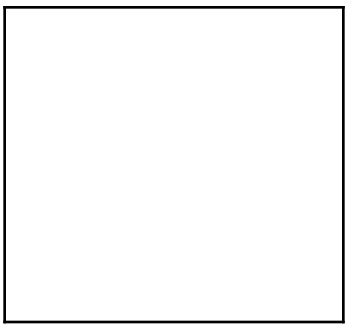
TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES
1338 ARLEGUI ST., QUIAPO, MANILA, METRO MANILA, PHILIPPINES

AIRAWARE: IoT-BASED GAS SENSOR AND AIR QUALITY MONITORING SYSTEM FOR HOUSEHOLDS



SUBMITTED TO:
ENGR. MALBOG, MON ARJAY

SUBMITTED BY:
CATAMBAY, JEANETTE
GALIAS, JOHN MARCO
LUMBES, AARON JOHN
MARISCOTES, REINIER
MEJES, JAMES
VAQUEZ, RAMON

CPE400- CPE41S
12/14/2023

1.1 Introduction

Every home, which serves as both the center of everyday activities and a meeting spot for meals, frequently has hidden dangers to the security of the family. Gas leakage poses a significant risk to households. According to a 2022 report from the Philippine Gas Institute, there were over 280 residential fires caused by gas leaks in the Philippines in 2018. As reported by the Bureau of Fire Protection (BFP), 104 related fires were recorded. However, gas leaks aren't the most common cause of fires in every home.

Apart from the risk of gas leaks, these incidents also release toxic substances that can seriously affect health. Exposure to these harmful compounds can lead to severe medical problems. Neglecting humidity and temperature control in our households carries its own set of problems. High humidity can foster mold growth, damage appliances, and pose health risks.

To address these concerns, the developers have created AirAware, an innovative IoT-based system designed to monitor gas leakage and air quality in households. Through AirAware's constant monitoring of humidity, gas levels, and air quality, possible threats are promptly detected, enabling homes to take proactive measures to maintain a safe and healthy atmosphere.

1.2 Problem

This project aims to solve the problem of inconsistent energy consumption by monitoring the temperature and humidity of a room and monitoring the gas if there’s a leak that can lead to accidents.

1. Unmonitored gas leaks create a potentially explosive environment, as the accumulated gas can reach flammable concentrations.
2. Gas leaks often release toxic substances that can pose significant health risks to individuals exposed to the leaking gas.
3. Neglecting to monitor humidity and temperature levels can result in inefficient energy usage. Which can lead to increased energy consumption and high utility costs.

1.3 Project Objectives

This project aims to solve the problem by developing AirAware: IoT-based Gas Leakage and Air Quality Monitoring System for Households

1. Develop a Monitoring system using Arduino with DHT11 and Gas Sensors
2. Set up a GUI for the user to monitor the status of temperature, humidity, and gas.
3. Refine the design and usage of AirAware based on the gathered data and continuous testing.

2. Embedded Product Development Life Cycle

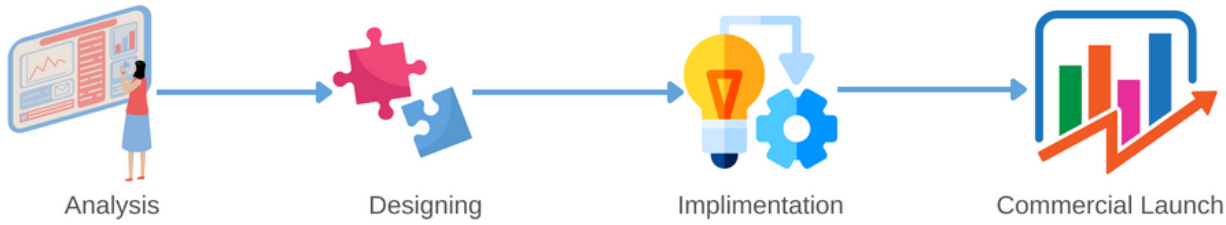


Figure 2. AirAware Embedded Product Development Life Cycle

2.1 Planning and analysis

AirAware's objective is to make homes safer and healthier. It improves with the prevention of gas leaks, which can cause fires and danger to humans. It also monitors air quality and regulates temperature and humidity, allowing families to stay healthy while saving money by using less energy. Users expect it to be simple to use and dependable.

