HO CHI MINH CITY

UNIVERSITY OF TECHNOLOGY AND EDUCATION

FACULTY OF INTERNATION EDUCATION

REPORT

EMBEDDED SYSTEM

Lecture: PhD. Bui Ha Duc

Name:	**ID:**
Nguyễn Châu Tấn Cường	**23146007**

Ho Chi Minh, 1 June, 2025

Table of Contents

I. Project Introduction

1. Motivation

Accurate, localized weather information is indispensable for numerous activities, from agricultural planning to daily commuting. In many rural or remote regions, traditional meteorological services remain either prohibitively expensive or altogether unavailable, leaving communities vulnerable to sudden weather changes. This project " ***Weather Prediction Using Machine Learning Based / BMP180 Driver*** "addresses that gap by leveraging a low cost BMP180 temperature and pressure sensor, coupled with a Raspberry Pi 3B+, to collect real time environmental data at any chosen location. Concurrently, publicly accessible historical weather records from WorldWeatherOnline enable the training of robust machine learning models that capture broader atmospheric patterns. When integrated into a cohesive pipeline sensor acquisition, data logging, model inference, and user facing visualization this system makes reliable short term forecasts accessible without dependence on national weather stations or paid subscriptions. In effect, we aim to democratize weather intelligence by combining Internet of Things (IoT) hardware with modern machine learning techniques to create an affordable, scalable, and user friendly platform.

2. Objectives

In the initial phase, we focus on collecting and preparing weather data from the WorldWeatherOnline API, specifically the time stamped temperature and pressure readings. All raw data are stored in CSV files to facilitate subsequent preprocessing and model training. Once data collection is complete, we address missing values and outliers, normalize the temperature and pressure ranges, and if needed create additional features to enhance predictive performance. Next, we implement and train eight different machine learning algorithms:

- Decision Tree
- Random Forest
- Gradient Boosting
- K-Nearest Neighbors (KNN)
- Support Vector Machine (SVM)
- Logistic Regression
- Naive Bayes
- Neural Network

Each model’s hyperparameters are tuned via cross validation to achieve the highest accuracy and generalization on the training data. After evaluating individual performance using metrics such as accuracy, precision, recall, and F1-score, we combine all eight base learners through a voting ensemble. This ensemble approach improves reliability and stability of predictions and reduces misclassification rates compared to any single model.
On the hardware side, we develop a driver for the BMP180 sensor attached to a Raspberry Pi 3B+. We read calibration coefficients directly from the BMP180 registers according to Bosch’s datasheet, then apply the manufacturer’s formulas to convert raw ADC values into

actual temperature (°C) and pressure (Pa). Sensor readings are continuously logged to a local CSV file on the Raspberry Pi, ensuring historical data is available for administrative purposes and further analysis. Finally, we build a FastAPI web application that runs on the same Raspberry Pi. This web app exposes REpresentational State Transfer endpoints to:

- Serve real time sensor data (temperature and pressure)
- Provide access to historical records
- Machine learning based weather forecasts using the two inputs temperature and pressure

The web interface includes a simple login system and displays interactive charts such as "Pressure vs. Time," "Temperature vs. Time," as well as historical data tables and forecast results with confidence scores. Parallel to the web app, we create a mobile application that mirrors these features. On the mobile side, users do not need to log in; they simply enter the network address of any Raspberry Pi + BMP180 node to view live data and predictions, no shared LAN is required, and access is possible from any Internet connection.

II. Implementation Methodology

1. Data Collection & Preprocessing

We begin by fetching historical and current weather data from the WorldWeatherOnline API, focusing specifically on timestamped temperature and barometric pressure values. Each API request returns a JSON payload, which we parse to extract the necessary fields and immediately append to a CSV file stored locally. Over time, these CSV files accumulate hundreds or thousands of records, capturing variations across days, weeks, or months. Once a sufficient volume of raw data is available, we perform preprocessing steps: missing or null temperature/pressure readings are identified and imputed either by interpolation from neighboring timestamps or, if isolated, by removing those rows entirely. Outliers that lie beyond three standard deviations from the mean are examined; if they appear to be sensor glitches or API errors, they are either corrected (when possible) or discarded. Finally, we normalize both temperature and pressure columns to a consistent scale so that subsequent machine learning models treat each feature comparably.

