IOT Weather data Monitoring Python GUI for Real-Time and Historical Data Visualization

Yihang Qiao - 649112361@qq.com

Achyut Jagini - achyut.jagini@gmail.com

Tianzhi Wang - tanzhunter97@gmail.com

Stockholm University

Table of Contents

Abstract

This report covers the Internet of Things (IoT) project of Weather data Monitoring and making a Python GUI for Real-Time and Historical Data Visualization.

The GUI application has features of
1)Historical weather data visualization
2)Weather trend analysis
3)Map integration showing device location
4)Real-time weather data display using raspberry Pi

Keywords: Internet of Things ,Python , GUI , Weather monitoring

Introduction

In this current era, The Internet of Things (IoT) has emerged as a revolutionary technology, particularly in environmental monitoring. This report delves into our IoT project aimed at harnessing weather data for comprehensive analysis and visualization.

Method

An open source WeatherHistory dataset is used which is obtained from the internet.

The dataset contains 96,453 entries and 12 features.

The features are Formatted Date, Summary, Precip Type, Temperature (C), Apparent Temperature (C), Humidity, Wind Speed (km/h), Wind Bearing (degrees), Visibility (km), Loud Cover, Pressure (millibars), Daily Summary.

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
20	2006-04-01 20:00:00.000 +0200	Mostly Cloudy	rain	11.550000	11.550000	0.77	7.3899	147.0	11.0285	0.0	1015.85	Partly cloudy throughout the day.
44	2006-04-10 20:00:00.000 +0200	Mostly Cloudy	rain	16.061111	16.061111	0.53	21.3969	149.0	9.9820	0.0	1008.11	Mostly cloudy throughout the day.
68	2006-04-11 20:00:00.000 +0200	Overcast	rain	8.961111	5.777778	0.93	23.2162	340.0	3.8157	0.0	1004.85	Foggy in the evening.
92	2006-04-12 20:00:00.000 +0200	Mostly Cloudy	rain	9.900000	7.716667	0.66	15.7297	348.0	11.0285	0.0	1005.48	Foggy overnight and breezy in the morning.
116	2006-04-13 20:00:00.000 +0200	Overcast	rain	8.072222	6.433333	0.75	9.5151	150.0	9.9820	0.0	1010.91	Overcast throughout the day.
140	2006-04-14 20:00:00.000 +0200	Mostly Cloudy	rain	13.850000	13.850000	0.64	11.5598	231.0	11.4471	0.0	1013.42	Mostly cloudy throughout the day.
164	2006-04-15 20:00:00.000 +0200	Mostly Cloudy	rain	13.088889	13.088889	0.67	9.2736	138.0	9.9820	0.0	1017.38	Mostly cloudy throughout the day.
188	2006-04-16 20:00:00.000 +0200	Mostly Cloudy	rain	13.922222	13.922222	0.78	11.7691	279.0	11.2861	0.0	1011.64	Mostly cloudy throughout the day.

Image 1 - one original dataset

A machine learning model is created using dataset

Real time weather data is collected using Raspberry Pi

Machine learning is also used for real time weather prediction

Google maps API is used for map of device

A GUI is created with the features of

1)Historical weather data visualization

2)Weather trend analysis

3)Local news generate

4)Map integration showing device location

5)Real-time weather data display and analysis

Design and Development

Historical Data Visualization and Analysis

Preprocessing

The weather data was loaded from a CSV file weatherHistory.csv using pandas.

Preliminary analysis was conducted to understand the dataset's structure using the .info() and .describe() methods.

The dataset was sorted based on the 'Formatted Date' to maintain chronological order.

Missing values were imputed, and irrelevant features like 'Daily Summary' and 'Loud Cover' were dropped to streamline the dataset.

Label encoding was applied to categorical variables such as 'Precip Type' to convert them into a format suitable for modeling.

The 'Wind Bearing (degrees)' and 'Wind Speed (km/h)' features were transformed into 'Wx' and 'Wy' components to capture wind direction more effectively.

Model

A Sequential model was constructed using Keras, with Dense layers chosen for their efficacy in handling sequential data.

