

# IoT-Based Real-Time Weather Monitoring And Prediction System

## A web application

Kaveesha lakdinu, Hasitha harshana , Safir Jameel Manghat  
Group 3-4

Department of Computer  
and Systems Sciences

Degree project XX HE credits

Computer and Systems Sciences

Degree project at the bachelor/master level

Autumn/Spring term 20XX

Supervisor: Name

Swedish title: XXX – not mandatory



# Abstract

In this project, a Raspberry Pi, a temperature sensor (DHT22), and Tell Stick for wireless data transfer are used to create a basic IoT-Based Real-Time Weather Monitoring System. The main goal is to develop a productive system that gathers temperature data in real time and presents it via an intuitive online interface. In order to improve functioning by projecting future weather conditions, the project also incorporates a weather prediction system. Accurate short-term weather forecasts are produced using machine learning models that have been trained on Kaggle datasets. For a cohesive user experience, the weather forecast system is also smoothly integrated with the dashboard.

Planning and setup, coding, integration, documentation, testing, debugging, and optional enhancements are all part of the project's organized methodology. The Raspberry Pi's backend web server application is developed using the Flask framework, which results in an easy-to-use dashboard that allows users to view current temperature data and weather forecasts.

The precision of temperature sensor readings, Tell Stick's data transfer dependability, and the dashboard's integration with the weather forecast system are the main areas of testing. Weeks 1-3 are dedicated to setup and planning; Weeks 4–5 are dedicated to software development and documentation; and Week 5 is dedicated to testing and troubleshooting.

The project deliverables include an extensive report, well documented source code for temperature sensor readings, dashboard, cloud platform integration, Wi-Fi connectivity, and an integrated weather forecast system. Together with precise short-term weather predictions and real-time temperature data, the web interface/dashboard is a stand-alone delivery.

*Keywords:* IoT, Temperature Sensor, Raspberry Pi, Tell Stick, Data Transmission, Flask, Dashboard, Real-Time, Internet of Things, Connectivity, Environmental Monitoring, Weather Prediction, Kaggle Dataset, Machine Learning.

# Synopsis

## Background

Accessible and precise weather monitoring systems are becoming more and more necessary as the demand for real-time weather information rises. Conventional weather monitoring techniques could not be as precise and immediate as needed for quick decision-making. The creation of an economical and effective solution is made possible by the integration of Internet of Things devices, such as temperature sensors and the Raspberry Pi microcontroller. The goal of this project is to use these technologies to develop a dependable and approachable weather monitoring system.

## Problem

The current weather monitoring systems frequently struggle with issues of affordability, usability, and quality of real-time data. Conventional systems could not be widely adopted since they are costly to set up and maintain. Furthermore, the accuracy of weather forecasts may be impacted by the delay in data retrieval and processing. By offering an accessible and reasonably priced Internet of Things (IoT)-based system that provides accurate short-term weather forecasts and real-time temperature data, this project aims to address these problems.

## Research Question

This project's main research topic is: How can an Internet of Things-based real-time weather monitoring system be developed and put into use to give customers precise, real-time temperature data as well as trustworthy, short-term weather forecasts? The technical factors of integrating hardware and software, the efficiency of the selected sensors, and the precision of machine learning models in forecasting weather patterns from previous data are all included in this question.

## Method

The approach entails putting in place a hardware configuration with a Tell Stick for wireless data transfer, a Raspberry Pi microprocessor, and a DHT22 temperature sensor. Programming for sensor readings, Wi-Fi connectivity, cloud platform integration, and machine learning model incorporation for weather prediction are all included in software development. The project has a set schedule that includes testing, planning, coding, and optional improvements.

Testing protocols evaluate the integrated weather forecast system's efficacy, the precision of temperature sensor readings, and the dependability of data transfer.

## Result

A fully working IoT-based weather monitoring system with a responsive dashboard that offers current temperature information and short-term weather forecasts is what the project seeks to accomplish. The outcome consists of an integrated system that effectively integrates sensor data with the cloud platform to provide accurate weather forecasts, as well as an extensive project report and well-documented source code.

## Discussion

The project results, difficulties faced, and possible areas for improvement are examined during the discussion phase. Critical evaluations are conducted on the efficiency of the software implementation, hardware configuration, and machine learning model integration. Future development and scalability considerations are covered, as well as the system's possible influence on giving consumers accurate and easily available weather information. The goal of the conversation is to shed light on the project's accomplishments as well as any areas that could need more improvement.

# Acknowledgement

This IoT-Based Real-Time Weather Monitoring and Prediction System was completed successfully thanks to the cooperation and assistance of several people and organizations. We want to express our sincere appreciation to everyone who helped make this initiative a reality. We would especially want to thank our project manager, Fahim Shahariar Nahin, and our instructor, Rahim Rahmani Chianeh, whose advice and knowledge were very helpful during the project. Their encouraging words and helpful criticism were crucial to the project's accomplishment. We express our gratitude to Stockholm University for their technical help, which enabled us to overcome obstacles pertaining to hardware configuration, software development, and testing. The project's objectives have been greatly aided by the combined efforts of these people and organizations. We appreciate your crucial contribution to our accomplishments.

# Table of Contents

<b>Abstract .....</b>	<b>ii</b>
<b>Synopsis.....</b>	<b>iii</b>
<b>List of figures .....</b>	<b>ii</b>
<b>List of Tables .....</b>	<b>ii</b>
Introduction.....	1
Problem Statement.....	2
Research question .....	2
<b>1 Second Chapter .....</b>	<b>3</b>
2.1 Evolution of Weather Monitoring Systems.....	3
2.2 Technologies in Contemporary Weather Monitoring .....	4
<b>2 Third Chapter .....</b>	<b>7</b>
<b>Methodology .....</b>	<b>7</b>
3.1 Research Strategies .....	7
3.1.1 Hardware Configuration .....	7
3.1.2 Software Development.....	7
3.1.3 Data collection .....	7
3.1.4 Qualitative analysis method.....	8
3.1.5 Quantitative analysis method.....	8
<b>3 Fourth Chapter .....</b>	<b>9</b>
4.1 Sensor value monitoring result .....	9
4.2 Weather Prediction Result .....	11
.....	11
.....	11
.....	12
4.3 Weather Prediction Testing.....	12
<b>4 Fifth Chapter .....</b>	<b>13</b>
5.1 Weather Prediction .....	13
Weather Prediction model.....	14
Flask web framework with API.....	16
5.1.1 Comparison with Previous Work .....	18
5.2 Sensor data reading .....	19
5.3 Dashboard Integration.....	22
<b>References .....</b>	<b>24</b>

# List of figures

Figure 1 The process of our project .....	2
Figure 2 Research Strategies and Research Methods.....	<b>Error! Bookmark not defined.</b>
Figure 3 sensor data .....	9
Figure 4 confirmation of sensor data.....	10
Figure 5 select only relevent columns.....	14
Figure 6 Statistical Testing .....	14
Figure 7 correlation analyze.....	15
Figure 8 Import my model.....	15
Figure 9 flask framework with API integration.....	16
Figure 10 get weather function.....	17
Figure 11 display weather function.....	18
Figure 12 app.py script.....	20
Figure 13 app.py script 2 .....	20
Figure 14 app.py script 3.....	20
Figure 15 flask web interface .....	21

# List of Tables

Table 1 sensor value result.....	9
Table 2 weather prediction result.....	12
Table 3 weather prediction testing.....	12

# List of Abbreviations

IOT – Internet of Things

ML – Machine Learning

RP2 – Raspberry PI 2

Temp -Temperature

SLP – Sea level Pressure

LCD - Liquid-crystal display

ESA - Endangered Species Act



# Introduction

The Internet of Things, or IoT, is a system of interconnected devices that communicates and exchanges data with the cloud and other IoT devices. IoT devices can be consumer goods, digital and mechanical machinery, or any combination of these, and they are often incorporated with technology such as sensors and software. (Gillis, 2023)

Our capacity to identify, decipher, and forecast weather patterns has only been useful to humans for the past 50 years. Even in the 18th and 19th centuries, the use of thermometers and barometers for weather forecasting was constrained by communication and a broad knowledge of weather patterns. (Wendt, 2023) The need for precise and up-to-date meteorological information is more than ever in this age of swift technological advancement. Conventional weather monitoring systems continue to encounter issues with accessibility, cost, and speed of data retrieval. To address these issues, our project aims to utilize Internet of Things (IoT) devices to create a complete, affordable, and user-friendly real-time weather monitoring system that includes temperature data and short-term weather forecasts.