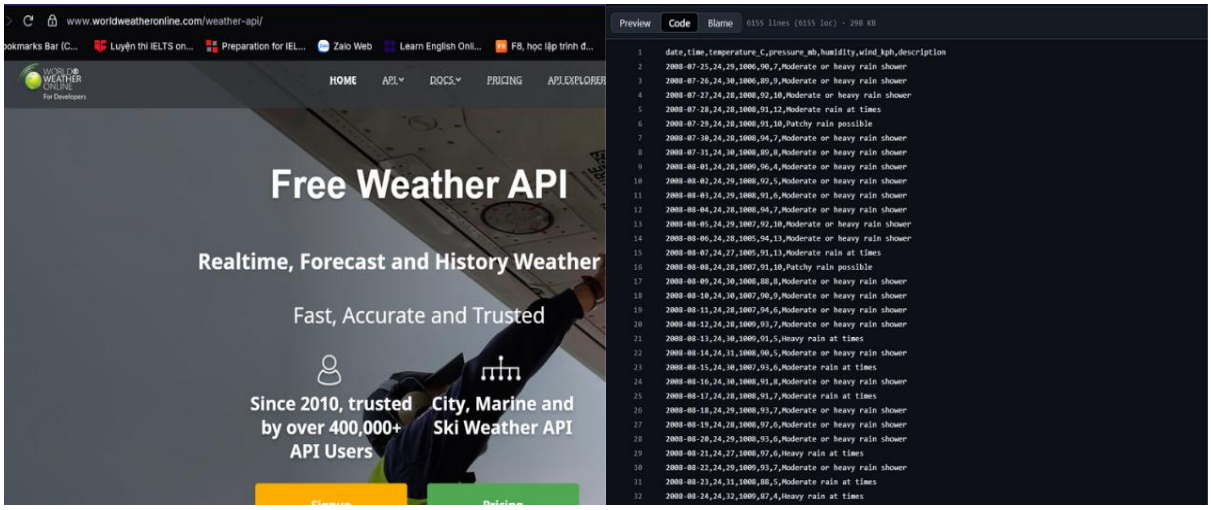

Fig 1:WorldWeatherOnline page(Left), Data from WWO(Right)

2. Machine Learning Training & Ensemble

With a clean, normalized dataset in hand, we proceed to train eight individual machine learning algorithms. For each model, we split the data into training and validation sets (commonly an 70/30 split) and perform k fold cross validation (k=5) to tune hyperparameters such as tree depth for decision trees, number of estimators for ensemble methods, number of neighbors for KNN, kernel and C parameter for SVM, regularization strength for logistic regression, and the architecture (number of hidden layers/neurons) for the neural network. After identifying optimal hyperparameters, we retrain each algorithm on the full training set and evaluate its performance on the held out validation set using accuracy, precision, recall, and F1-score. Although some models like Random Forest and Gradient Boosting tend to outperform simpler classifiers on tabular data, each algorithm's strengths and weaknesses can vary depending on seasonal patterns or API data quality. To leverage the complementary advantages of all eight, we implement a voting ensemble: for a given timestamped pair (temperature, pressure), each base learner give predicts; the ensemble then chooses the class that receives the majority of votes. In cases of a tie, we default to the model with the highest validation accuracy. Empirically, this combined approach yields more robust predictions, reducing variance and error compared to any single model alone.

