Weather Monitoring System

Mufrad Mahmud, Abu Taher, Md. Noman, Tanmoy Chowdhury

1 Introduction

The Weather Monitor is an initiative to design and develop an intelligent and energy-efficient system that constantly measures parameters like temperature, humidity, pressure, etc. With the help of Raspberry Pi Pico W capabilities, special sensors, and cloud services integration, the system will help the users make the right decisions in terms of **HVAC** control. It also processes real-time data using a machine-learning model to detect and respond to anomalies.

The purpose here is to provide a more focused and user-oriented solution for the socio-technical system that will include both physically stable and ergonomically friendly hardware and software components for constant and efficient indoor environment control and maintenance. The Weather Monitor will include a real-time data graphical interface, data history analysis, and control mechanisms that will help the users conserve energy and regulate the environment to their comfort.

To achieve the objectives, the project implements the following core functionalities:

1. **Sensor Data Collection:**

   - The system collects environmental data from two sensors:
     - *BMP280:* Records temperature and pressure values.
     - *DHT22:* Records temperature and humidity.

     These sensors are connected to the Raspberry Pi Pico W, a low-cost, Wi-Fi-enabled microcontroller to allow the transfer of data via wireless communications. The data collection is done at 2-second intervals, at which continuous data is recorded from the subjects, and the data collected is formatted in CSV format on personal computers for further analysis.

     We collected temperature, humidity, and atmospheric pressure data. At first, we collected the data in a normal room environment and named it normal data. Then we collected abnormal data by blowing warm air into the sensor and named it abnormal data.

2. **Data Processing:**

   - Edge Processing on the Raspberry Pi Pico W:

  - The gathered sensor data is processed on Colab.
  - Averages or other thresholds are calculated to decide trends within numbers or big values as well as detect anomalies.
  - Processed data is in CSV format for subsequent structured data processing.
  - With regard to machine learning, the anomalies can be predicted using historical weather data and used to improve system accuracy.

3. **Output Control:**

   - The processed data drives automated actions and user-facing outputs:
     - *Output Devices:* Information about the monitored situation is provided in Grafan. Grafan will send an email immediately if there is any anomaly detection by the model.
     - *Event-Based Adjustments:* Users can set completely new events to change the temperature of **HVAC** systems during certain periods to suit their needs, as well as reduce energy consumption.
     - *Manual Control:* Users will be able to change or overrule the decision and set heating or cooling in case of an undesired instance.

Further, the project MUST implement AT LEAST TWO of the functionalities below:

1. **Wireless Connectivity:**

   - Wireless communication with MQTT or HTTP should be implemented. Wi-Fi and Bluetooth can be for wireless connectivity. Pico W has an onboard Wi-Fi module available. Bluetooth requires an external module.

2. **Cloud integration:**

   - Integrate with cloud platforms such as InfuxDB, MangoDB, AWS IoT, or Google

3. **Security:**

   - Incorporate data encryption

4. **Edge Machine Learning:**

   - Implement edge-based machine learning models on the Pico W.

In this section, describe your application and which of the above functionalities it implements. Provide the overall view of the application area, zooming in on the particular application you've implemented.

2 Background Study

The study by Sharma and Dutta argues how wireless networking, AI and sensor technologies and IoT is disrupting the process of weather monitoring. [1] Taken together, these technologies enable improved computational experiences and what can be described as real-time data acquisition. In this paper, four different cases are presented in order to support the statement that IoT and wireless networking are merging into a new paradigm of transformative computing. The study emphasizes shifts towards using technologies in gathering and utilizing weather data and shows how these technologies can revolutionize weather monitoring systems.

The aim of this paper is to explore the approaches to estimate the room occupancy from environmental data using machine learning algorithms. The abstract reveals a methodology for the creation of a polyharmonic microwave photonic vector analyzer as a sensing tool for increasing the efficiency of the sensor-based data collection in complex environments. [2] This approach is useful in almost all smart building applications that require energy efficient control of building utilities such as HVAC system where room occupancy has to be predicted in order to regulate energy use.

In the paper by Ahmed et al. the author focuses on the use of IoT and machine learning for efficient consumption of energy in HVAC systems. [3] Their application and advantages in high-power traction and distribution networks are discussed, along with proposing a new MIMC topology capable of providing variable-frequency operation. While the abstract focuses on power converters, Madrid notes that applying the research to IoT for energy applications is vast, especially in HVAC systems with limited spaces. Experimental outcomes support this approach, while the paper highlights AI as a tool for enhancing control and effectiveness of a system.