Our method is based on the integration of IoT devices, such as the Raspberry Pi ,Tell stick and temperature sensors (such the DHT22). Our goal is twofold: providing users with accurate, real-time temperature data and addressing the shortcomings of traditional weather monitoring systems. In addition, the project aims to offer users with accurate short-term weather forecasts by including a weather prediction system that uses machine learning models trained on Kaggle datasets.

We will go into great detail about our design, execution, approach, and outcomes in the upcoming chapters. Our IoT-Based Real-Time Weather Monitoring System combines short-term weather prediction with real-time temperature monitoring in an effort to not only satisfy current customer demands but also set the stage for future weather monitoring systems that are flexible and scalable.

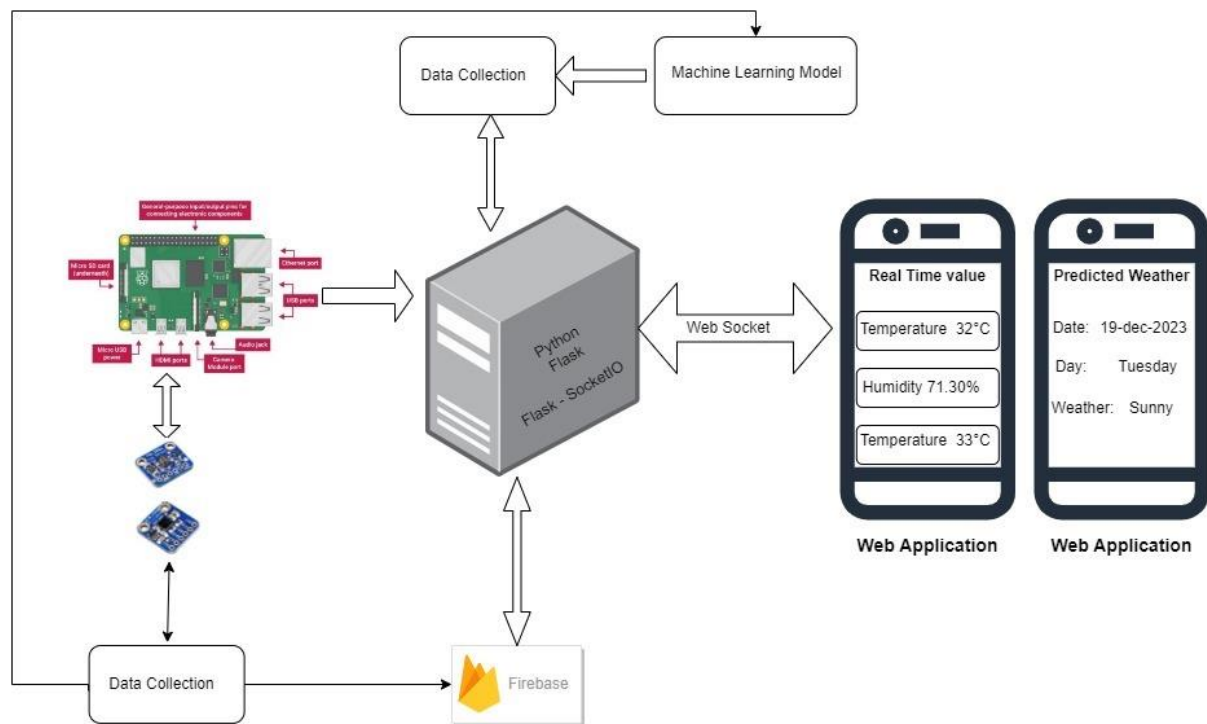


Figure 1 The process of our project

## Problem Statement

Traditional weather monitoring systems struggle with real-time data quality, implementation costs, and continuing maintenance costs. Weather prediction accuracy can be impacted by processing and data retrieval delays, which can hinder prompt decision-making. Our initiative attempts to provide an affordable and easily accessible Internet of Things solution to address these problems. This technology improves the efficacy of weather monitoring by giving users access to real-time temperature data in addition to precise short-term weather forecasts.

## Research question

We formulate our main research question as follows: How can an accurate real-time weather monitoring system powered by the Internet of Things be created and put into operation to provide users with trustworthy short-term weather forecasts and accurate real-time weather data? This investigation covers a wide range of technical factors, such as the combination of software and hardware, the effectiveness of certain sensors, and the precision of machine learning models in predicting weather patterns from past data.

# 1 Second Chapter

This chapter explores the history of weather monitoring systems in detail, including how technology and methods have evolved and the difficulties encountered in providing accurate real-time weather information. It also highlights relevant research findings that have advanced our knowledge of machine learning integration in environmental monitoring, weather prediction models, and Internet of Things-based weather monitoring systems.

## 2.1 Evolution of Weather Monitoring Systems

Weather monitoring used to be done by hand using antiquated methods that had developed over centuries. Prior to the development of advanced technology, meteorologists used straightforward devices and direct observation to record meteorological conditions. The first techniques used shadow lengths and sundials to determine the time and approximate locations of the sun, giving information about daily weather trends. In order to predict impending weather changes, sky observations, cloud forms, and wind directions were also carefully recorded. More sophisticated equipment, including barometers for measuring air pressure and thermometers for recording temperature, were added as scientific understanding grew. The 19th century saw the installation of weather stations and the telegraph, which made it easier to gather and share meteorological data between areas.

Since climate change keeps having a detrimental impact on people's quality of life, it is no longer news. Thus, keeping an eye on the weather is essential for both reducing and observing the day-to-day consequences of climate change. Temperature, humidity, and light intensity are all monitored via a weather monitoring system built on an Arduino platform. The measured parameters were input into a PC, where they were recorded as a text file accessible for later use and concurrently shown on an LCD screen.

Eventually, a temperature and humidity sensor-based weather monitoring system was created. An Arduino Uno board and a DHT 11 sensor were used in the development of the system, which tracks a room's temperature and humidity and stores the data in Microsoft Excel. Only temperature and humidity could be measured by the device. (Abdulkadir H. Alkali1, 2023)

Grasp the difficulties and advancements in modern weather monitoring requires a fundamental grasp of the historical development of weather monitoring systems. This section describes the

development of reliable and up-to-date weather data from manual observation-based classical methods to automated stations and satellite technology.

## 2.2 Technologies in Contemporary Weather Monitoring

With the introduction of new tools and methods, weather forecasting is always changing. Using machine learning, new types of sensors, crowdsourcing data, and improving models to anticipate extreme weather occurrences are some of the current developments in weather forecasting.

GPS radio occultation sensors are manufactured by PlanetiQ. This technique examines how GPS signals are distorted by the atmosphere. Additionally, the air density and other variables affect the signal's bending. Put differently, the bending of the GPS signal provides information on the state of the atmosphere at the moment and its physical characteristics.

ESA's wind mission is called Aeolus, which is also the name of the Greek wind deity from antiquity. It use wind LiDAR based on satellites. A wind LiDAR is a type of remote sensing device that measures wind direction and speed using lasers. Additionally, this technology is used by the Aeolus mission to gain a better understanding of worldwide wind patterns. An ESA project that makes use of GPS signals is called CAMALIOT. It detects the alterations in GPS signals and then applies machine learning to correlate these observations with variations in atmospheric conditions. The focus is mostly on humidity. (Wolf, 2023)

A thorough review of the technology used in contemporary weather monitoring is provided, emphasizing how Internet of Things (IoT) gadgets like sensors and microcontrollers are integrated to create more functional and approachable solutions. It is also examined how data analytics, cloud computing, and wireless connectivity might improve the functionality of weather monitoring systems.