The model architecture was composed of a dense input layer, multiple Dense layers, and a dense output layer to predict one feature for a time.

The Adam optimizer and mean squared error loss function are used.

Model Training

The data was split into training, validation, and testing sets to evaluate the model's performance and prevent overfitting.

A function MinMaxScaler() was used to reshape the data so it is suitable for Dense input.

```
def read_data(N, df, column_name='humidity'):

    # Assuming the column_name exists in your CSV file
    data = df[column_name].values

    X = []
    Y = []

    for i in range(N, len(data)):
        s = []
        for j in range(i - N, i):
            s.append(data[j])
        X.append(s)
        Y.append(data[i])

    return np.array(X), np.array(Y)
```

```
X,Y=read_data(7,df,column_name='humidity')

min_max_scaler = MinMaxScaler()
min_max_scaler.fit(X)
x = min_max_scaler.transform(X)
x_ = min_max_scaler.transform([[80.5,89.0,76.0,75.0,77.0,81.5,82.0]])
y=Y
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2)

model = keras.Sequential([
        keras.layers.Dense(500,activation='relu',input_shape=[7]),
        keras.layers.Dense(500,activation='relu'),
        keras.layers.Dense(250,activation='relu'),
        keras.layers.Dense(250,activation='relu'),
        keras.layers.Dense(1)])
model.compile(loss='mean_absolute_error',optimizer='Adam')
model.fit(x_train,y_train,batch_size = 126,epochs=1000)
path = '/content/gdrive/MyDrive/model/model_hum1.h5'
model.save(path)

y_=model.predict(x_)
```

Image 2 - Code creating model

The model was trained on the training dataset with a specified number of epochs and batch size, with checkpoints used to save the best model

The trained model is saved into a file.

Performance Evaluation

Predictions were made using trained models and graphs were obtained.

```
from keras.models import load_model

path = '/content/gdrive/MyDrive/model/model_hum.h5'
loaded_model = load_model(path)

min_max_scaler = MinMaxScaler()
min_max_scaler.fit(X)
x_new_processed = min_max_scaler.transform([[70,72.2,74.3,76.5,78.2,80,81]])
print(x_new_processed)
predictions = loaded_model.predict(x_new_processed)

print('predict :', predictions)
```

```
[[0.51666667 0.55333333 0.58833333 0.625      0.65333333 0.68333333
  0.7        ]]
1/1 [==============================] - 0s 71ms/step
predict : [[80.441185]]
```

Image 3 - Code testing model

Historical data visualization

We saved the visualized historical data into HTML and will be opened in the browser.

