Air Quality Monitoring System Project Report

1. Introduction

Overview

The Air Quality Monitoring System is designed to measure environmental parameters such as temperature, humidity, and air quality using sensors. The data is collected by an ESP32 microcontroller and sent to Firebase for storage and analysis. A React Native application provides real-time visualization and historical data access.

Problem Statement

Air pollution is a significant environmental issue affecting human health and the ecosystem. Monitoring air quality in real-time can help in taking timely actions to mitigate its effects.

Objectives

- To develop a system that measures and reports air quality and environmental conditions.
- To provide real-time data visualization through a mobile application.
- To store historical data for analysis and trend observation.

Scope

The project focuses on using low-cost sensors and microcontrollers to create an accessible air quality monitoring solution. It includes both hardware and software components, with potential for future enhancements.

2. System Model

Design and Architecture

The system consists of the following components:

- **ESP32 Microcontroller**: Acts as the central processing unit, collecting data from sensors and communicating with Firebase.
- **DHT11 Sensor**: Measures temperature and humidity.
- **MQ135 Gas Sensor**: Measures air quality.
- **Firebase**: Serves as the backend for data storage.
- **React Native App**: Provides a user interface for data visualization.

Component Interactions

- The ESP32 microcontroller reads data from the DHT11 and MQ135 sensors.
- Data is sent to Firebase using Wi-Fi connectivity.
- The React Native app retrieves data from Firebase and displays it to the user.

3. Hardware Design

Components Used

- **ESP32 Development Board**: Chosen for its Wi-Fi capabilities and ease of use.
- **DHT11 Sensor**: Selected for its simplicity and reliability in measuring temperature and humidity.
- **MQ135 Gas Sensor**: Used for detecting air quality levels.

Configuration

- The DHT11 sensor is connected to GPIO 32 on the ESP32.
- The MQ135 sensor is connected to GPIO 33 on the ESP32.
- Power and ground connections are made to the respective pins on the ESP32.

4. Software Design

Architecture

- The software is divided into two main parts: the firmware for the ESP32 and the React Native application.
- The ESP32 firmware is responsible for sensor data acquisition and communication with Firebase.
- The React Native app handles data retrieval and user interface rendering.

Algorithms

- Data from the sensors is read at regular intervals.
- The ESP32 processes the data and formats it for Firebase storage.
- The React Native app calculates air quality percentages and updates the UI accordingly.

5. Results and Discussion

Outcomes

- Successful real-time monitoring of temperature, humidity, and air quality.
- User-friendly mobile application for data visualization.

Analysis of Results

- The system provides accurate and timely data, which can be used for environmental monitoring and decision-making.
- Firebase integration allows for scalable data storage and retrieval.

Challenges Faced

- Calibration of the MQ135 sensor for accurate air quality readings.
- Ensuring reliable Wi-Fi connectivity in varying environments.

Future Scope

- Integration of additional sensors for more comprehensive environmental monitoring.
- Implementation of TinyML models for predictive analytics.
- Expansion of the mobile app to include alerts and notifications.

Aslı Ayşe Şahin

2011051022