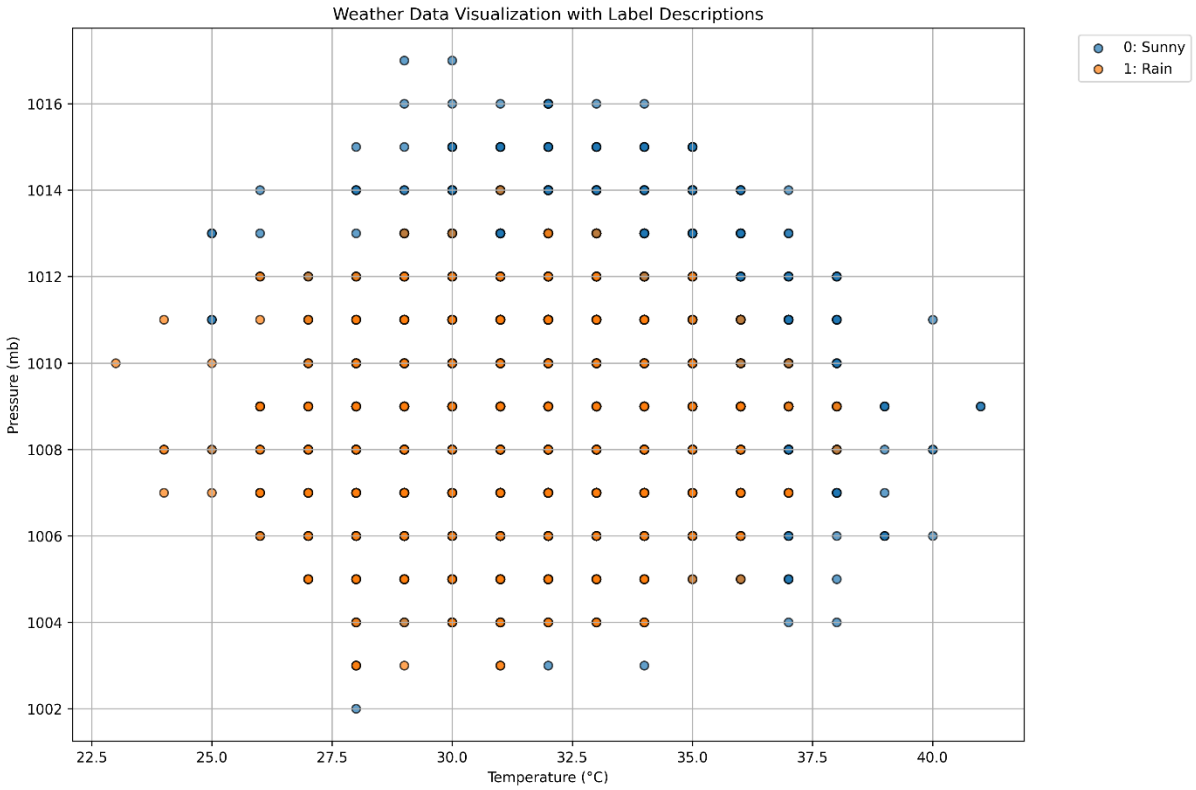

Fig 2:Data after processing

Logistic Regression				
Lớp	*Precision*	*Recall*	*F1-score*	*Support*
0	*0.80*	*0.76*	*0.78*	*1026*
1	*0.72*	*0.77*	*0.74*	*821*
Accuracy			*0.76*	*1847*
Macro avg	*0.76*	*0.76*	*0.76*	*1847*
Weighted avg	*0.77*	*0.76*	*0.76*	*1847*

DecisionTree				
Lớp	*Precision*	*Recall*	*F1-score*	*Support*
0	*0.78*	*0.78*	*0.78*	*1026*
1	*0.73*	*0.73*	*0.73*	*821*
Accuracy			*0.76*	*1847*
Macro avg	*0.75*	*0.75*	*0.75*	*1847*
Weighted avg	*0.76*	*0.76*	*0.76*	*1847*

RandomForest				
Lớp	*Precision*	*Recall*	*F1-score*	*Support*
0	*0.79*	*0.76*	*0.78*	*1026*
1	*0.71*	*0.75*	*0.73*	*821*
Accuracy			*0.76*	*1847*
Macro avg	*0.75*	*0.76*	*0.76*	*1847*
Weighted avg	*0.76*	*0.76*	*0.76*	*1847*

NaiveBayes				
Lớp	*Precision*	*Recall*	*F1-score*	*Support*
0	*0.80*	*0.78*	*0.79*	*1026*
1	*0.73*	*0.75*	*0.74*	*821*
Accuracy			*0.77*	*1847*
Macro avg	*0.77*	*0.77*	*0.77*	*1847*
Weighted avg	*0.77*	*0.77*	*0.77*	*1847*

GradientBoosting				
Lớp	*Precision*	*Recall*	*F1-score*	*Support*
0	*0.80*	*0.77*	*0.79*	*1026*
1	*0.73*	*0.76*	*0.74*	*821*
Accuracy			*0.77*	*1847*
Macro avg	*0.76*	*0.77*	*0.77*	*1847*
Weighted avg	*0.77*	*0.77*	*0.77*	*1847*

NeuralNetwork				
Lớp	*Precision*	*Recall*	*F1-score*	*Support*
0	*0.81*	*0.76*	*0.78*	*1026*
1	*0.72*	*0.78*	*0.75*	*821*
Accuracy			*0.77*	*1847*
Macro avg	*0.77*	*0.77*	*0.77*	*1847*
Weighted avg	*0.77*	*0.77*	*0.77*	*1847*

KNN				
Lớp	*Precision*	*Recall*	*F1-score*	*Support*
0	*0.76*	*0.72*	*0.74*	*1026*
1	*0.67*	*0.72*	*0.69*	*821*
Accuracy			*0.72*	*1847*
Macro avg	*0.72*	*0.72*	*0.72*	*1847*
Weighted avg	*0.72*	*0.72*	*0.72*	*1847*

SVM				
Lớp	*Precision*	*Recall*	*F1-score*	*Support*
0	*0.80*	*0.77*	*0.79*	*1026*
1	*0.73*	*0.76*	*0.74*	*821*
Accuracy			*0.77*	*1847*
Macro avg	*0.76*	*0.77*	*0.76*	*1847*
Weighted avg	*0.77*	*0.77*	*0.77*	*1847*

Fig 3: Trained model accuracy

3. BMP180 Hardware Programming

On the hardware side, we connect a BMP180 sensor to a Raspberry Pi 3B+ via the I²C interface, ensuring proper wiring of SDA and SCL lines, 3.3 V power, and ground. To convert raw ADC values into meaningful measurements, our driver first reads the set of calibration coefficients (AC1 through AC6, B1, B2, MB, MC, MD) directly from the BMP180's registers as specified in Bosch's datasheet. Then, using Bosch's provided formulas, the driver calculates the true temperature in degrees Celsius and the absolute pressure in Pascals. Specifically, we compute an intermediate "uncompensated temperature" value, apply a series of integer and floating point operations to derive B5, and then calculate the final temperature. For pressure, we follow a longer sequence: read the uncompensated pressure value, compute intermediate variables B6, X1–X3, B3, B4, B7, and then derive the compensated pressure. Once calibrated, the driver enters a continuous loop triggering a

measurement every 0.5 seconds (or another configurable interval) reading the raw registers, performing the compensation calculations, and then writing a new row to a local CSV log. Each log entry includes a UTC timestamp, calculated temperature, calculated pressure, and any error codes if the reading failed. This CSV file serves both as a backup in case of network disruptions and as a historical record for later analysis or model retraining.