The work done by Chowdhury and Lee specifically targets the cloud-IBC IoT for weather monitoring particularly traffic flows and SFC. [4] It also looks at how the service functions, enhanced by the SDN and NFV can be implemented in a multi-domain optical network. Although the abstract is articulated with specifying the networking and traffic management in the networking areas, the main idea is fitting with any IoT-based WEATHER SYSTEM that requires dependable data transportation in large scales. This work underlines that path provisioning and data management in the cloud-based IoT arrangements inherent to weather monitoring are crucial.

The application of IoT and AI is revolutionizing the HVAC industry in a dramatic way. First of all, IoT introduced a simple option to remotely control the HVAC systems using the smartphone APPLICATIONS, which improved the comfort. Nonetheless, development in the field of AI and big data has brought significant changes in system technology that allow the system to collect and

analyze data non-stop. Automatic systems remove the requirement for immediate user involvement by figuring out the user's desires, offer smart and specific enhancements, and greatly enhance energy effectiveness and conservation. [5]

In this paper, the implications of IoT, WSN, and ML for weather forecasting and monitoring in real-time was examined. [6] The system comprises temperature, humidity, pressure, wind speed sensors placed at various locations to obtain meteorological data. ML everyday and historical data to analyze and forecast short-term weather data in the environment. These predictive capabilities also allow for the immediate dissemination of danger signals for specific severe weather occurrences to such sectors as farming, transport, and calamity response. The system architecture is based on an Internet of Things, so it does not involve cumbersome processes of accumulation and explores meteorological data, including and offering web and mobile applications' access for users.

Weather monitoring is necessary for agriculture and cities to predict the climate change and provide suitable measures for their inhabitants. The modern solution can be offered through the use of sensors and IoT in the case of the weather monitoring system. As a result, the utilisation of this system means that decision-making is much more credible – especially when it comes to something like agricultural yields or the effective usage of a city's resources. [7] It also describes how to create a regime of such a system, which also underscores its practicality and possibilities of creating the interconnected devices with improved workflows.

Various outcomes demonstrate that the integration of IoT to HVAC systems is revolutionizing how controlled indoor environments are. HVAC systems that incorporate IoT are able to provide more comfort, better quality of the air that is being circulated inside a building as well as very many more advantages such as energy and cost savings. [8] These systems can adopt smart solutions to guard and get the right settings in an on-going fashion for that most proficient performance. This paper focuses on benefits of such smart HVAC systems and outlines technological innovations that foster this change.

3 Architecture

The Weather Monitor application is structured into three layers: Based on the above classification, the **LBS** has three key layers, including the sensor layer, the networking layer, and the data management layer. All of them are significant in order to guarantee the proper work, productivity, and orientation toward the user of the system. Figure 1 depicts the architecture of an AI-powered weather monitoring system.

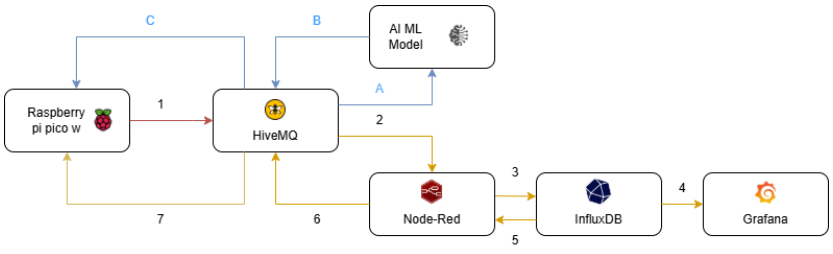

Figure 1: Architecture of AI-powered weather monitoring system

- **Sensor Layer:** This layer gathers the environmental information and constitutes the basic elements of the hardware of the system.
  - *Components:*
    - BMP280 Sensor: Records temperature and atmospheric pressures intuitively and independently.
    - DHT22 Sensor: It is used for measuring temperature as well as humidity.
  - *Microcontroller:*
    - Raspberry Pi Pico W: Co-ordinates interaction with the sensors, keeps its low power consumption, and provides wireless connectivity.
  - *Functionality:*
    - Data was captured at 2-second intervals using sensors.
    - Pre-processing of data and model training was done in the Colab environment.
- **Networking Layer:** This layer allows an efficient communication channel of interaction between the various sensors, the central system, and the user interface.
  - *Connectivity:*
    - With the Raspberry Pi Pico W, it is possible to have a wireless connection to local networks through the integrated Wi-Fi module.
  - *Protocols:*
    - For messaging with Hivemq, we used the MQTT protocol. For encryption, we used the SSL/TLS protocol.
- **MQTT Broker:** Pico W, Node-Red, and Colab communicate with each other using HiveMQ as a broker.
  - *Real-Time Communication:*