Understanding the effect and relevance of the concerns addressed by AirAware is important: gas leaks pose urgent safety and health risks, while poor humidity and temperature regulation harm health, increase energy use, and destroy appliances. Reactive responses increase risks and expenses in the absence of monitoring solutions. Technical studies of sensor technology, data proprocessing,d financial analysis for budgeting are all part of determining its feasibility. Exploring existing IoT monitoring systems, evaluating their strengths, shortcomings, and gaps, and aligning AirAware with industry standards and best practices are all part of the research process.

2.2 Designing

2.2.1 System Architecture

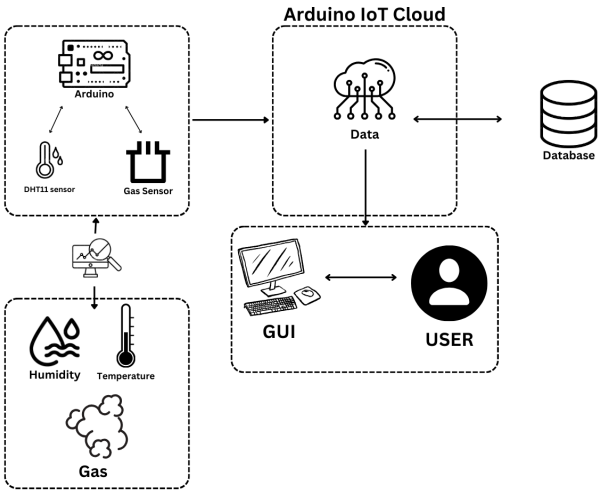


Figure 2.2: Airaware System Architecture

Figure 1 represents the system architecture of the proposed project. It has an Arduino connected to a dht11 sensor that gets the humidity and temperature, paired with a gas sensor that detects if there is a gas leak. The Arduino sends the data to the IoT cloud and stores it in a local database. The GUI from Arduino IoT Cloud can be accessed by the user and monitors the current status. It will also show the necessary information that the user needs.

2.2.3 Tools and Technologies

Figure 2.2.8 Arduino IDE

Figure 2.2. Shows the Arduino IDE, The Arduino IDE, or Integrated Development Environment, is a software platform designed for programming Arduino microcontrollers. It offers a user-friendly interface for writing, compiling, and uploading code to Arduino boards. Widely used by hobbyists and professionals alike, the IDE supports the Arduino programming language, making it accessible for a broad range of users in the development of electronic projects.

Figure 2.2.9 MySQL

Figure 2.2.9 shows MySQL an open-source relational database management system (RDBMS) that is widely used for managing and organizing data. Developed by Oracle Corporation, MySQL uses a structured query language (SQL) for defining and manipulating the data within its databases. It is a popular choice for web applications and various software development projects, offering scalability, reliability, and ease of use in handling large datasets and complex queries.