### **2.3 Challenges in Weather Monitoring and prediction**

Predicting the future condition of the atmosphere is a difficult and complex undertaking in weather forecasting. Even while the accuracy of contemporary weather models has improved significantly, there is still a limit to how far ahead of time we can reliably predict the weather. This restriction results from the atmosphere's chaotic character, which makes it challenging to pinpoint the beginning circumstances with accuracy.

The atmosphere's chaotic character is one of the main reasons that weather forecasting is challenging. The atmosphere is a complicated system that reacts quickly to even minute modifications to its starting circumstances. The butterfly effect, which describes this sensitivity, states that minor changes made to one component of the system can have a big impact on other components. Our restricted capacity to observe the atmosphere is another element contributing to the difficulty of weather forecasting. Even with all of the instruments at our disposal—radars, weather balloons, satellites, etc.—there are still a lot of parts of the atmosphere that are not visible to the naked eye. (Meloni, 2023)

A number of difficulties arise in weather monitoring and prediction, from obtaining precise sensor readings to creating reliable forecast models. Acquiring trustworthy sensor data is an important but challenging endeavor since uncertainties can arise from the environment, sensor calibration, and even malfunctions. Once obtained, resolving data visualization concerns, guaranteeing lucidity, and upholding real-time updates are necessary for properly presenting this data on a dashboard. Another major problem is creating an accurate prediction model, which requires integrating machine learning algorithms with historical data. Further levels of complexity are added by choosing suitable characteristics and making sure the model can adjust to changing weather patterns. A smooth connection to a database for storing and retrieving massive volumes of meteorological data also presents issues with timeliness, scalability, and data integrity. Resolving these complex issues is critical to improving weather monitoring and forecast systems' precision and effectiveness. (instrumentchoice, 2025)

## **2.4 Integration of Machine Learning in Weather Prediction**

The use of machine learning to weather forecasting has transformed the discipline and opened the door to the development of more complex and accurate forecasting models. Huge datasets with historical weather patterns are analyzed using machine learning methods including decision trees, neural networks, and support vector machines. These algorithms can forecast future weather conditions by finding complex patterns and correlations within the data. Because machine learning models are flexible, they can include new data and adapt to changing climatic trends, which helps them consistently increase prediction accuracy over time.

By capturing minute details and non-linear correlations, short-term weather prediction systems may be developed, which eventually improves forecasting accuracy. Even with these developments, there are still issues with selecting features, maximizing model performance, and guaranteeing the interpretability of the outcomes. As machine learning develops, its application to weather prediction looks set to improve forecast accuracy by helping us better comprehend the intricate dynamics of the atmosphere. (Reilly, 2023)

## **2 Third Chapter**

# **Methodology**

### **3.1 Research Strategies**

This project uses a mix of hardware and software development, IoT device integration, and machine learning for weather prediction as research methodologies. Through the deliberate use of the following elements, the project seeks to provide a dependable and easily accessible real-time weather monitoring system.

#### **3.1.1 Hardware Configuration**

The hardware configuration involves a Raspberry Pi 2, a Teldus TellStick, and a Wi-Fi router. The Raspberry Pi acts as the central controller, interfacing with the TellStick to fetch sensor data. The devices are connected to the same local network via the Wi-Fi router.

#### **3.1.2 Software Development**

Creating a dashboard to forecast weather data which we gain from the Sensors and developing a weather prediction with machine learning is developed as a part of software development process. The steps of software development include coding, testing, debugging and documentation.

#### **3.1.3 Data collection**

The process of collecting data entails utilizing the Tell Stick to wirelessly communicate temperature values taken in real time from the DHT22 sensor. Furthermore, Kaggle datasets are used to train machine learning models, which improve the precision of short-term weather forecasts. The gathered information is essential for training the prediction models as well as for real-time weather monitoring. And get the weather data through API to the prediction model. And we get some ideas by reviewing past feather forecasting models with IOT integration.

### **3.1.4 Qualitative analysis method**

We explore user and stakeholder tales and experiences while using narrative analysis for our Internet of Things-Based Real-Time Weather Monitoring and Prediction System. We want to comprehend the subjective aspects of major project participants' interactions with the weather monitoring system through user interviews and narratives gathered from them. We identify important components from these stories, such obstacles overcome, accomplishments, and the concrete influence of real-time weather information on decision-making processes, as part of our coding process. In order to identify similar patterns and arcs, structural analysis focuses on dissecting the storytelling structures by going over the beginning, middle, and finish of each tale. The integration of machine learning predictions, the accuracy of weather forecasts, and the dashboard's usability are among the recurrent topics that thematic research reveals.

### **3.1.5 Quantitative analysis method**

Our IoT-Based Real-Time Weather Monitoring and Prediction System uses statistical methods in its quantitative analysis to extract numerical insights from gathered data. The statistical approach entails quantifying and interpreting the data acquired via the methodical use of statistical and mathematical tools. Analyzing numerical data pertaining to temperature measurements, system performance measures, and any other observable variable falls under this category. Standard deviations and mean temperature values are examples of descriptive statistics that provide a brief overview of the dataset's variability and primary tendency. Applying inferential statistics can help shed light on the wider ramifications of our results by allowing us to draw conclusions or predictions about a population from a sample.



# 3 Fourth Chapter

## 4.1 Sensor value monitoring result

The principal aim of the project is to furnish customers with precise and up-to-date temperature information using an Internet of Things-based system. Real-time temperature measurements are successfully captured by the Tell Stick and Raspberry Pi when combined with the DHT22 temperature sensor. A sample of the temperature data gathered during the testing period is shown in the following table:

Date/Time	Temperature	Humidity
2024.01.08 -13.48.49	21.7	20
2024.01.08 -14.07.35	23.8	20

Table 1 sensor value result

And also screenshots and images prove these

At 2024.01.08 -13.48.49

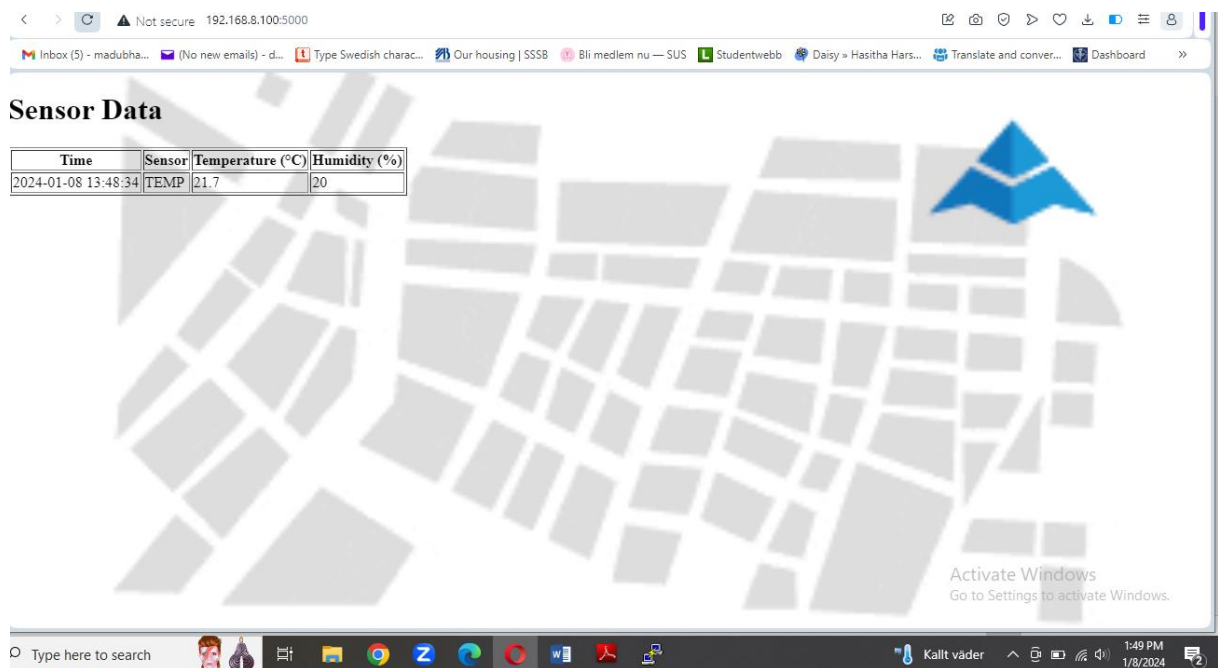


Figure 2 sensor data



*Figure 3 confirmation of sensor data*

## 4.2 Weather Prediction Result

A key component of our research is Weather Prediction, which uses machine learning to provide precise short-term forecasts. The machine learning model that was created and trained using Kaggle datasets has been effectively incorporated into the system. The Flask framework seamlessly integrates the model, which is saved as a.PKL file, to provide a dynamic and responsive user interface. Our system uses an API to retrieve the most recent weather information when users search for a certain location. Our machine learning algorithm, which has been trained beforehand, receives these specifics and uses them to interpret the data and produce short-term weather forecasts. This connection increases the usefulness and efficiency of our Internet of Things (IoT)-based real-time weather monitoring system by ensuring that users receive reliable forecasts in addition to up-to-date weather information.