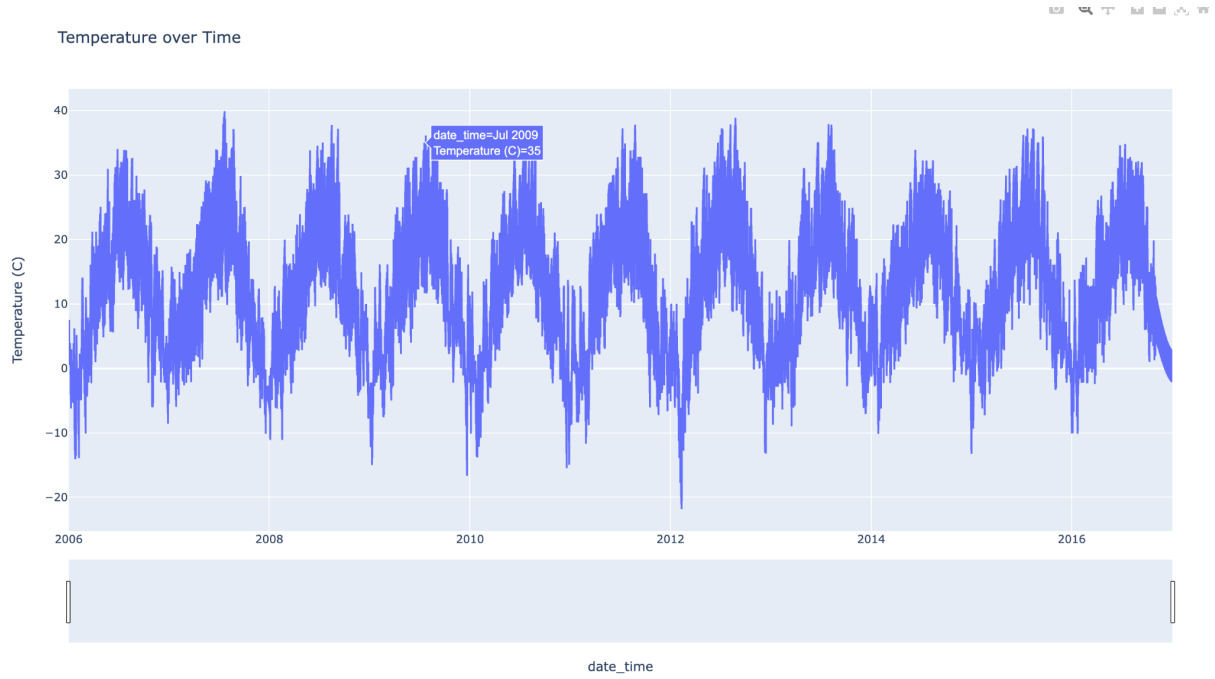

Image 4 - Temperature visualize

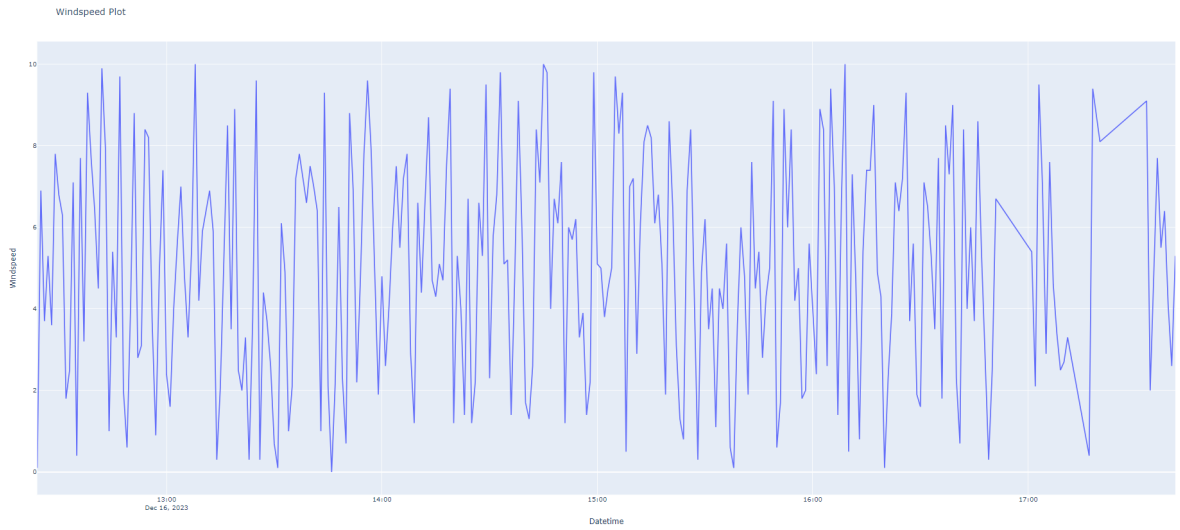

Image 5 - wind speed visualize

Local News

We use gnewsclient to get local news in Swedish and linked with a button to refresh and present it.

```
def update_news(self, button_number):
    # use gnewsclient get news
    client = gnewsclient.NewsClient(language='swedish', location='Sweden', topic='World',
    news_list = client.get_news()

    # get botton
    button = getattr(self, f'news_button_{button_number}')

    # refresh label
    for idx, item in enumerate(news_list):
        if idx == button_number - 1:
            # 1 will not change
            if button_number != 1:
                button.setText(item['title'])
            break

        # press 1 refresh 2-5
    if button_number == 1:
        for i in range(2, 6):
            self.update_news(i)
```