BMP180 Reg Adr			
Parameter	***MSB***	***LSB***	***Values***
AC1	*0xAA*	*0xAB*	*8492*
AC2	*0xAC*	*0AD*	*-1056*
AC3	*0AE*	*0AF*	*-14273*
AC4	*0B0*	*0B1*	*33682*
AC5	*0B2*	*0B3*	*25835*
AC6	*0B4*	*0B5*	15882
B1	0B6	0B7	6515
B2	0B8	0B9	36
MB	0BA	0BB	-32768
MC	0BC	0BD	-11786
MD	0BE	0BF	15882

Fig 4: Calibration coefficient (each device has its own coefficient)

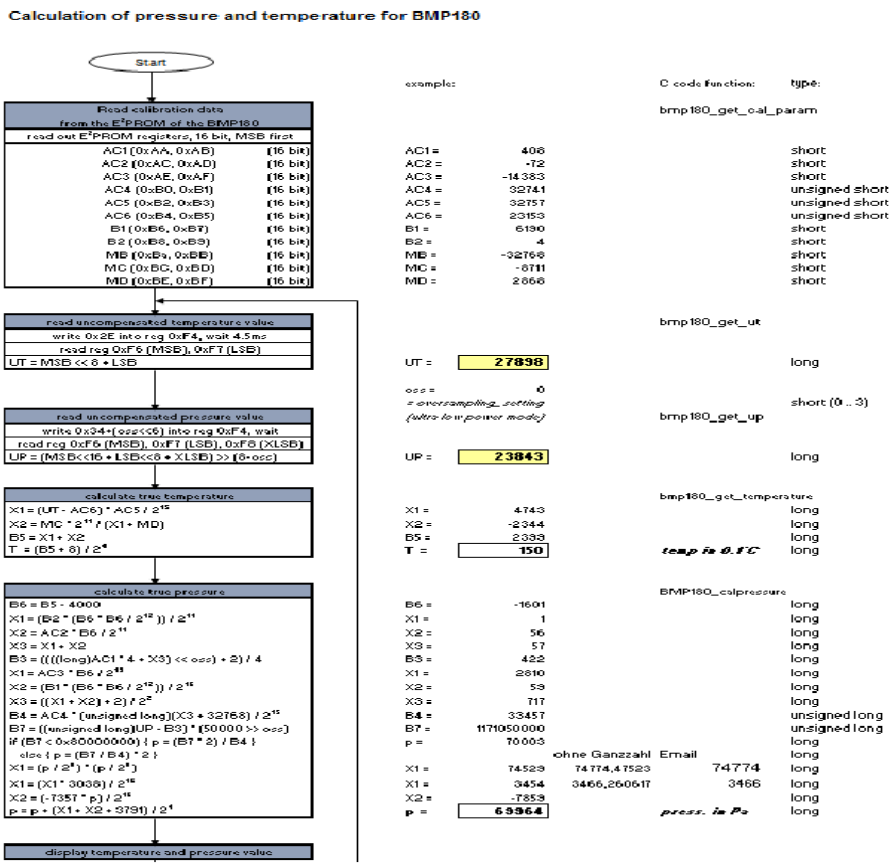

Fig 5: Formula provided by Bosch

4. Web and Mobile Application Development

After hardware integration is complete, we launch a FastAPI based web application directly on the Raspberry Pi. The web app defines several REpresentational State Transfer endpoints: one returns the latest temperature and pressure readings, another returns the past 24 hours of logged data as JSON (for frontend charting), and a third accepts current temperature and pressure values to return a prediction from our preloaded voting ensemble model. For security, we implement a simple login system: users register with a username and password stored in a local database; upon successful authentication, a session cookie grants access to the dashboard. The dashboard's frontend uses JavaScript (Chart.js) to display interactive line charts one for pressure over time and another for temperature over time along with a table showing recent sensor logs and a panel that visualizes the machine learning driven forecast. Simultaneously, we develop a mobile application using Flutter that mirrors the web interface's functionality but omits the login requirement: the user provides the URL (or IP address) of any Raspberry Pi node, and the app fetches the same endpoints to show live readings, history charts, and forecast results. Crucially, because the mobile app communicates over HTTP/HTTPS, it can access any node regardless of local network so long as that Pi has an externally reachable address or is behind a properly configured NAT with port forwarding. Together, the web and mobile components complete the system by presenting real time sensor data and model predictions to end users anywhere, on any device.