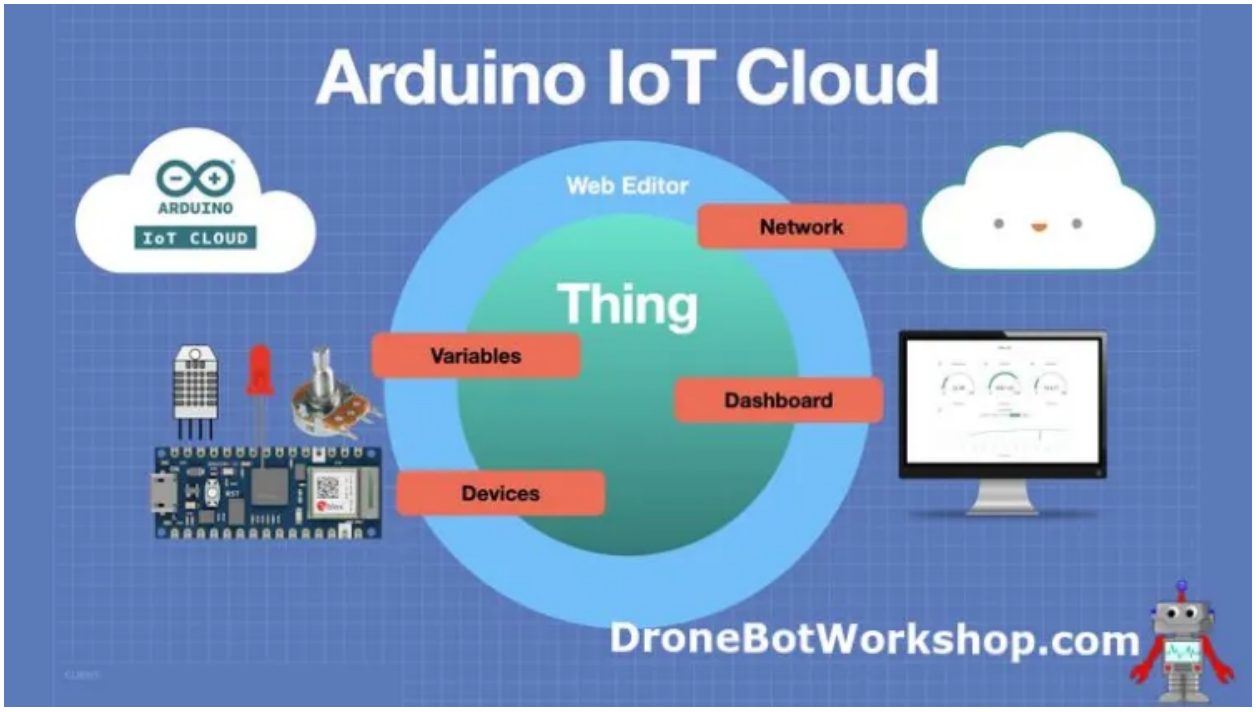


Figure 2.2.10 Arduino IoT Cloud

Figure 2.2.10 shows the Arduino IoT Cloud. The Arduino IoT Cloud is a user-friendly platform that makes it easier to construct IoT applications by making it simple for Arduino devices to connect to the internet. It provides a dashboard interface for remote control and monitoring, enabling users to view sensor data, operate actuators, and maintain the status of devices. It simplifies the process of creating Internet of Things apps and makes it accessible to users with different levels of coding experience thanks to its code generating features and interaction with different Arduino boards. The platform offers a complete solution for developing, connecting, and administering Arduino-based Internet of Things (IoT) applications, with a focus on data transmission security and expanded user access to APIs.

Hardware Components

Features of the Arduino UNO board:

