PHASE 3 DEVELOPING PART

An IoT-Based Devoloping Air Quality Monitoring Platform

The IoT-based indoor air quality monitoring platform is primarily divided into the Smart-Air and the web server. The set of sensing devices necessary to collect the data to analyze air quality comprised a laser dust sensor, a CO sensor, a CO_2 sensor, a VOC sensor, and a temperature and humidity sensor. Each device transmitted data to the web server via the LTE module to determine air quality and visualize the result.

Furthermore, cloud computing technology was integrated with a web server. The main benefits of the cloud computing-based web server are faster speed, flexibility, and greater accessibility. The web server provided faster and more flexible data processing functions with a large amount of data, which is essential for a monitoring platform. The cloud computing-based web server is easily accessible through most browsers to allow ubiquitous monitoring.

In this study, Amazon Web Services (AWS) was used as the web server to analyze, visualize, and present the data collected from Smart-Air. Also, the web server provides a database to store that data in the cloud. Furthermore, a mobile application was developed for the system to visualize air quality with the web server "anywhere, anytime" in real time.

(i)We propose the use of the Smart-Air for the precise monitoring of indoor air quality

(ii)We propose the utilization of an IoT for efficient monitoring of real-time data

(iii)We propose the adoption of cloud computing for real-time analysis of indoor air quality

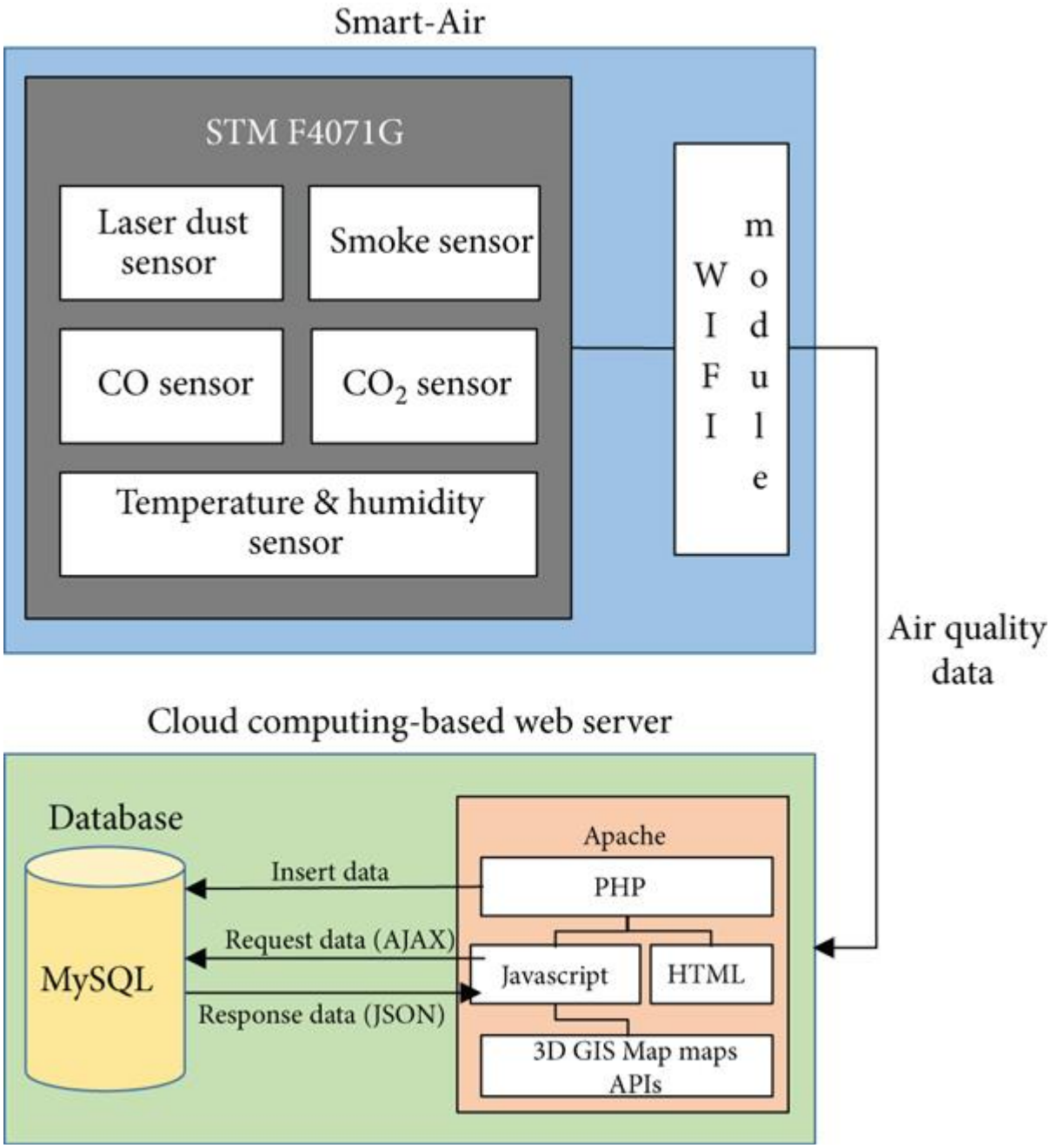
(iv)We originally developed a mobile application to make the proposed IoT system with features of anytime, anywhere

(v)The device has been tested for reliability of the data and the platform has been implemented in a building to test its feasibility

2. Smart-Air

An accurate data measurement of indoor air quality is the most important factor for the platform. Thus, Smart-Air was developed to collect accurate and reliable data for indoor air quality monitoring. Because the monitoring area is not constant, the device was designed to be easily customized to an environment by using an expandable interface. Thus, various types of sensors can be installed or adjusted based on the environment. Also, a Long-Term Evolution (LTE) modem is mounted in the device to transmit detected data directly to the web server for classifying and visualizing air quality.

For most IoT platforms, gateway or data loggers are installed to gather and transmit data wirelessly to the web server. However, in this study, a microcontroller was installed in the device to gather the data from the sensors and transmit it to the web server using the LTE modem, eliminating the need for a gateway and a data logger.



DEVOLOPING PYTHON PROGRAMMING:

 # importing Randomforest

from sklearn.ensemble import AdaBoostRegressor

```
from sklearn.ensemble import RandomForestRegressor
# creating model
m1 = RandomForestRegressor()
 # separating class label and other attributes
train1 = train.drop(['air_quality_index'], axis=1)
target = train['air_quality_index']
 # Fitting the model
m1.fit(train1, target)
'''RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                max_depth=None, max_features='auto', max_leaf_nodes=None,
                max_samples=None, min_impurity_decrease=0.0,
                min_impurity_split=None, min_samples_leaf=1,
                min_samples_split=2, min_weight_fraction_leaf=0.0,
                n_estimators=100, n_jobs=None, oob_score=False,
                random_state=None, verbose=0, warm_start=False)'''
 # calculating the score and the score is  97.96360799890066%
m1.score(train1, target) * 100
 # predicting the model with other values (testing the data)
# so AQI is 123.71
m1.predict([[123, 45, 67, 34, 5, 0, 23]])
 # Adaboost model
# importing module
 # defining model
 m2 = AdaBoostRegressor()
# Fitting the model
m2.fit(train1, target)
```

```
'''AdaBoostRegressor(base_estimator=None, learning_rate=1.0, loss='linear',
                    n_estimators=50, random_state=None)'''
 # calculating the score and the score is  96.15377360010211%
m2.score(train1, target)*100
 # predicting the model with other values (testing the data)
# so AQI is 94.42105263
m2.predict([[123, 45, 67, 34, 5, 0, 23]])
```