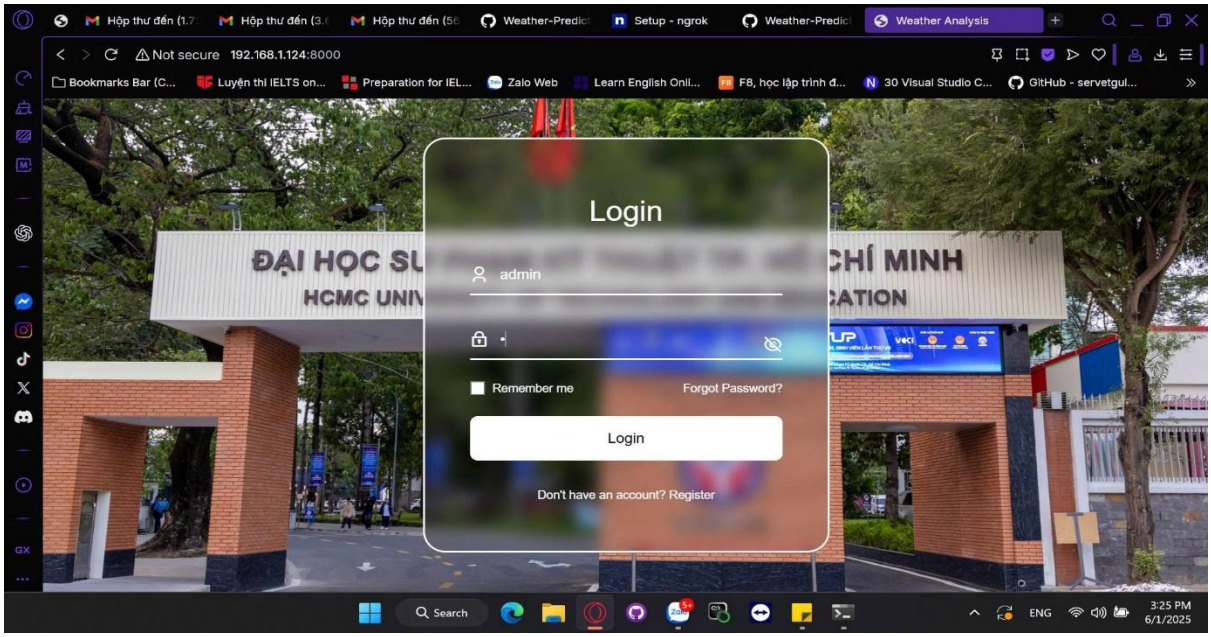

Fig 6: Login page(WEB)

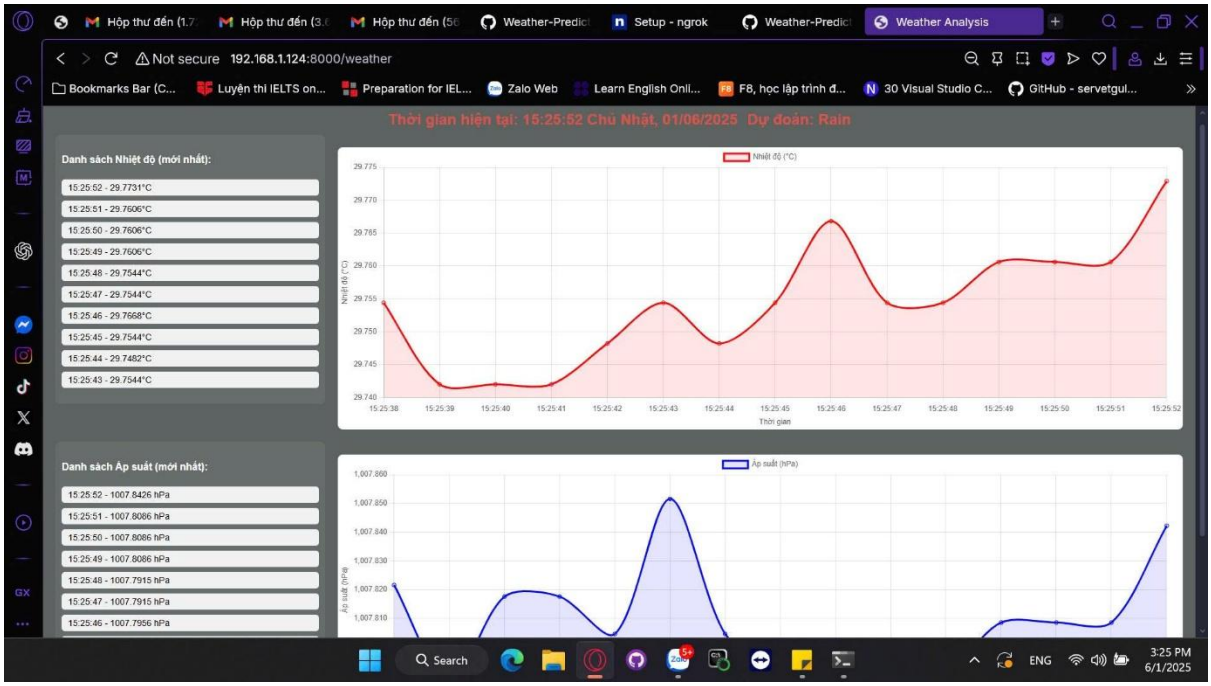

Fig 7: Weather analysis page(WEB)