    - Two sensors send data through a Raspberry Pi to the HiveMQ, which acts as a broker, sending the accumulated data to every part of the main application of the system.
    - The parameters as well as the messages are exchanged in the JSON format to guarantee synergy and easy processing.

- **Data Management Layer:** This layer is mainly focused on data storage, analysis, and output.
  - *Data Storage:*
    - InfluxDB is a time series database used to store all unprocessed and processed data. the focuses on managing time-stamped data.
    - This data platform empowers experiments to securely store real-time and historical data to be queried concurrently.
  - *Data Analysis:*
    - InfluxDB stores the historical data for time series, where the trending and variation analysis is conducted.
    - These data are used to train a machine learning model for anomaly detection depending on the weather information.
  - *Outputs:*
    - The output of the influxDB is sent to the Grafan, which is set up in the local machine. In Grafana, we created a dashboard to visualize the incoming data. We also set up an email alert, which will be activated if there is any anomaly.

- **ML Model Deployment:** In this project, we used the ML model for the classification of incoming data. We used the Random Forest model for this purpose.
  - *Real-Time classification:*
    - We trained the model with normal and abnormal data. In the Colab environment, we retrieved real-time data from hivemq and classified the input data as 'normal' and 'abnormal'.
    - We sent the classification report to HiveMQ and from HiveMQ the message was forwarded to the Raspberry Pi Pico, where it turned 'ON' and 'OFF' the onboard LED, respectively for 'normal' and 'abnormal' messages.

4 Methods & Tools

The following figure shows the circuit diagram of the dht22 and bmp280 sensors. Figure 2 depicts our system's circuit diagram.

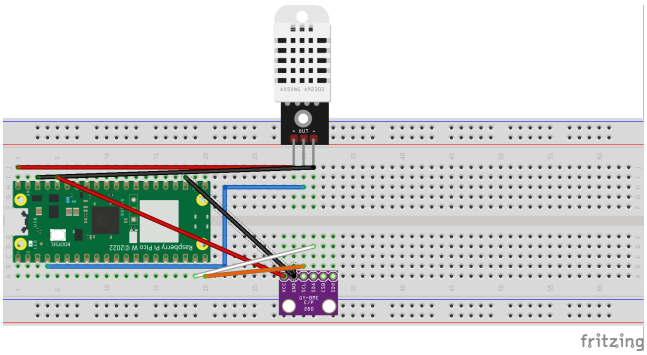

Figure 2: Circuit diagram of the DHT22 and BMP280 sensors.

DHT22 Pin	**Raspberry Pi Pico W Pin**
VCC (+)	5V (Pin 40)
Data	GP2 (Pin 4)
GND (-)	Pin 38

Table 1: Pin configuration between DHT22 and Raspberry Pi Pico W.

BMP 280	**Raspberry Pi Pico W**
VCC	PIN 36
GND	PIN 23
SDA	GP 14 (PIN 19)
SLA	GP 15 (PIN 20)

Table 2: Example of a 5x2 table in Overleaf.

- **Sensor Integration**
  - *Hardware:*
    - BMP280: It is being used in taking measurements of both temperature and pressure.
    - DHT22: It has been employed in ending temperature and humidity.
  - *Programming Language:*
    - MicroPn was employed as the programming language of the Raspberry Pi Pico W.
  - *Integration Steps:*
    - All the sensors were interfaced with Raspberry Pi Pico W through GPIO pins. We use the I2C communication protocol for both sensors.

    - Lastly, libraries like machine and BMP280 were used to communicate with the BMP280 sensor.
    - The DHT22 sensor was connected via the built-in MicroPython dht module in the official Raspberry Pi Pico W package manager.

- **Networking and Communication**
  - *Wireless Communication:*
    - The data is sent to the application via the built-in Wi-Fi of the Raspberry Pi Pico W. Figure 2 shows successful data publication in HiveMQ.
  - *Protocols:*
    - For messaging with Hivemq, we used the MQTT protocol. For encryption, we used the SSL/TLS protocol.
  - *Libraries:*
    - A MicroPython library called 'Simple' was used to facilitate the MQTT interactions.