Image 6 - code to update news

Map Integration

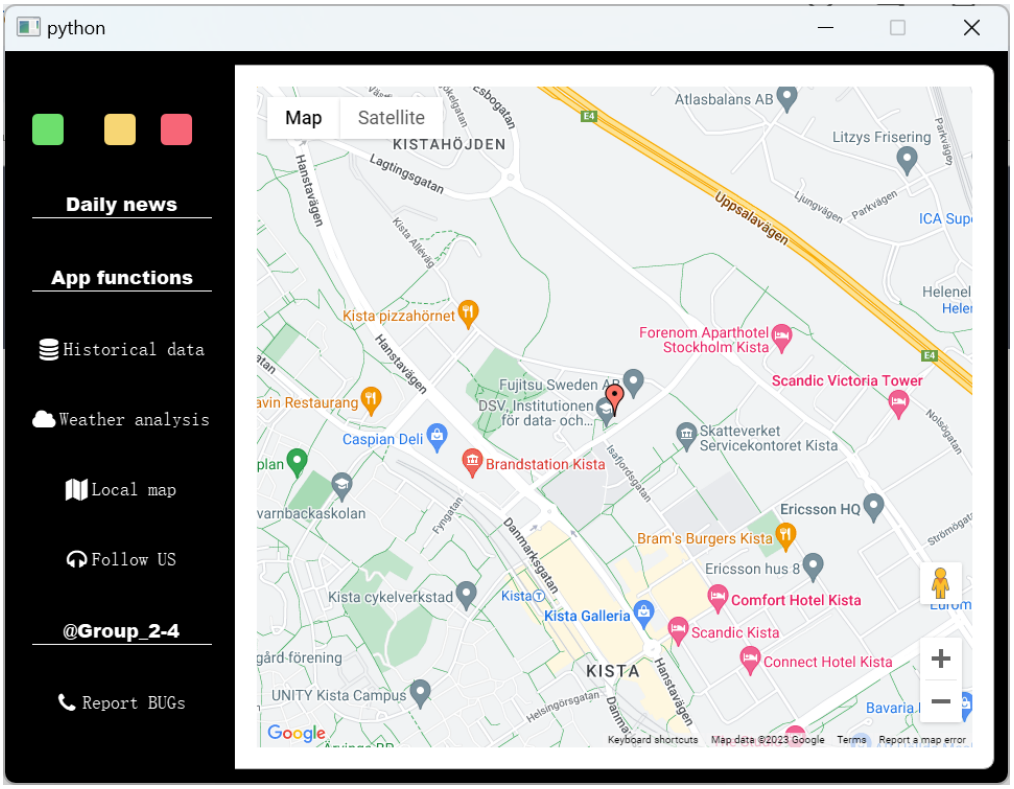

Image 7 - Map display

We've added a map to our Python GUI to show device location. We used PyQt5 for this function because it's good for building user interfaces. Our map feature uses the Google Maps API, which lets us put an interactive map right into our program.

To show the location of the IoT device, we put a marker on the map. We write some HTML and JavaScript code to get the map from Google Maps and put it in a QWebEngineView widget, which is a part of PyQt5 that can show web pages. The map loads in the background asynchronously so it doesn't slow the app down, and this enables users to see and use the map without any lag. Using the Google Maps API is helpful because it's reliable and lets us customize the map as we need.

GUI

The GUI was written using python library PyQt5.

The GUI has buttons that link to the different features of the app.