User Input city name	Current Weather details and Predicted weather	
Colombo		
	Humidity	8.2%
	Max Temperature	28.97 °C
	Min Temperature	28.97 °C
	Max Temp	28.97 °C
	Feels Like	35.1 °C
	Wind Speed	2.11 m/s
	Sea Level Pressure	1009 m/s
	Predicted Weather	Rain, Partially cloudy
Malmo		
	Humidity	8.6%
	Max Temperature	-3.07 °C
	Min Temperature	-5.95 °C
	Max Temp	-3.32 °C
	Feels Like	-8.31 °C
	Wind Speed	3.93 m/s
	Sea Level Pressure	1038 m/s
	Predicted Weather	Snow, Rain, Partially cloudy

<b>Beijing</b>	Humidity	6.5%
	Max Temperature	-5.06 °C
	Min Temperature	-5.06 °C
	Max Temp	-5.06 °C
	Feels Like	-5.06 °C
	Wind Speed	0.04 m/s
	Sea Level Pressure	1021 m/s
	Predicted Weather	Snow, Rain, Partially cloudy

Table 2 weather prediction result

## 4.3 Weather Prediction Testing

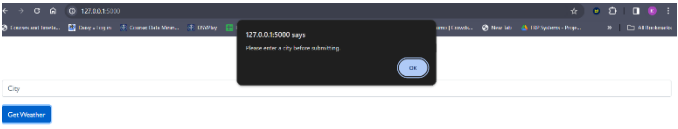

Description	Predicted result	Actual result
User click Predict weather Button without entering a city name	Show notification mentioning ” Please enter a city before submitting.”	
User enter a invalid city name	Show notification mentioning “Please enter a correct city name”	

Table 3 weather prediction testing

# 4 Fifth Chapter

## 5.1 Weather Prediction

With the use of cutting-edge machine learning algorithms, our weather monitoring and prediction system offers a thorough investigation into the field of meteorological forecasting, resulting in precise and timely forecasts. The trip starts with the use of many datasets, most notably the "Seattle" dataset from Kaggle, which includes wind, temperature minimums and maximums, and precipitation. Using the robust XGBoost model, we carefully preprocess the data, filling in any missing values, using cross-validation, and displaying the dataset in several ways. With its remarkable accuracy, this methodical methodology lays the groundwork for a strong predictive model. By incorporating this model into our Flask framework, we are able to create a smooth interface between machine learning and human interaction.

We go from user-inputted weather data to a more useful method in our system's evolution: retrieving real-time values via an API. This modification not only makes the system more useful, but it also presents additional difficulties, most notably the lack of precipitation values in the API data. Not to be defeated, we go to the "Weather" dataset on Kaggle and use XGBoost once more to build a model that can forecast precipitation amounts. Even if the model's accuracy is somewhat lower for this dataset, our dedication to strict data pretreatment methods guarantees accurate predictions.

The integration of the second model with Flask, which results in an easy-to-use online interface for weather prediction, is the ultimate accomplishment of our work. Users are given real-time weather information using this interface, which interacts with the underlying prediction model with ease. The versatility and precision of our system highlight its potential as a useful instrument in the field of weather forecasting, opening the door for more improvements and uses in the constantly changing field of meteorology.

# Weather Prediction model

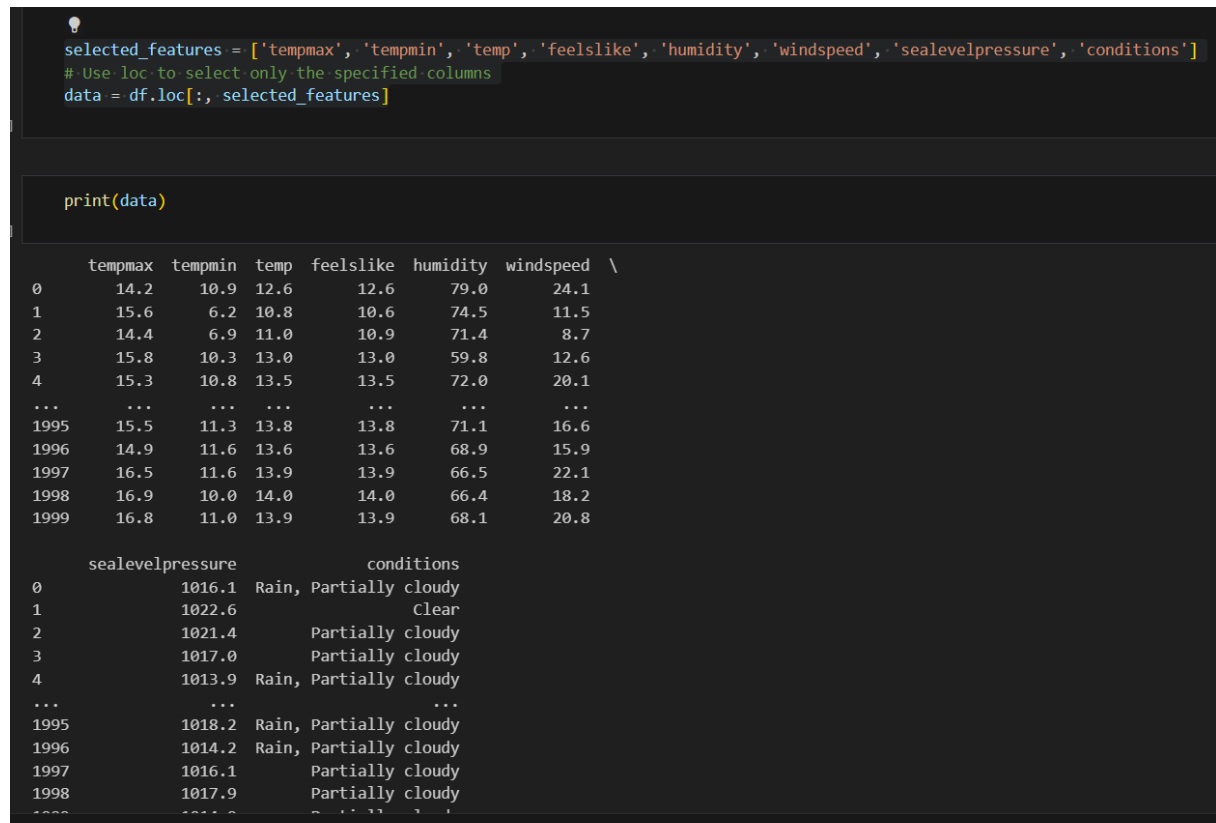


Figure 4 select only relevent columns

A selection of the features—such as variables relating to temperature, humidity, wind speed, sea level pressure, and weather—is chosen for study.

