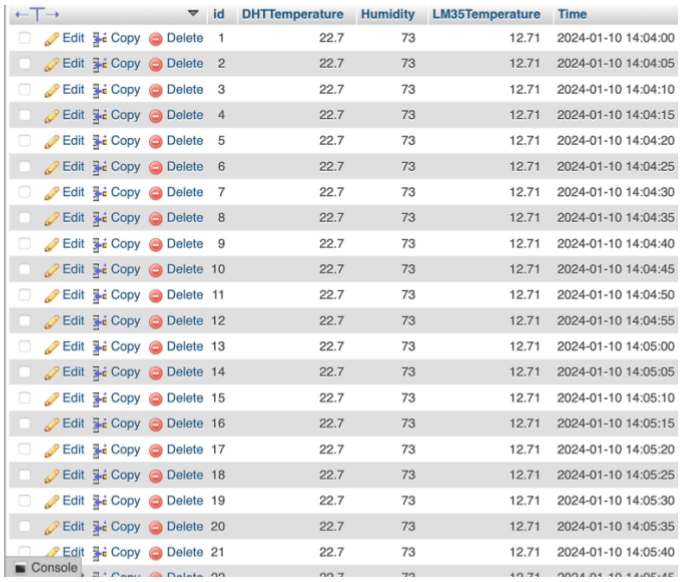
Results

Data Preparation:



```
anhdang@anhdang:~/Desktop/SWE3001-IoT/IndividualProject/database $ python3 main.py
 * Serving Flask app "main" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://192.168.2.4:8080/ (Press CTRL+C to quit)
Database 'Arduino' created or already exists.
Table 'SensorData' created or already exists.
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:04:01] "GET /api/v1/sensor-data HTTP/1.1" 200 -
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:04:04] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:04:09] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:04:19] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:04:25] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:04:28] "GET /api/v1/sensor-data HTTP/1.1" 200 -
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:04:28] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:04:33] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:04:38] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:04:43] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:04:50] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:04:54] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:04:58] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:05:03] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:05:08] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:05:13] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:05:18] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:05:23] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:05:28] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
MySQL connection is closed
192.168.2.1 - - [10/Jan/2024 14:05:33] "GET /api/v1/sensor-data HTTP/1.1" 200 -
Record inserted successfully into SensorData table
```