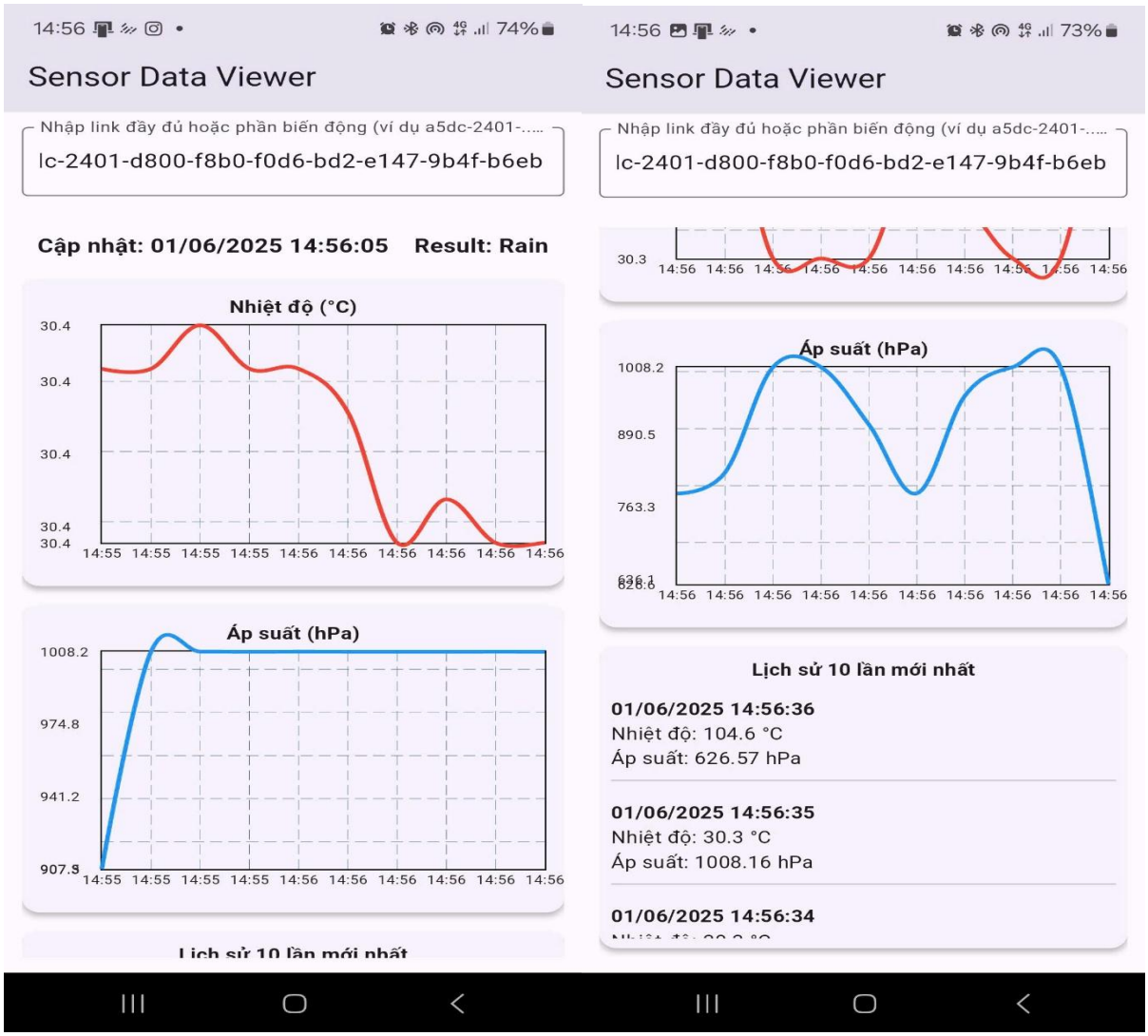

Fig 8: UI(APP)