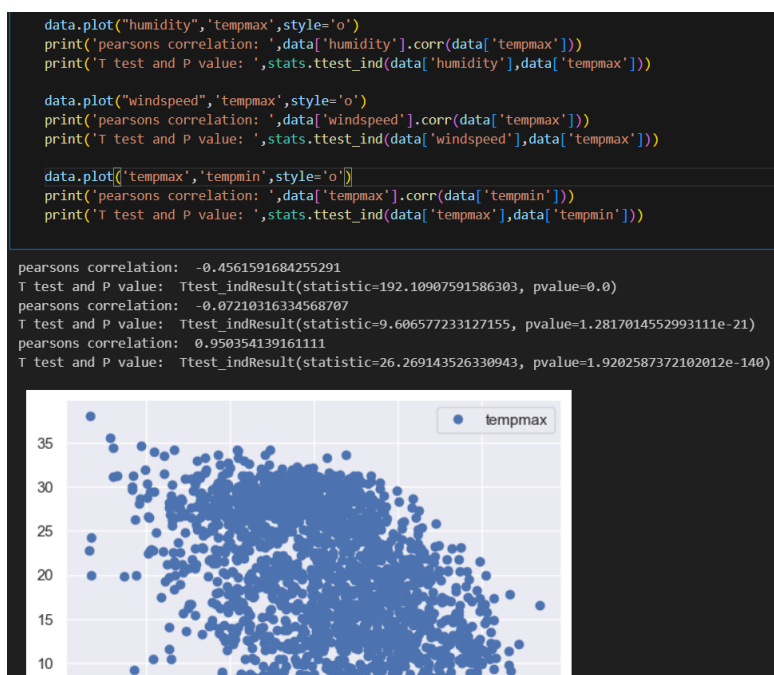


Figure 5 Statistical Testing

The program runs statistical tests to examine the links and significance between certain pairs of numerical variables, including t-tests and Pearson's correlation.

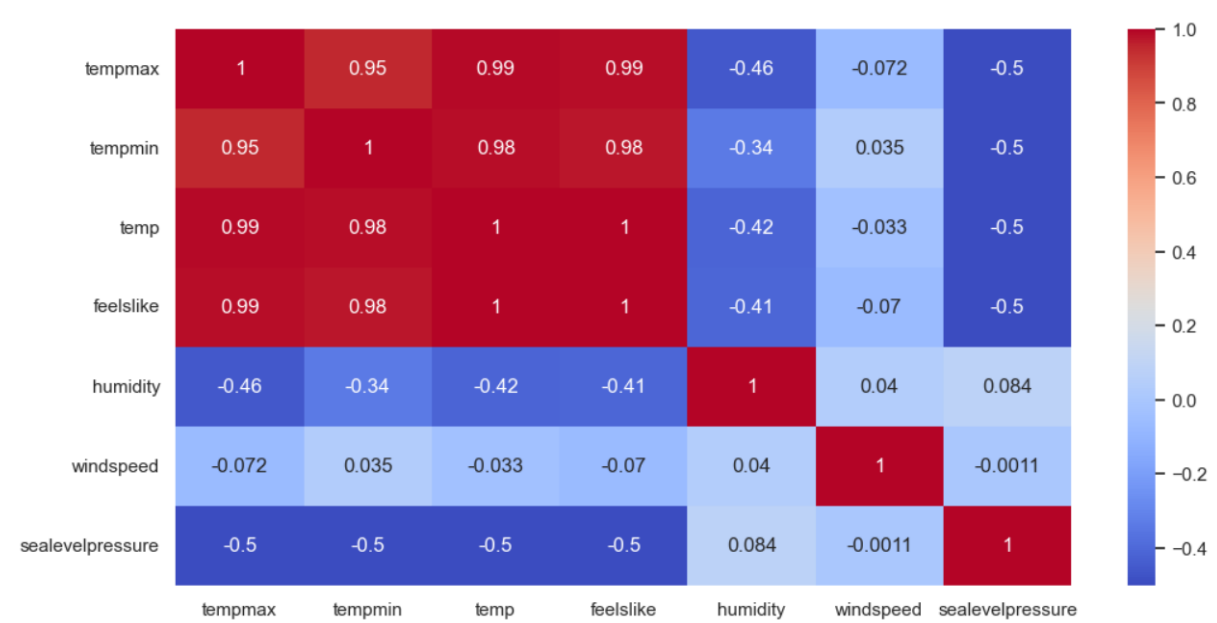


Figure 6 correlation analyze

Seaborn is used in correlation analysis to investigate the connections between numerical variables. Correlations are shown visually via a heatmap. Then i used label encoder to encode weather condition column and handle missing values by filling missing values with mean of the sea level pressure column. And used Interquartile Range (IQR) method to remove the outliers from wind speed and humidty values. Hyperparameter tuning used for XGBoost with grid serach CV to find optimal parameters.

```
import pickle
file = 'model.pkl'
pickle.dump(xgb, open(file, 'wb'))
```

Figure 7 Import my model

The trained XGBoost model is saved to a file using pickle for integrate it with use interface.

## Flask web framework with API

```
@app.route('/get_weather', methods=['POST'])
def get_weather():
    try:
        # Get city from the form
        city = request.form.get('city')

        # Make the API call to OpenWeatherMap
        api_key = '0dab6e6906a53782e5226a108170f24f'
        api_url = f'https://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}&units=metric'

        response = requests.get(api_url)
        data = response.json()

        if response.status_code != 200:
            return jsonify({'error': f'Failed to fetch data. Status code: {response.status_code}',
                            'notification': 'Please enter a correct city name.'})

        # Extract features from the API data
        humidity = data['main']['humidity'] / 10
        tempmax = data['main']['temp_max']
        tempmin = data['main']['temp_min']
        temp = data['main']['temp']
        feelslike = data['main']['feels_like']
        windspeed = data['wind']['speed']
        sealevelpressure = data['main']['sea_level']
```

Figure 8 flask framework with API integration

This Python script takes use of the Flask web framework to build a basic web application that communicates with the OpenWeatherMap API and applies a machine learning model that has already been trained to forecast the weather. The importing of the required modules starts the code. The web application is created using Flask, and other modules include pickle for serializing and deserializing Python objects, jsonify for producing JSON replies, request for handling HTTP requests, and render\_template for displaying HTML templates. Using the pickle module, the script imports a pre-trained machine learning model and opens a file called "model.pkl" in binary read mode. Weather forecasts are subsequently created using this model. The '/get\_weather' route, which manages POST requests, is linked to the get\_weather() method. After obtaining the city name from the completed form, it utilizes an API key to send a call to the OpenWeatherMap API, analyzes the result, extracts pertinent meteorological information, and loads a machine learning model to forecast the weather. With the use of Flask, the OpenWeatherMap API, and a machine learning model, this script creates a web application that lets users enter the name of a city, retrieve current weather information, and use a trained model to forecast the kind of weather. The forecasts are then shown in JSON format with the weather data that was obtained.



```

function getWeather() {
  var city = document.getElementById('city').value;

  if (!city) {
    alert('Please enter a city before submitting.');
```

 return;
 }

 // Make an asynchronous request to Flask app to get weather data
 fetch('/get\_weather', {
 method: 'POST',
 headers: {
 'Content-Type': 'application/x-www-form-urlencoded',
 },
 body: new URLSearchParams({
 'city': city,
 }),
 })
 .then(response => response.json())
 .then(data => {
 console.log('API Response:', data);
 if (data.error) {
 // Handle API error and show notification
 alert(`Error: \${data.error}\n\${data.notification}`);
 } else {
 // Display weather data
 displayWeather(data);
 }
 })
 .catch(error => {
 console.error('Error making API request:', error);
 alert('An error occurred. Please try again.');
 });
}

Figure 9 get weather function

The "Get Weather" button is clicked by the user to initiate this function. The name of the city is retrieved from the input field. It emits an alarm and halts further execution if no city is entered. After that, it sends an asynchronous POST request to the `/get_weather` Flask app endpoint. After processing the answer as JSON, it either shows the weather information or raises an error notice based on the data obtained.

```

function displayWeather(data) {
  var resultContainer = document.getElementById('result-container');

  if (data.error) {
    // Handle API error
    resultContainer.innerHTML = `<p>Error: ${data.error}</p>`;
    return;
  }

  if (data.weather_data.main && data.weather_data.wind) {
    var humidity = data.weather_data.main.humidity/10;
    var tempmax = data.weather_data.main.temp_max;
    var tempmin = data.weather_data.main.temp_min;
    var temp = data.weather_data.main.temp;
    var feelslike = data.weather_data.main.feels_like;
    var windspeed = data.weather_data.wind.speed;
    var sealevelpressure = data.weather_data.main.sea_level;

    // Check if 'sea_level' is not available
    if (sealevelpressure === undefined) {
      sealevelpressure = 'Not available';
    }

    var predictedWeather = data.predicted_weather;
  }
}

```

Figure 10 display weather function

This function is in charge of making the weather information visible on the website. It starts by looking for API errors. In that case, an error notice appears. If meteorological data is available, it pulls out a number of parameters and puts the data into an HTML table. It shows a notice saying that no data was discovered if there is no weather data available.