Figure 3 - MySQL, API Connection

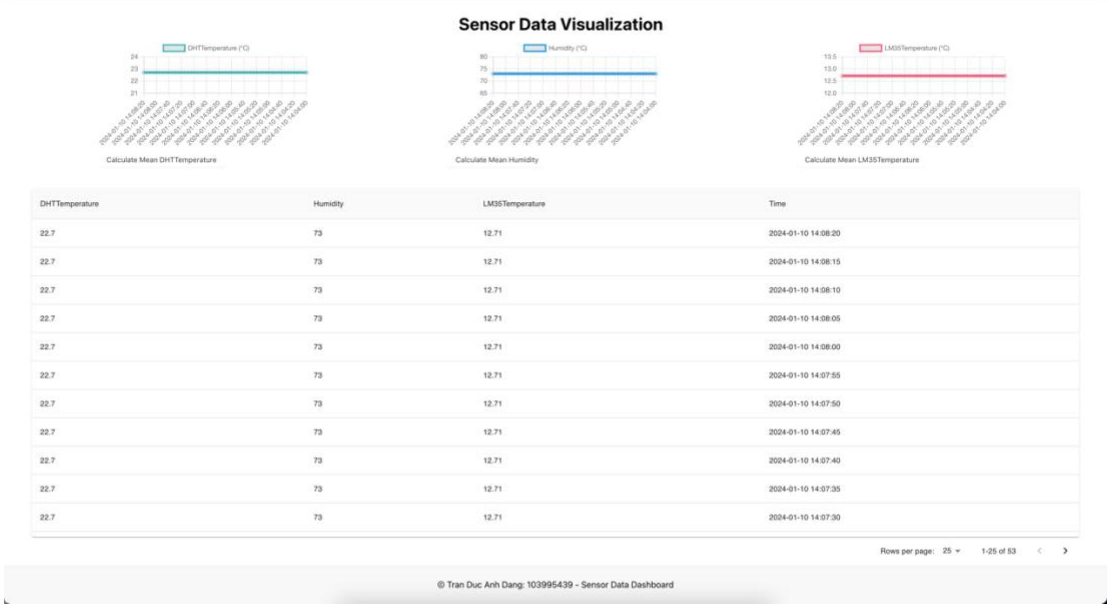


	id	DHTTemperature	Humidity	LM35Temperature	Time
Edit Copy Delete	1	22.7	73	12.71	2024-01-10 14:04:00
Edit Copy Delete	2	22.7	73	12.71	2024-01-10 14:04:05
Edit Copy Delete	3	22.7	73	12.71	2024-01-10 14:04:10
Edit Copy Delete	4	22.7	73	12.71	2024-01-10 14:04:15
Edit Copy Delete	5	22.7	73	12.71	2024-01-10 14:04:20
Edit Copy Delete	6	22.7	73	12.71	2024-01-10 14:04:25
Edit Copy Delete	7	22.7	73	12.71	2024-01-10 14:04:30
Edit Copy Delete	8	22.7	73	12.71	2024-01-10 14:04:35
Edit Copy Delete	9	22.7	73	12.71	2024-01-10 14:04:40
Edit Copy Delete	10	22.7	73	12.71	2024-01-10 14:04:45
Edit Copy Delete	11	22.7	73	12.71	2024-01-10 14:04:50
Edit Copy Delete	12	22.7	73	12.71	2024-01-10 14:04:55
Edit Copy Delete	13	22.7	73	12.71	2024-01-10 14:05:00
Edit Copy Delete	14	22.7	73	12.71	2024-01-10 14:05:05
Edit Copy Delete	15	22.7	73	12.71	2024-01-10 14:05:10
Edit Copy Delete	16	22.7	73	12.71	2024-01-10 14:05:15
Edit Copy Delete	17	22.7	73	12.71	2024-01-10 14:05:20
Edit Copy Delete	18	22.7	73	12.71	2024-01-10 14:05:25
Edit Copy Delete	19	22.7	73	12.71	2024-01-10 14:05:30
Edit Copy Delete	20	22.7	73	12.71	2024-01-10 14:05:35
Edit Copy Delete	21	22.7	73	12.71	2024-01-10 14:05:40

Figure 4 - Database

To guarantee the consistency and integrity of the data collected from the environmental sensors, the data preparation procedure was carefully planned. The pictures show that the Python and Flask-designed system backend was able to connect to the MySQL database and construct a table specifically for storing the sensor data. The temperature and humidity readings from the DHT11 and LM35 sensors were among the data gathered. As can be seen from the database images, every entry in the database was timestamped to provide the environmental data a historical context. This careful documentation produced a solid dataset that was available for analysis and visualisation and was essential in tracking environmental changes over time.

Analysis:



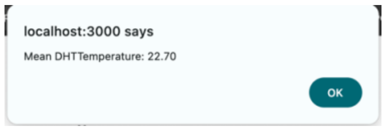
Sensor Data Visualization
DHTTemperature (°C)
Humidity (°C)
LM35Temperature (°C)
Calculate Mean DHTTemperature
Calculate Mean Humidity
Calculate Mean LM35Temperature

DHTTemperature	Humidity	LM35Temperature	Time
22.7	73	12.71	2024-01-10 14:08:20
22.7	73	12.71	2024-01-10 14:08:15
22.7	73	12.71	2024-01-10 14:08:10
22.7	73	12.71	2024-01-10 14:08:05
22.7	73	12.71	2024-01-10 14:08:00
22.7	73	12.71	2024-01-10 14:07:55
22.7	73	12.71	2024-01-10 14:07:50
22.7	73	12.71	2024-01-10 14:07:45
22.7	73	12.71	2024-01-10 14:07:40
22.7	73	12.71	2024-01-10 14:07:35
22.7	73	12.71	2024-01-10 14:07:30

Rows per page: 25 ▾ 1-25 of 53 ‹ ›

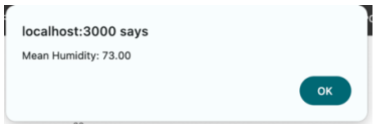
© Tran Duc Anh Dang: 103995439 - Sensor Data Dashboard

Figure 5 - Sensor Dashboard


localhost:3000 says

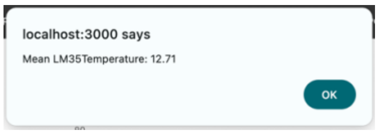
Mean DHTTemperature: 22.70

Figure 6- DHT Temperature Calculation


localhost:3000 says

Mean Humidity: 73.00

Figure 7 - Humidity Calculation


localhost:3000 says

Mean LM35Temperature: 12.71

Figure 8 - LM35Temperature Calculation

Both frontend graphical representation and backend data aggregation were used in the multifaceted examination of the prepared data. The data was sorted and filtered by the backend, which was essential in making the data accessible via a clear API endpoint. Frontend images demonstrate how the data is dynamically retrieved and presented in a dashboard manner that is easy to use, complete with real-time tables and charts. The user interface enabled the React application to parse the input and perform mean calculations. The dashboard's charts, which could compute average values throughout the show data range, gave users an instant visual assessment of the environmental trends.