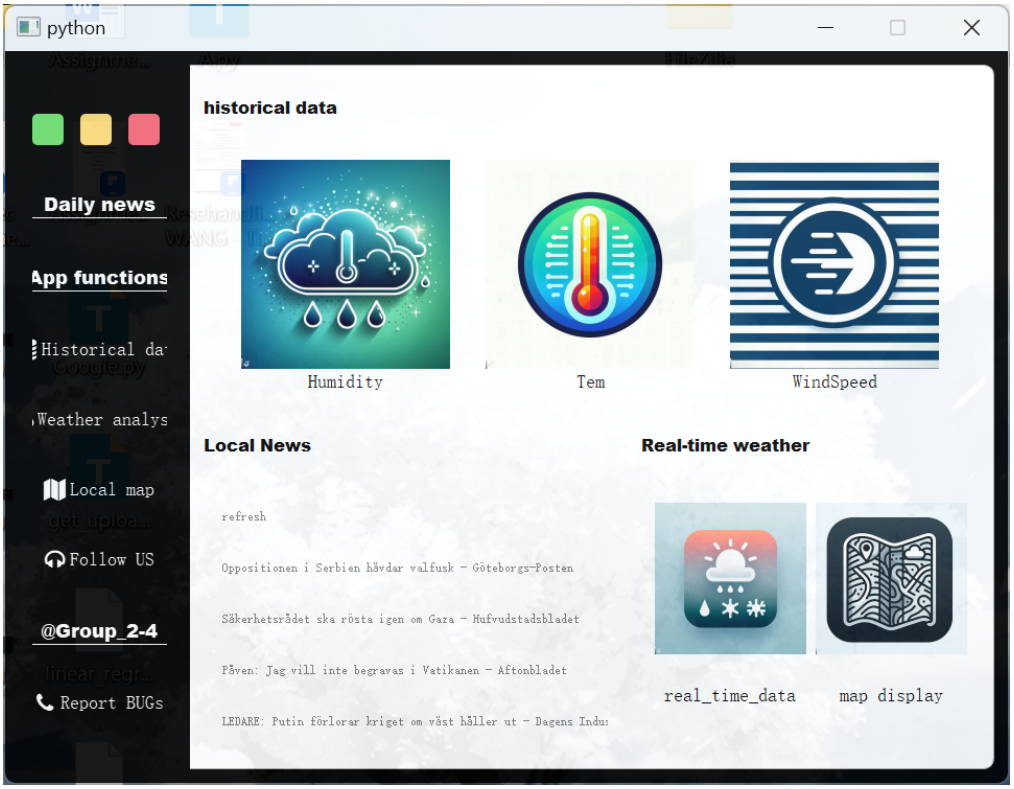

Image 8 - Main python GUI

The GUI of the weather monitoring application is intuitive and user-friendly. The main screen is divided into sections .

The local news section can be refreshed and fetches the news real-time.

The map display links to the map functionality showing the device location.

The historical data analysis shows the model result for predictions on historical data.

Real time data shows the graphs for real time weather data collected by sensors and the Raspberry pi.

Raspberry Pi part

We wrote a python script to get data from the sensors connected to the Raspberry Pi. The real time data is then transmitted to the computer via the MQTT protocol. The computer can use the received data for further analysis and presentation. In addition, the data is saved locally and uploaded to the google drive using the Google api.

```python
def get_sensor_data(sensor_id):
    command = "tdtool --list-sensors | awk -F '[= \t]+' '/id={}/ {{print $10, $12}}'".format(sensor_id)
    result = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE)
    output = result.communicate()[0].decode("utf-8").strip()
    return output


mqtt_broker_address = "raspberrypi2b"
mqtt_topic = "sensor_data"
csv_file_path = "WeatherData.csv"
# sensor ID
sensor_id = 247
```

Image 9 - Get sensor data

```python
    # publish  MQTT
    message = "Time: {}, Temperature: {:.2f}, Humidity: {}, Windspeed: {:.2f}".format(current_time, temperature, humidity, windspeed)
    publish.single(mqtt_topic, message, hostname=mqtt_broker_address)

    print("Published:", message)
```

Image 10 - Publish data

Real time data collection and analysis

The Raspberry pi is used to collect real time weather data.

The real time weather data is displayed in the form of a graph in the GUI in the weather analysis function.