### 5.1.1 Comparison with Previous Work

By addressing atmospheric model inaccuracies, the article of A literature review on Improvement of Weather prediction by using Machine learning by Deepak Dodake, Chetan Waghmare highlights the importance of neural network design in improving weather forecasts. In a similar vein, our study uses state-of-the-art machine learning algorithms—specifically, the XGBoost model—to improve the accuracy of meteorological forecasts. The paper's discussion of the use of historical data is consistent with our method of training and fine-tuning our model utilizing datasets like Kaggle's "Seattle" dataset. The importance of machine learning techniques in weather prediction is acknowledged in both the study and our research. Furthermore, our choice of characteristics for analysis in the Weather Prediction model aligns with the paper's emphasis on variables including temperature, humidity, wind direction, and rain. Although decision trees, K-NN, and Random Forest algorithms are mentioned in the paper, our project uses the XGBoost model specifically because of its exceptional accuracy. We also incorporate the predictions into an easy-to-use Flask framework that offers real-time API data retrieval, resulting in a seamless interface for weather forecasting. (Deepak Dodake, 2024)

In order to simulate the dynamics of a simplified general circulation model for weather and climate prediction, deep neural networks are applied in this article called Artificial intelligence in weather and climate prediction by Sebastian Scher. It shows that, especially when trained on 50–100 years of data, artificial intelligence is capable of producing stable long-term climatic timeseries and accurate forecasts several days in advance. The study also investigates the use of singular value decomposition in conjunction with neural network techniques to provide probabilistic ensemble forecasts. On the other hand, in order to improve weather, forecast accuracy, our research makes use of machine learning methods, more precisely the XGBoost model. The study focuses on deep neural networks and their shortcomings when it comes to generalizing to new areas of the system, however our approach depends on XGBoost's stability to provide a predictive model that is incorporated into a Flask framework. While the research emphasizes the use of deep neural networks, our effort highlights the possibilities of XGBoost in the context of real-time applications and realistic weather forecasting. Both techniques seek to solve the issues presented by the complexity of weather and climate dynamics. The paper's investigation of uncertainty prediction is consistent with our project's use of probabilistic forecasting techniques and confidence intervals, with the same objective of enhancing forecast dependability. (Scher, 2024)

## 5.2 Sensor data reading

### Hardware Setup

The hardware configuration involves a Raspberry Pi 2, a Teldus TellStick, and a Wi-Fi router. The Raspberry Pi acts as the central controller, interfacing with the TellStick to fetch sensor data. The devices are connected to the same local network via the Wi-Fi router.

### Software Stack

- Raspberry Pi: Operating System: Raspbian
- Python Version: 3.x
- Libraries: Flask, requests
- Teldus TellStick: API Key: Obtained from the Teldus API portal

### Raspberry Pi Configuration

The Raspberry Pi is configured with a static IP address (192.168.8.100) to ensure consistent communication within the local network. Relevant Python libraries are installed using pip.

### Teldus TellStick Configuration

But it didn't configure because of someone get using email address and created a location on teldus tellstick site. Therefore i have to connect with the tellstick customer support and deleted that mail address. This one took about two weeks to solve. Then API keys are obtained from the Teldus API portal, and the TellStick is configured to allow access from the Raspberry Pi.

### Python Code Implementation

Dependencies

Flask: Used for creating a lightweight web server.

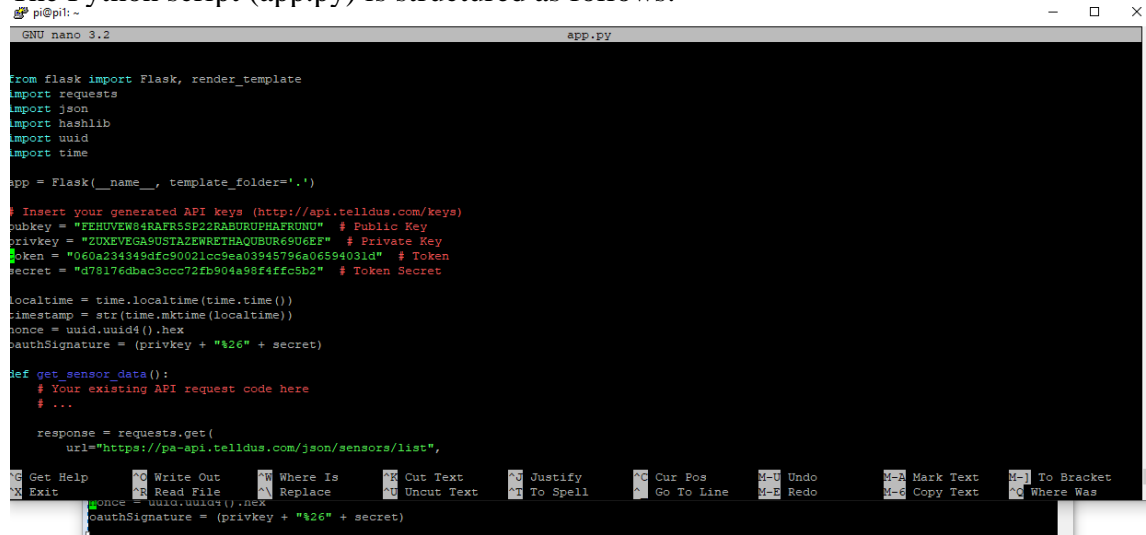
requests: Handles HTTP requests to the Teldus API.

bash

pip install Flask requests

## Code Structure

The Python script (app.py) is structured as follows:



```
GNU nano 3.2 app.py

from flask import Flask, render_template
import requests
import json
import hashlib
import uuid
import time

app = Flask(__name__, template_folder='.')

# Insert your generated API keys (http://api.telldus.com/keys)
pubkey = "FEHUVW84RAFR8SP22RABURUPHAFRUND" # Public Key
privkey = "ZUXEVEG8GUSTAZENRETHAQUUR69UEF" # Private Key
token = "060a234349dfc90021cc9ea03945796a0659403ld" # Token
secret = "d78176dbac3ccc72fb904a98f4ffc5b2" # Token Secret

localtime = time.localtime(time.time())
timestamp = str(time.mktime(localtime))
nonce = uuid.uuid4().hex
oauthSignature = (privkey + "%26" + secret)

def get_sensor_data():
    # Your existing API request code here
    # ...

    response = requests.get(
        url="https://api.telldus.com/json/sensors/list",
        headers={
            "Authorization": 'OAuth oauth_consumer_key="{}", oauth_nonce="{}", oauth_signature_method="PLAINTEXT", oauth_timestamp="{}", oauth_token="{}"',
            "Content-Type": "application/json",
        },
    )

    responseData = response.json()

    sensor_data = []

    for sensor in responseData.get("sensor", []):
        sensor_name = sensor.get("name", "Unknown Sensor")

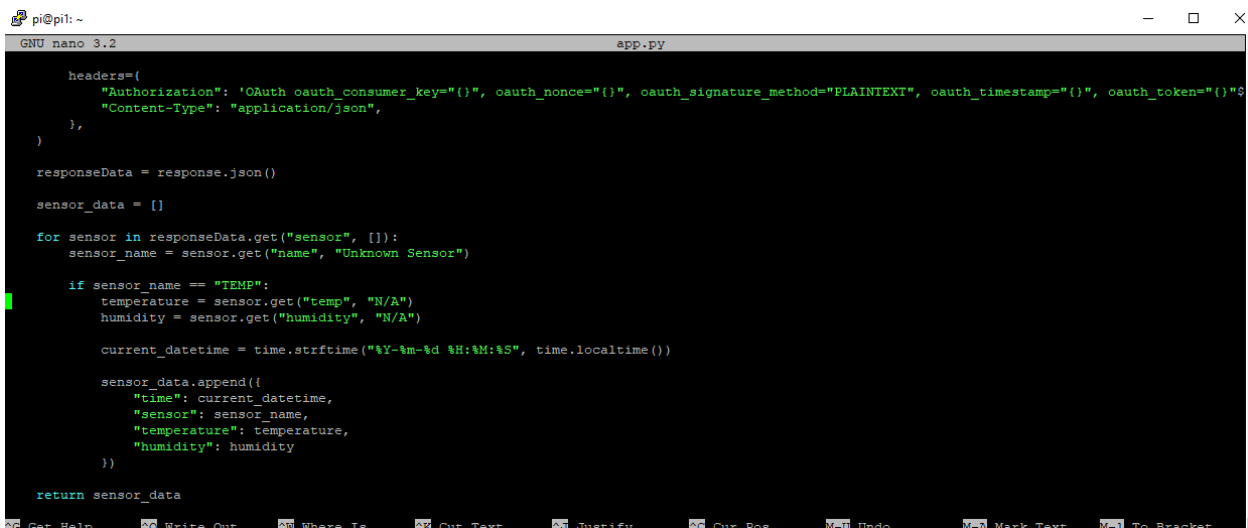
        if sensor_name == "TEMP":
            temperature = sensor.get("temp", "N/A")
            humidity = sensor.get("humidity", "N/A")

            current_datetime = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())

            sensor_data.append({
                "time": current_datetime,
                "sensor": sensor_name,
                "temperature": temperature,
                "humidity": humidity
            })

    return sensor_data
```