Figure 3 depicts successful data publication in HiveMQ.

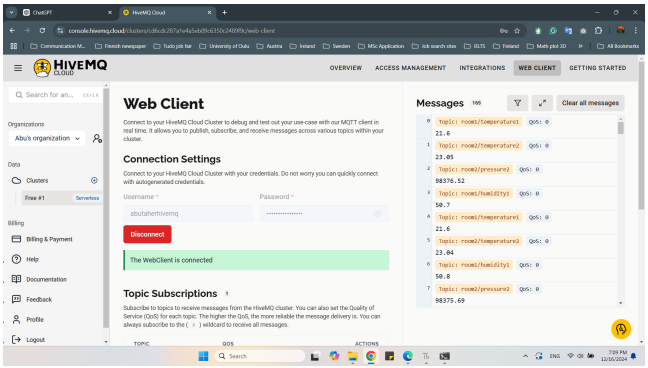

Figure 3: Successful data publication in HiveMQ

- **Data Storage and Management**
  - *Local Data Storage:*
    - Information received by the sensors was sent to the HiveMQ as topics. We used four topics 'room1/temperature1', 'room1/humidity1', 'room2/temperature2', and 'room2/pressure2'. Although in topic names we used room1 and room2 naming, we used it as data collected from a single room.
  - *Cloud Database:*

* We simulated this operation by sending data and storing it in InfluxDB, which stores time series data.
* We used Node-red to collect data from HiveMQ and then store it in InfluxDB.

Figure 4 and 5 represents our data retrieval process from HiveMQ using Node-Red.

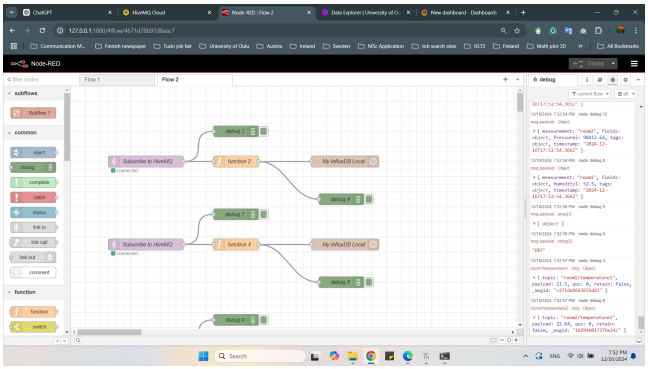

Figure 4: Data retrieval from HiveMQ using Node-Red

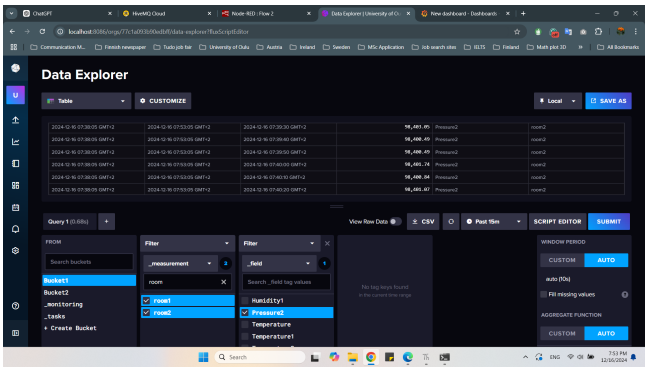

Figure 5: Data retrieval from HiveMQ using Node-Red

– *Visualization:*

* For visualization we used Grafana, which collects real-time data from InfluxDb and periodically updates the dashboard.

Figure 6 depicts our grafana dashboard which was integrated with influxDB for data visualization.