III. Block Diagram and Circuit Schematic

1. Overall System Block Diagram

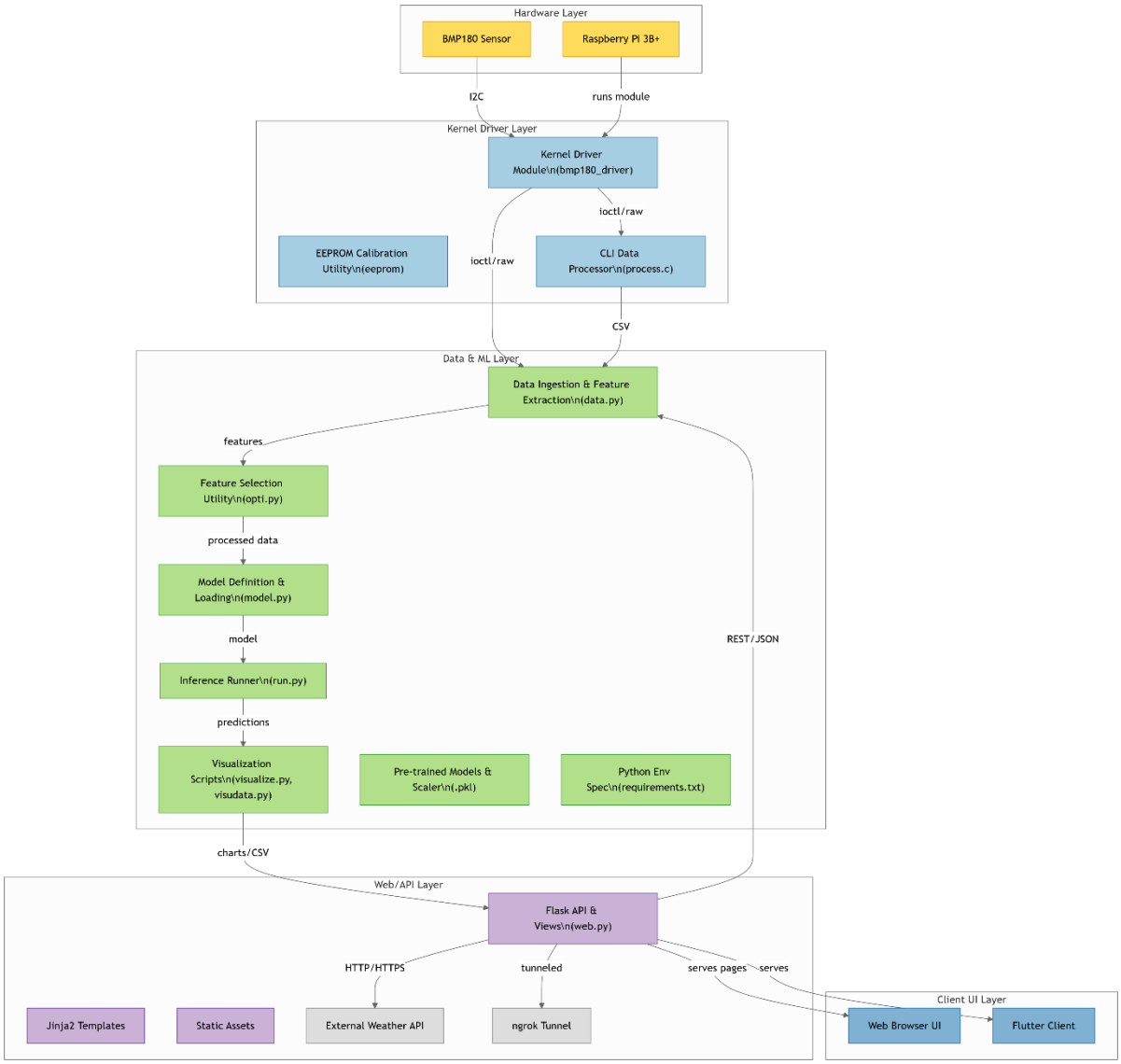

Fig 9: Pipeline

2. BMP180 Circuit Schematic

					Pi 3B+					
BCM	***wPi***	***Name***	***Mode***	***V***	***Physical***	***V***	***Mode***	***Name***	***wPi***	***BCM***
		3.3V			*1*					
2	*8*	*SDA.1*	*ALT0*	*1*	*3*					
3	*9*	*SCL.1*	*ALT0*	*1*	*5*	*14*		*0V*		