Separate graphs are obtained for temperature, humidity and wind speed

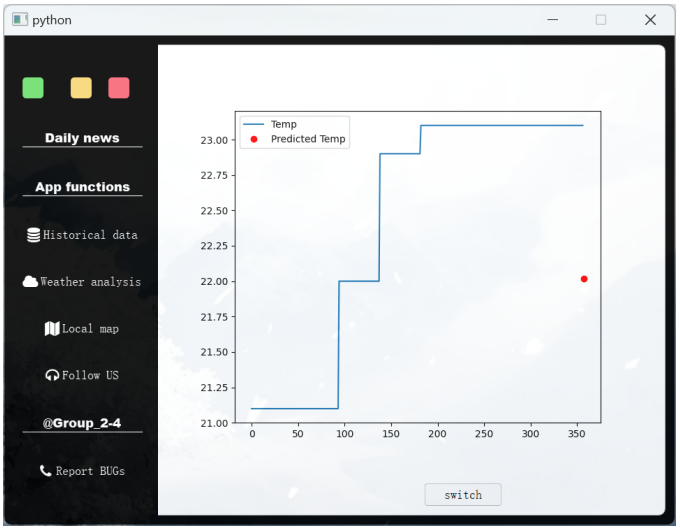

Image 11 - Real-time Temperature show and predict

Image 12 - Real-time Humidity show and predict

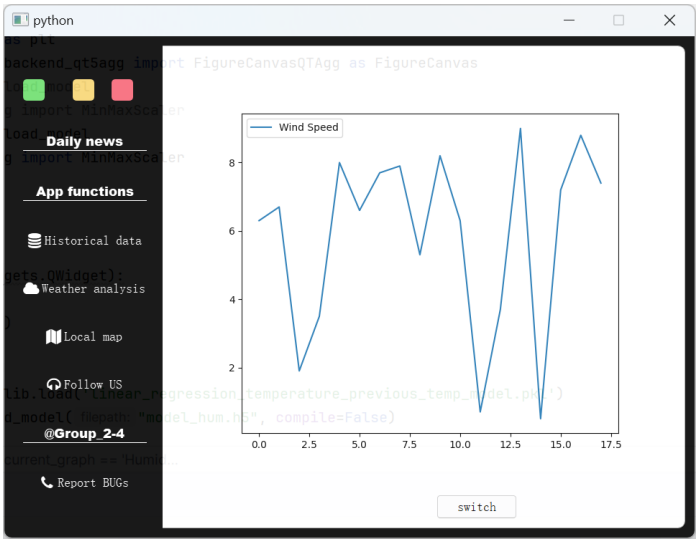

Image 13 - Real-time Wind speed show

Discussion

The reason why we choose this project is that IOT plays a more and more important role in our life. Using IOT we can generate data from the environment and use several methods to analyze data.

In this project, we visualize the historical data to analyze the weather trends futures and use machine learning to create model and predict the weather which involves monitoring real-time weather from local sensor and transfer message from Raspberry Pi to client, implement machine learning models in GUI, storage and use the transferred message to predict future data.

This project aims to raise people's awareness of changing weather in living space by helps people to compare weather nowadays with historical data.

Future work

Our long-term vision for the weather monitoring application is centered around scaling our application to a global level. With accurate models already in place, we foresee the application becoming a tool for individuals and communities worldwide. The scalability of our application is about reaching more users and also enhancing the precision and relevance of the weather information provided, regardless of geographic location.

As we look toward the future, we imagine our application becoming a platform that serves a diverse user base. The goal is to refine our existing models to ensure they maintain high accuracy as they handle a larger variety of environmental conditions across different regions.

We can leverage cloud technologies that allow for the efficient handling of data and high user volumes without compromising on performance.

References

Python Software Foundation (2023) Python 3 Documentation
https://docs.python.org/3/ [2023-11-20]

PyQt5 Documentation (2023) Introduction to PyQt5
https://www.riverbankcomputing.com/static/Docs/PyQt5/ [2023-11-21]

Matplotlib Developers (2023) Matplotlib Documentation
https://matplotlib.org/stable/users/index.html[2023-11-20]

Pandas Development Team (2023) Pandas Documentation
https://pandas.pydata.org/pandas-docs/stable/ [2023-11-20]

Raspberry Pi Foundation (2023) Raspberry Pi Documentation
https://www.raspberrypi.org/documentation/ [2023-11-24]

Github link

https://github.com/achyutjagini/IOT-project-weather-data-GUI