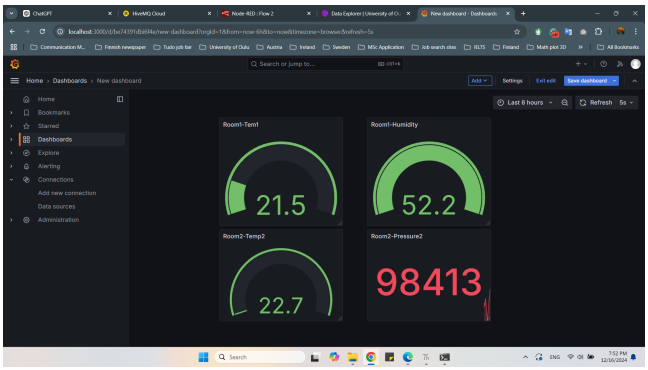

Figure 6: Grafana dashboard integration with InfluxDB

- **Data Analysis and Machine Learning**
  - *Exploratory Data Analysis (EDA):*
    - The data was extracted, reviewed, and analyzed from the Core Python libraries Pandas and NumPy.
  - *Machine Learning Model:*
    - A random forest model was trained using weather parameters such as temperature, humidity, and pressure to predict anomalies. Utilities like the Sci-kit Learn were used in the assessment.
    - Dataset: Our manual collected weather data acted as the dataset for model training, which is done with training and test data. There were 43 normal data points and 52 abnormal data points or rows. Each row has four columns [temp1, temp2, hum1, press2]. In total, we had 95 rows of data. We divided this dataset into 80-20 percent for training and testing, respectively.
      Figure 7, 8, and 9 verfies about the availability of our dataset.

```
print("X_train:", X_train.shape)
print("y_train:", y_train.shape)
print("X_test:", X_test.shape)
print("y_test:", y_test.shape)

X_train.head()
```

```
X_train: (75, 4)
y_train: (75,)
X_test: (20, 4)
y_test: (20,)
```

	temp1	temp2	hum1	press2
3	21.7625	22.862500	48.575000	98424.2700
6	21.8000	22.808571	48.371429	98423.5100
27	21.7500	22.562500	47.900000	98426.7575
31	21.7375	22.517500	47.625000	98428.2675
19	21.8000	22.658750	48.000000	98423.6225

Figure 7: Dataset

* Model: The architecture of the model and hyperparameters used in the model are described below.

Hyperparameter	**Value**
bootstrap	True
ccp_alpha	0.0
class_weight	balanced
criterion	gini
max_depth	sqrt
max_features	5
max_leaf_nodes	-
max_samples	-
min_impurity_decrease	0.0
min_samples_leaf	1
min_samples_split	2
min_weight_fraction_leaf	0.0
monotonic_cst	-
n_estimators	100
n_jobs	42
oob_score	False
random_state	0
verbose	0
warm_start	False

Table 3: Hyperparameters of the Random Forest model.

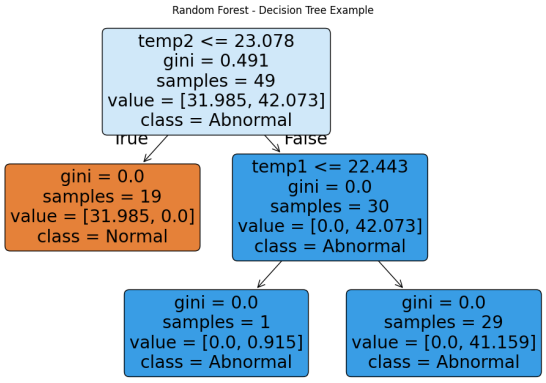

Figure 8: Dataset

```
print("X_train:", X_train.shape)
print("y_train:", y_train.shape)
print("X_test:", X_test.shape)
print("y_test:", y_test.shape)

X_train.head()
```

```
X_train: (75, 4)
y_train: (75,)
X_test: (20, 4)
y_test: (20,)
```

	temp1	temp2	hum1	press2
3	21.7625	22.862500	48.575000	98424.2700
6	21.8000	22.808571	48.371429	98423.5100
27	21.7500	22.562500	47.900000	98426.7575
31	21.7375	22.517500	47.625000	98428.2675
19	21.8000	22.658750	48.000000	98423.6225

Figure 9: Dataset

- **Application Development**
  - *App Framework:*
    - For the desktop visualization, the app has been created using Tkinter, a GUI library for Python.
  - *Functionality:*
    - Shows current trends of data and even predicts some trends.
    - Can control the temperature via a touchtone phone.
  - *Edge Processing:*
    - Initially, the Raspberry Pi Pico W performed data preprocessing with the aim of minimizing delays before passing data to the application. We couldn't compile the model in C/C++ format so we used the model in Colab the process and generate real-time data. We sent an anomaly detection message to the HiveMQ and then received the message in Pico W. Using the message, we turned the onboard LED ON and OFF.
- **Testing and Debugging**
  - *Unit Testing:*
    - Various elements, such as the sensor reading transfers and the integration of the database, were individually tested.
  - *End-to-End Testing:*

    - Enrolled a part of the sensors to ensure that every component in the system was working effectively right from the data collection to the app visualization.
- *Tools:*
    - Thonny IDE with which to build and debug MicroPython programs.
    - Postman when performing testing of REST API interactions.