Users were able to examine data points right down to the precise second of recording thanks to the table visualisation, which provided a paginated, comprehensive picture of each sensor readout. The granularity and digestibility of the analysis were guaranteed by the mix of tabular data and graphical charts, which catered to users who need both fast overviews and in-depth investigation of the sensor readings.

The screenshots above demonstrate how the data preparation and analysis steps were successfully implemented, demonstrating the system's capacity to efficiently collect, store, and show data in an understandable and instructive way. This feature, which gives end users a thorough tool for environmental monitoring and analysis, is essential to the system's function.

Discussion

Interpretation of Results:

The data captured and analyzed by the system provides a detailed account of environmental conditions over time. Interpretation of the sensor data reveals not only the current state of the environment but also allows for the observation of trends and patterns. The real-time monitoring and historical data analysis can facilitate predictive insights, enabling pre-emptive actions in response to environmental changes.

Implication:

This system has a wide range of effects. Improved interior climate management might lead to improved comfort and health for residential users. It can help with crop management and irrigation choices in agricultural applications. It could result in increased worker safety and process optimisation in industrial settings. Because of the system's flexibility and adaptability, it may be used in a variety of industries where environmental conditions are crucial.

Limitation:

While the system has demonstrated reliability, there are limitations to consider. The accuracy of the sensors determines the precision of the data collected, and sensor degradation over time could impact performance. Additionally, the current configuration is limited by the range of the sensors and the microcontroller's processing power, potentially restricting the scale of deployment.

Conclusion

This project's completion marks a big leap in the domain of IoT systems, notably in the environmental monitoring sector. An Internet of Things (IoT) node containing a variety of sensors and actuators has been developed and deployed successfully. The system's capacity to gather data in real-time and to take action in response to certain environmental circumstances says a lot about the technological expertise of the project.

The establishment of reliable serial communication between the Arduino and the Raspberry Pi stands as a testament to the system's robust data handling and transfer capabilities. The integration of a MySQL database on the Raspberry Pi for storing raw data further underscores the system's capacity for data management and scalability.

Furthermore, the system's intelligence as well as its potential for automation and real-time reaction to environmental events have been demonstrated via the deployment of edge analytics using straightforward conditional rules on the Raspberry Pi using Python. The real-world implementation of these rules like turning on a fan when the temperature rises above a predetermined point, highlights how the system can develop from a monitoring instrument to an automated response system.

A key element of system interaction, the user interface was created with efficiency and simplicity in mind. It offers a complete and user focused experience by enabling users to see current data, modify conditional rules, and analyse acquired data.

With these achievements, the project has not only met but surpassed the stated learning goals, showcasing an integrated and useful IoT system. In the future, there is room for improvement and growth of the system. Future improvements might involve adding more kinds of sensors for more comprehensive environmental monitoring, creating more sophisticated edge analytics for predictive insights, and improving the user interface for an even more user friendly experience.

In conclusion, the project has created a strong framework for IoT applications in environmental monitoring going forward, offering a model for theoretical and applied developments in this rapidly advancing sector of technology.

References

Cited sources:

1. Circuit Design App for Makers- circuito.io (no date) Circuito.io. Available at: https://www.circuito.io/ (Accessed: January 10, 2024).
2. How Real-Time Environmental Monitoring Systems are Improving our Relationship with the Planet (no date) Heavy.ai. Available at: https://www.heavy.ai/blog/how-real-time-environmental-monitoring-systems-are-improving-our-relationship-with-the-planet (Accessed: January 10, 2024).
3. Gonzalez, V. *et al.* (2022) “Real-time environmental monitoring platform for wellness and preventive care in a smart and sustainable city with an

urban landscape perspective: The case of developing countries," *Land*, 11(10), p. 1635. doi: 10.3390/land11101635.

4. Last Minute Engineers (2021) *Interfacing LM35 temperature sensor with arduino*, *Last Minute Engineers*. Available at: https://lastminuteengineers.com/lm35-temperature-sensor-arduino-tutorial/ (Accessed: January 10, 2024).
5. Zafar, S. *et al.* (no date) *An IoT based real-time environmental monitoring system using arduino and cloud service*, *Semanticscholar.org*. Available at: https://pdfs.semanticscholar.org/ccef/9364d35331adb759fc476cf25f33c842a0df.pdf (Accessed: January 10, 2024).
6. Kavitha, A. K. and Marypraveena, S. (no date) *Design and implementation of Weather Monitoring System using wireless communication*, *Ijaiet.com*. Available at: https://www.ijaiet.com/wp-content/uploads/2018/02/Design-and-Implementation-of-Weather-Monitoring-System-using-Wireless-Communication.pdf (Accessed: January 10, 2024).
7. *Arduino IoT bundle* (no date) *Arduino Online Shop*. Available at: https://store-usa.arduino.cc/collections/kits/products/iot-bundle (Accessed: January 10, 2024).