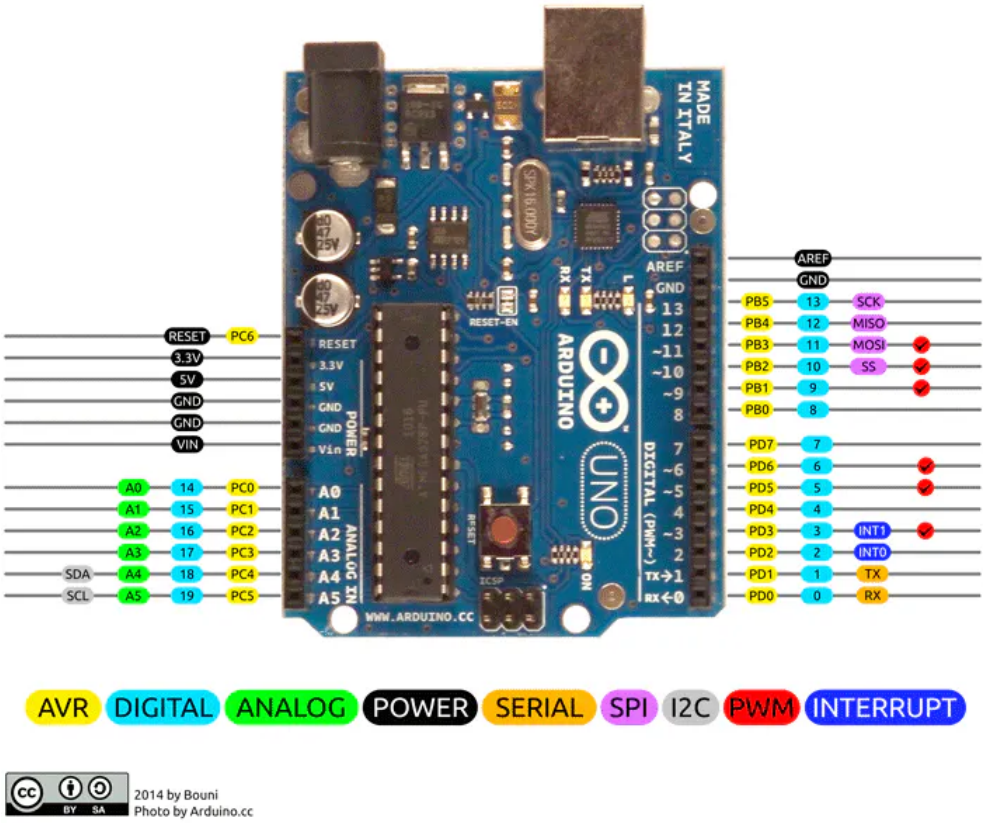


Figure 2.2.3 Arduino Board

Figure 3.66 shows the components of the Arduino Uno, a microcontroller board that follows open-source hardware and software standards, allowing for design and programming flexibility and modification.