Figure 11 app.py script



```
GNU nano 3.2 app.py

        headers={
            "Authorization": 'OAuth oauth_consumer_key="{}", oauth_nonce="{}", oauth_signature_method="PLAINTEXT", oauth_timestamp="{}", oauth_token="{}"',
            "Content-Type": "application/json",
        },
    )

    responseData = response.json()

    sensor_data = []

    for sensor in responseData.get("sensor", []):
        sensor_name = sensor.get("name", "Unknown Sensor")

        if sensor_name == "TEMP":
            temperature = sensor.get("temp", "N/A")
            humidity = sensor.get("humidity", "N/A")

            current_datetime = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())

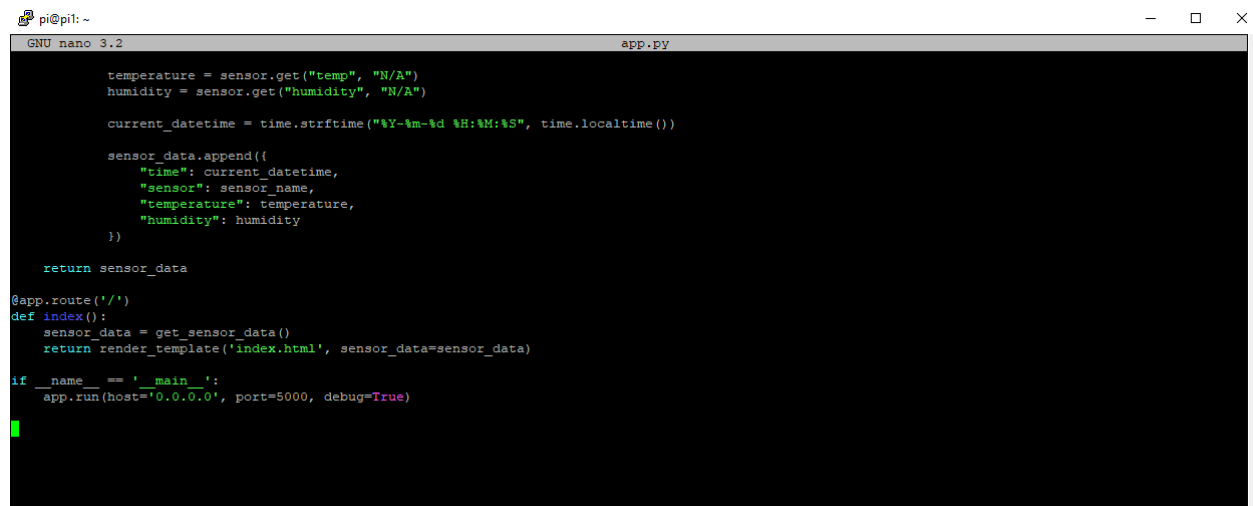
            sensor_data.append({
                "time": current_datetime,
                "sensor": sensor_name,
                "temperature": temperature,
                "humidity": humidity
            })

    return sensor_data

@app.route('/')
def index():
    sensor_data = get_sensor_data()
    return render_template('index.html', sensor_data=sensor_data)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Figure 12 app.py script 2



```
GNU nano 3.2 app.py

            sensor_data.append({
                "time": current_datetime,
                "sensor": sensor_name,
                "temperature": temperature,
                "humidity": humidity
            })

    return sensor_data

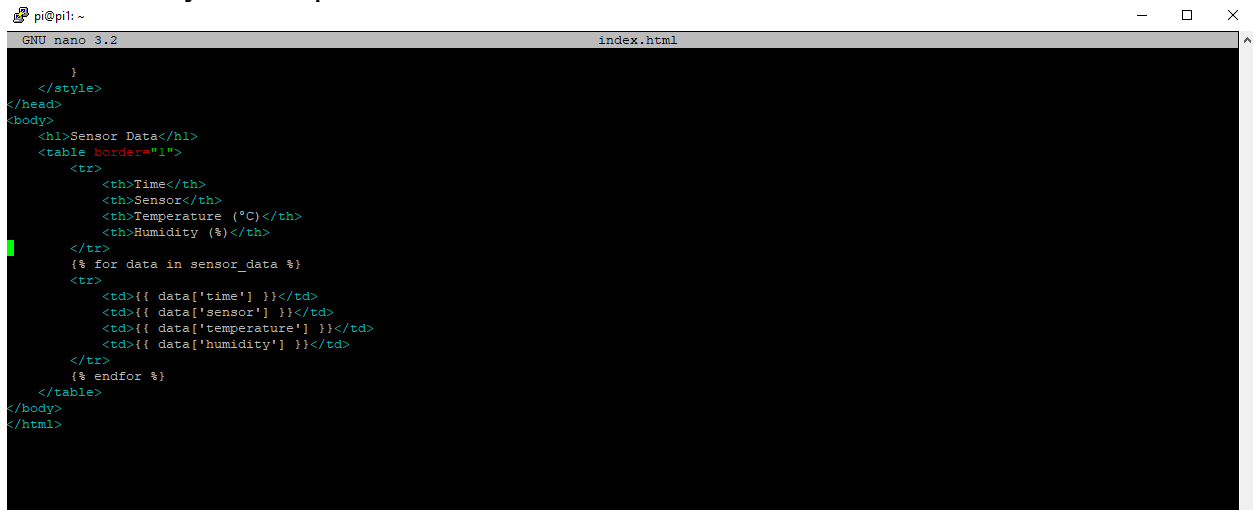
@app.route('/')
def index():
    sensor_data = get_sensor_data()
    return render_template('index.html', sensor_data=sensor_data)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Figure 13 app.py script 3

- Import necessary libraries.
- Define a Flask application.
- Implement a route to fetch sensor data from the Teldus API.
- Extract temperature and humidity values from the API response.
- Render an HTML template with the obtained sensor values.
- HTML Template
- The HTML template (index.html) is a simple structure displaying temperature and humidity values dynamically.