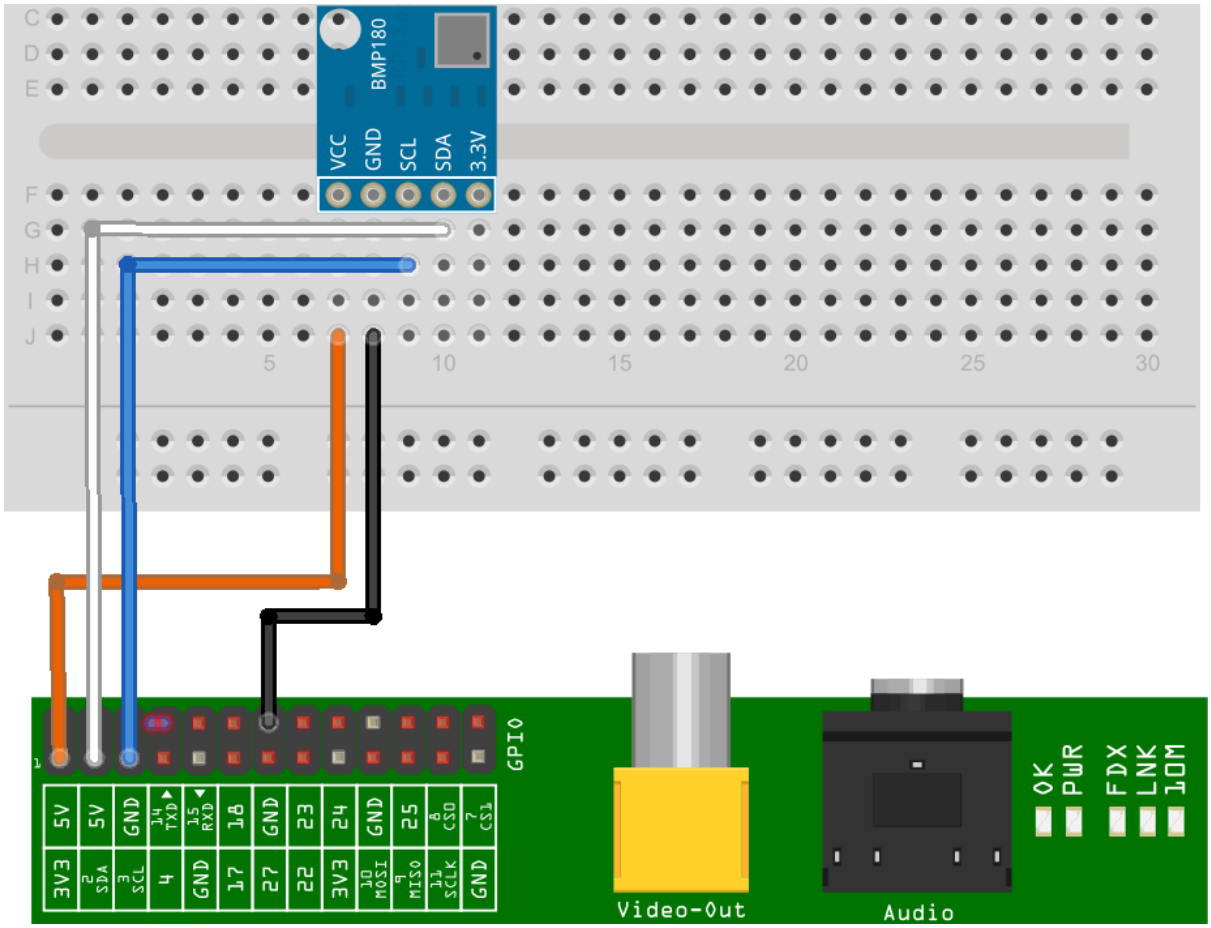

Fig 10: Wiring diagram

IV. Conclusion

1. Summary

In this project, we developed an integrated weather monitoring and prediction system that combines BMP180 sensor data with machine learning models trained on historical data from WorldWeatherOnline. We began by collecting and preprocessing temperature and pressure records, then trained eight distinct algorithms Decision Tree, Random Forest, Gradient Boosting, K-Nearest Neighbors, Support Vector Machine, Logistic Regression, Naive Bayes, and a Neural Network tuning each via cross validation. By ensembling these models through a voting classifier, we obtained more robust short term forecasts than any single algorithm could provide. On the hardware side, we implemented a BMP180 driver on a Raspberry Pi 3B+ to read calibration coefficients and convert raw ADC values into accurate temperature and pressure readings, continuously logging each measurement to a local CSV file. Finally, we built a FastAPI web application and a companion mobile client that serve real time sensor data, historical trends, and machine learning based forecasts. Users can access dashboards secured by a simple login on the web or by entering a Pi node's address on mobile to visualize interactive time series charts and receive confidence scored predictions. Altogether, these components form a seamless, end to end solution that enables accessible, localized weather insights without relying on expensive or centralized meteorological infrastructure.

2. Advantages

One of the system's primary strengths is its cost effectiveness: the combination of a low cost BMP180 sensor and an inexpensive Raspberry Pi 3B+ offers a budget friendly alternative to conventional weather stations. Because our machine learning models use only two input features (temperature and pressure), both data collection and preprocessing remain lightweight, allowing rapid model training and inference even on the Pi itself. The voting ensemble further enhances forecast accuracy by leveraging the complementary strengths of multiple algorithms, resulting in more reliable predictions. From a deployment standpoint, hosting the FastAPI server directly on the Raspberry Pi eliminates dependency on external cloud services; users simply point their web browser or mobile app to the Pi's address to retrieve data and forecasts. This design also makes the system highly scalable: additional Pi + BMP180 nodes can be installed at different locations, and each node operates independently. Finally, providing both web and mobile interfaces maximizes accessibility users can view real time readings and forecasts from any device, over any network, without needing to be on the same local LAN.

3. Limitations

Despite its benefits, the system has several notable limitations. First, by relying solely on temperature and pressure as input variables, our forecasts capture only a subset of the meteorological picture; factors such as humidity, wind speed, and solar radiation are not considered, which can reduce the models' predictive accuracy in certain conditions. Second, the forecasting task is limited to binary classification over a short term horizon, meaning that precise rainfall amounts, cloud cover, or longer range forecasts are beyond its capabilities. Hardware wise, the BMP180 itself has inherent accuracy constraints, and our implementation does not dynamically compensate for sensor drift over time. Because both the sensor driver and web server run on the same Raspberry Pi, resource contention may arise under heavy load, potentially increasing latency or causing occasional data service interruptions. Additionally, the system assumes a stable Internet connection; there is no offline buffering or deferred synchronization, so a network outage effectively halts remote access and data backups. Lastly, user authentication is limited to a basic username/password schema, lacking multi factor or role based controls, which may not meet rigorous security requirements in more demanding environments.

4. Future Improvements

Many opportunities exist to enhance both the hardware and software components. Incorporating additional environmental sensors such as a humidity module, an anemometer for wind speed, or a light sensor would provide richer feature sets and potentially improve forecast precision. On the machine learning front, extending the prediction horizon to multi class outputs or regression based rainfall amount estimates could make the system more versatile. Implementing adaptive model retraining where the system periodically retrains on newly collected sensor data would allow models to adjust to evolving local weather patterns. To address reliability, migrating the FastAPI backend to a lightweight cloud service or edge

cluster could offload processing from the Raspberry Pi and introduce automatic failover. Adding offline buffering and deferred synchronization would enable uninterrupted data collection during transient network outages. Enhancing security with multi factor authentication and encrypted data transmission would strengthen user privacy. Finally, aggregating data from multiple Pi + BMP180 nodes into a central dashboard could facilitate microclimate mapping and community wide weather analysis, laying the groundwork for more sophisticated applications such as crop recommendation or early storm warning systems.

Link Github Project :

https://github.com/aysinemu/Weather-Prediction-Using-Machine-Learning-Based-on-BMP180-Sensor-Data