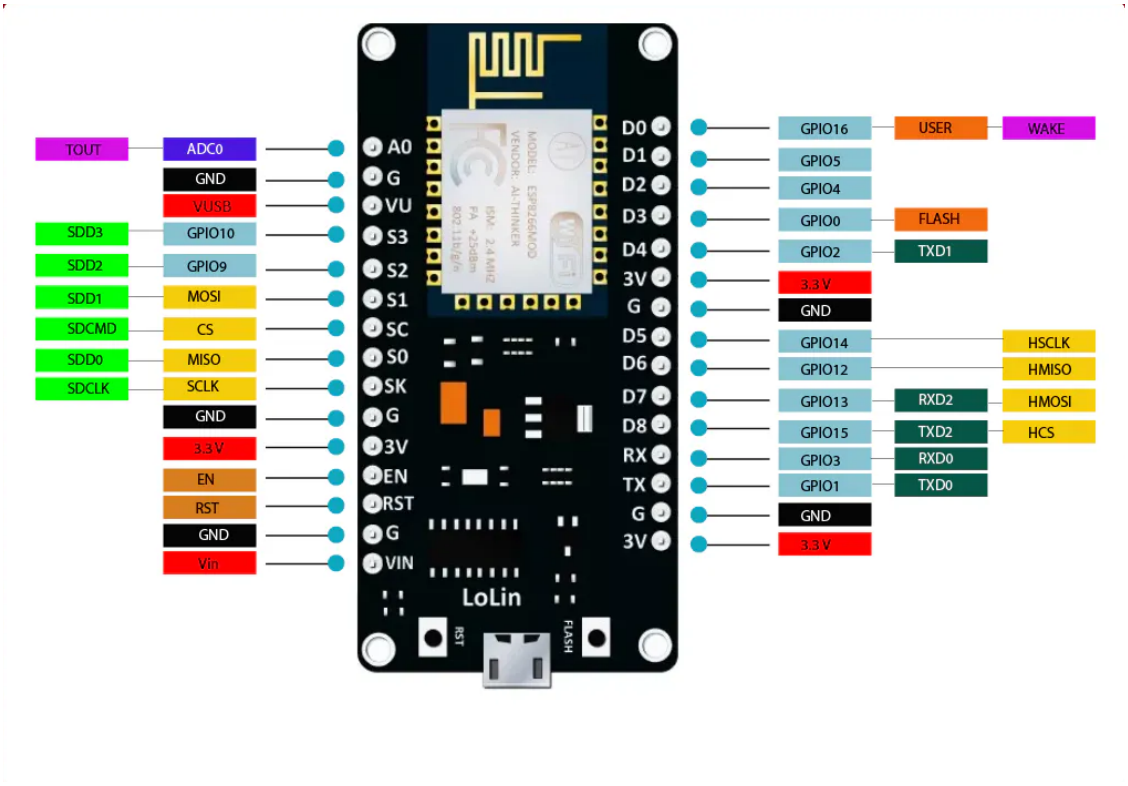


Figure 2.2.3 NodeMCU

Figure 2.2.3 shows the nodeMCU, NodeMCU is an open-source firmware and development kit based on the ESP8266 Wi-Fi module, using Lua scripting language to simplify Internet of Things (IoT) application development with easy programming, GPIO support, and a USB-to-serial interface.

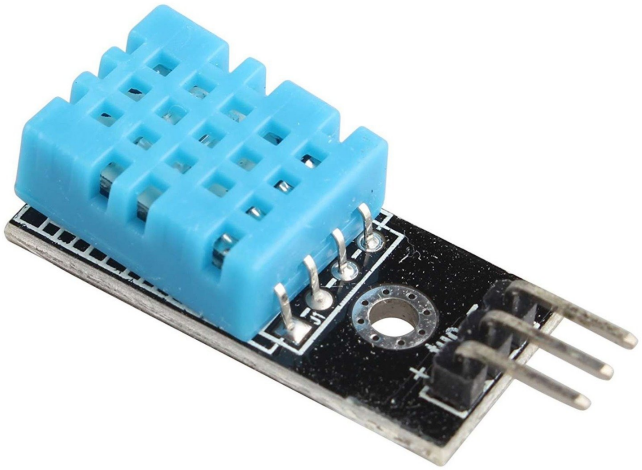


Figure 2.2.4 DHT11 Module

Figure 2.2.4 shows the DHT11 Module, The DHT11 module is an affordable sensor designed for measuring temperature and humidity in electronic projects. It utilizes a digital sensor with three pins for power, ground, and a digital signal, making it easy to interface with microcontrollers like Arduino. The module outputs data in a standardized format, providing accurate temperature and humidity readings within a specified range. Widely used in

applications such as weather stations and home automation systems, the DHT11 is a popular choice for environmental monitoring in diverse projects.

Figure 2.2.6 MQ135 Gas Sensor

Figure 2.2.6 shows the MQ135 gas sensor. The MQ135 gas sensor is widely used to detect gases like carbon dioxide, ammonia, and methane in the air. Operating on the principle of resistance changes, it provides an analog output proportional to the concentration of different gases. With a straightforward interface, the MQ135 is commonly integrated into projects for air quality monitoring and gas leakage detection. Its applications span industrial and domestic settings, contributing to safety and environmental sensing.

Figure 2.2.7 18650 lithium ion battery

Figure 2.2.7 shows the 1650 lithium-ion batter, The 18650 lithium-ion battery is a rechargeable cylindrical cell with standard dimensions of 18mm in diameter and 65mm in length. Renowned for its high energy density and long cycle life, it is commonly used in devices such as laptops, power tools, and electric vehicles. With a nominal voltage of approximately 3.7 volts, these batteries are often configured in series or parallel to meet specific voltage and capacity needs.

In design process, the developers finalized the approach when solving the problem. The components that will be used were, DHT11 sensor for capturing the temperature and humidity, MQ135 Gas Sensor for detecting gas leaks. It will be connected to an Arduino to act as the brain of the system. The arduino will send the data to the Arduino IoT cloud and will be stored in a local database using MySQL. A GUI will also be provided so the user can monitor the status. Arduino IDE was used to create the program that will be embedded in the arduino.