5 Result Analysis

We successfully implemented the data processing and visualization pipeline and demonstrated that our project works. For data processing and visualization, we used InfluxDB and Grafana. In Grafana we successfully implemented email alerts.

We trained the Random Forest model using raw data collected using sensors. We achieved 100% accuracy on the test dataset. The confusion matrix of the model is given below.

Figure 10 reflects the confusion matrix that was acquired.

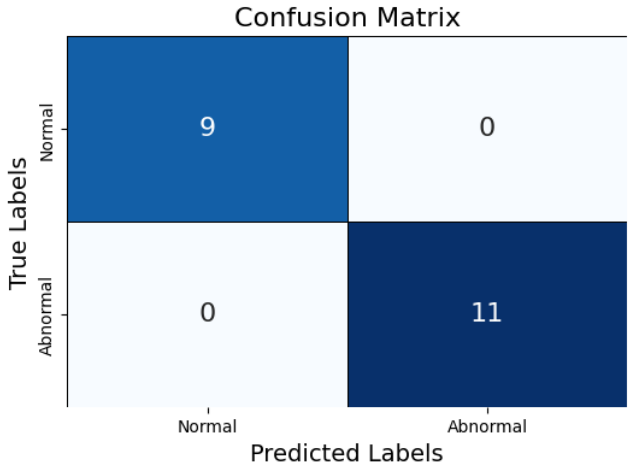

Figure 10: Dataset

We made predictions on real-time data in Colab and sent the prediction to the pico W.

6 Discussion

All requirements of the project were satisfied. while we implemented all the parts successfully, we encountered some issues. The most important issue was delay.

When we sent a prediction message to the pico we noticed that it sometimes stopped operating. It can be because of our string message value.

7 Conclusion

In this project, we integrated all the parts correctly to satisfy the requirements.

We set up a Wi-Fi connection and sent data over the wireless connection to the broker. In order to protect the data we use an SSL/TLS certificate to ensure encryption and security. We used the SSL/TLS protocol to facilitate communication between HiveMQ to Node-Red and HiveMQ to Colab.

We integrated InfluxDB to store the time series data and Grafana to visualize incoming data.

We implemented a random forest machine learning model to make predictions about real-time data.

References

[1] Sharma, R., & Dutta, D. (2019). IoT-Based Weather Monitoring System: A Comprehensive Review. *IEEE Access*. Retrieved from `https://ieeexplore.ieee.org/document/8747277`

[2] Li, J., Yang, Y., & Zhang, H. (2021). Machine Learning Applications for Predicting Room Occupancy Using Environmental Data. *IEEE Sensors Journal*. Retrieved from `https://ieeexplore.ieee.org/document/9415994`

[3] Ahmed, K., Wang, X., & Khan, A. (2019). Energy-Efficient HVAC Control with IoT and Machine Learning Integration. *IEEE Transactions on Sustainable Energy*. Retrieved from `https://ieeexplore.ieee.org/document/8684890`

[4] Chowdhury, A., & Lee, S. (2020). Cloud-Based IoT Weather Monitoring Systems: Implementation and Analysis. *IEEE Internet of Things Journal*. Retrieved from `https://ieeexplore.ieee.org/document/9189853`

[5] Hitachi Air Conditioning. (n.d.). Energy transformation in HVAC: IoT, AI, and HVAC efficiency. Retrieved from `https://www.hitachiaircon.com/uk/magazine/energy-transformation-in-hvac-iot-ai-and-hvac-efficiency`

[6] International Journal on Recent and Innovation Trends in Computing and Communication. (n.d.). IoT-based machine learning weather monitoring and prediction system. Retrieved from `https://ijritcc.org/index.php/ijritcc/article/view/7990`

[7] Vayuyaan. (n.d.). Weather monitoring system using IoT. Retrieved from `https://vayuyaan.com/`

blog/weather-monitoring-system-using-iot/?srsltid=
AfmBOopscfOWJgOROROz4sOkZeStGofNc8BjnADv8AlIIEOA-k7qnfHS

[8] Kaa IoT. (n.d.). IoT in HVAC system: Benefits with smart solutions. Retrieved from https://www.kaaiot.com/iot-knowledge-base/iot-hvac-system-benefits-and-smart-solutions