Execute the Python script to start the Flask web server:



```

pi@pi: ~
GNU nano 3.2 index.html
    }
    </style>
</head>
<body>
    <h1>Sensor Data</h1>
    <table border="1">
        <tr>
            <th>Time</th>
            <th>Sensor</th>
            <th>Temperature (°C)</th>
            <th>Humidity (%)</th>
        </tr>
        {% for data in sensor_data %}
        <tr>
            <td>{{ data['time'] }}</td>
            <td>{{ data['sensor'] }}</td>
            <td>{{ data['temperature'] }}</td>
            <td>{{ data['humidity'] }}</td>
        </tr>
        {% endfor %}
    </table>
</body>
</html>

```

Figure 14 flask web interface

The web server will be accessible at <http://192.168.8.100:5000> (replace with your Raspberry Pi's IP address).

### Accessing the Web Page

Open a web browser on any device connected to the local network and navigate to the Raspberry Pi's IP address and port.

### Troubleshooting

Ensure the Raspberry Pi and other devices are connected to the same Wi-Fi network.

Check API key validity and TellStick configuration.

Confirm firewall settings allow access to the Flask app.

(telldus, 2024) (flask.palletsproject, 2024)

## 5.3 Dashboard Integration

Requirements:

1. Python ( $\geq 3.7$ )
2. Flask ( $\geq 2.0.0$ )
3. Flask-Socket IO ( $\geq 5.0.0$ )
4. Pyre base ( $\geq 4.4.0$ )
5. NumPy ( $\geq 1.20.0$ )
6. XGBoost ( $\geq 1.4.0$ )
7. Pickle ( $\geq 5.0.0$ )

Usage:

1. Run the server by executing `python app.py`.
2. Open a web browser and visit `http://localhost:5000/` to view the application

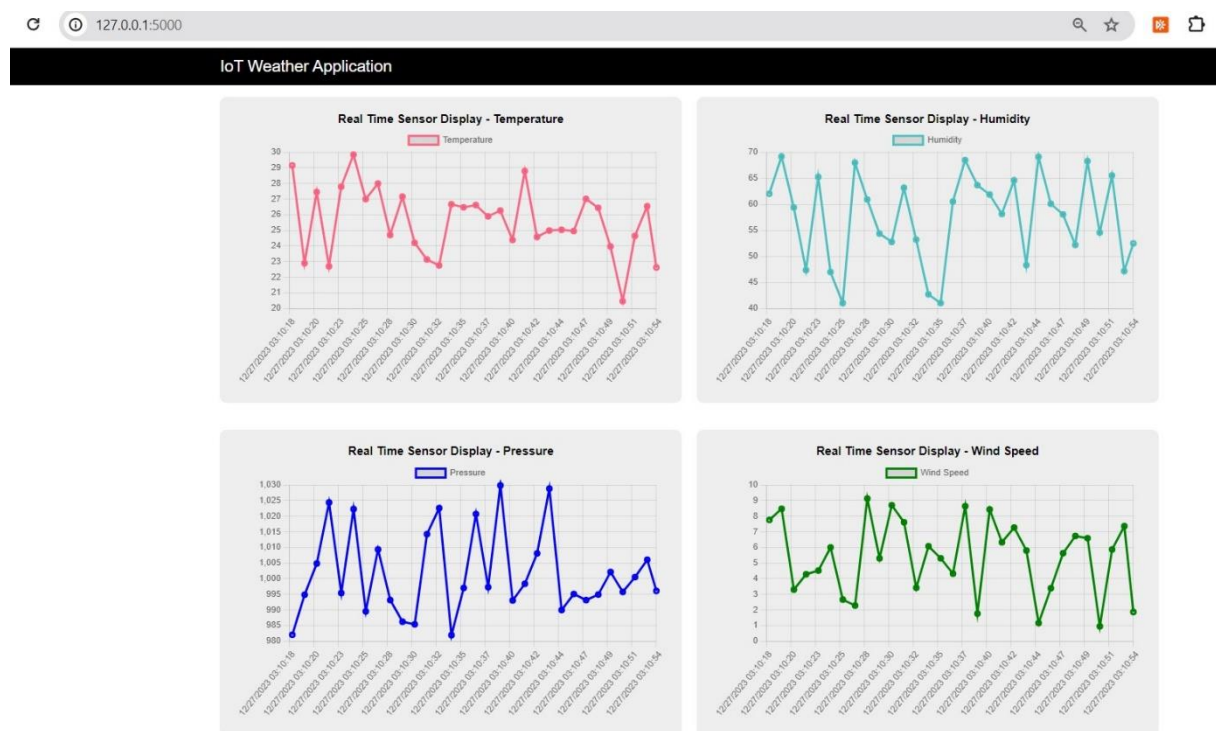


Figure 15 Dashboard

**Algorithm:**

1. The server continuously generates sensor values and sends them to the client through WebSocket.
2. The server receives user input from the client-side form.
3. The server processes the user input and feeds it into the pre-trained machine learning model.
4. The model generates a weather prediction based on the user input and sensor data.
5. The server stores the predicted weather data in a Firebase Firestore database.
6. The server sends the updated weather data to the client-side application.
7. The client-side application displays the updated weather data and sensor data in real-time using a graph.

**Steps involved in building the DB:**

1. Import the necessary libraries
2. Set up the Flask app, Firebase, and Socket IO
3. Define the function 'load\_model' to load the saved XGBoost model
4. Define the function 'predict\_weather' to predict the weather based on the current date and time
5. Load the model and predict the weather
6. Define the function 'get\_current\_datetime' to get the current date and time
7. Define the function 'background\_thread' to generate random sensor values and send them to the Firebase database and the frontend via Socket IO
8. Initialize a global variable 'thread' and a lock for thread safety
9. Define the route '/' to render the 'index.html' template
10. Define the route '/predict' to get the input data, make the prediction, save the data to Firebase, and render the 'index.html' template with the result
11. Define the function 'save\_to\_firestore' to save the data to Firebase
12. Define the Socket IO event 'connect' to start the background thread
13. Define the Socket IO event 'disconnect' to handle the client disconnection
14. Run the Flask app with Socket IO

# References

- (2024, 01 08). Retrieved from <https://api.telldus.com/>
- Abdulkadir H. Alkali1, S. J. (2023, 12 25). *researchgate*. Retrieved from researchgate.net: [https://www.researchgate.net/publication/354499488\\_Design\\_Implementation\\_and\\_Evaluation\\_of\\_Remote\\_Weather\\_Monitoring\\_System](https://www.researchgate.net/publication/354499488_Design_Implementation_and_Evaluation_of_Remote_Weather_Monitoring_System)
- Deepak Dodake, C. W. (2024, 01 07). *ijrpr*. Retrieved from ijrpr.com: <https://ijrpr.com/uploads/V3ISSUE5/IJRPR3805.pdf>
- flask.palletsproject*. (2024, 01 08). Retrieved from flask.palletsproject.com: <https://flask.palletsprojects.com/en/3.0.x/>
- Gillis, A. S. (2023, 12 23). *techtarget*. Retrieved from techtarget.com: <https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT>
- instrumentchoice*. (2025, 12 25). Retrieved from instrumentchoice.com: <https://www.instrumentchoice.com.au/common-issues-with-weather-stations-and-how-to-fix-them>
- Meloni, P. (2023, 12 25). *medium*. Retrieved from medium.com: <https://medium.com/@pietro.meloni/the-limitations-of-weather-forecasting-310a75ab9a08>
- Reilly, J. (2023, 12 25). *akkio*. Retrieved from akkio.com: <https://www.akkio.com/post/weather-prediction-using-machine-learning#:~:text=ML%20can%20integrate%20diverse%20data,information%20sources%20than%20traditional%20methods>
- Scher, S. (2024, 01 08). *diva-portal*. Retrieved from diva-portal.org: <https://su.diva-portal.org/smash/get/diva2:1425352/FULLTEXT01.pdf>
- telldus*. (2024, 01 08). Retrieved from telldus.com: <https://api.telldus.com/>
- Wendt, Z. (2023, 12 23). *arrow*. Retrieved from arrow.com: <https://www.arrow.com/en/research-and-events/articles/weather-forecasting-and-iot>
- Wolf, F. (2023, 12 25). *mergeflow*. Retrieved from mergeflow.com: <https://mergeflow.com/research/emerging-technologies-better-weather-forecasting>