2.3 Implementation

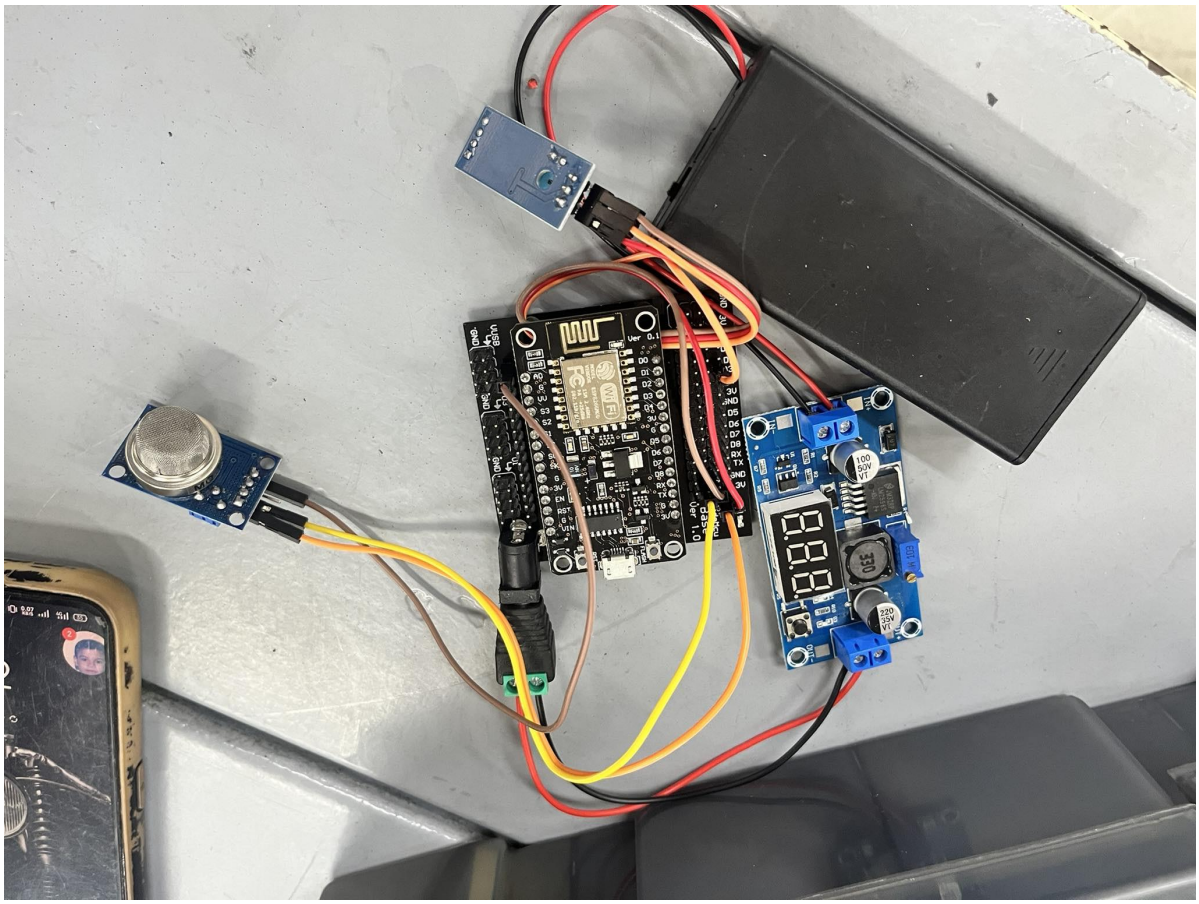


Figure 2.3 Prototype

Prototyping is an essential milestone in the development of AirAware since it allows for early validation and testing. These models serve as tangible representations of the intended user interface and monitoring system, making it easier to evaluate concepts and functionality. Technical and design faults are found and fixed through extensive testing, guaranteeing that the finished product successfully achieves its objectives. These prototypes facilitate continuous modifications, allowing for the enhancement of both the usability of the user interface and the essential functionalities of the monitoring system.

2.4 Commercial launch

AirAware represents innovation at its core. Leveraging cutting-edge technology and design, our team has crafted a prototype that showcases the incredible potential of this game-changing system. Our prototype has undergone testing, demonstrating its prowess in early detection of potential hazards, maintaining excellent air quality, and optimizing environmental conditions within homes.

3. Testing and Result

https://drive.google.com/drive/u/0/folders/1tq9LGDz-tGLiEoiMN68hx81L_ncLC3yB

4.1 Picture of the circuit/device

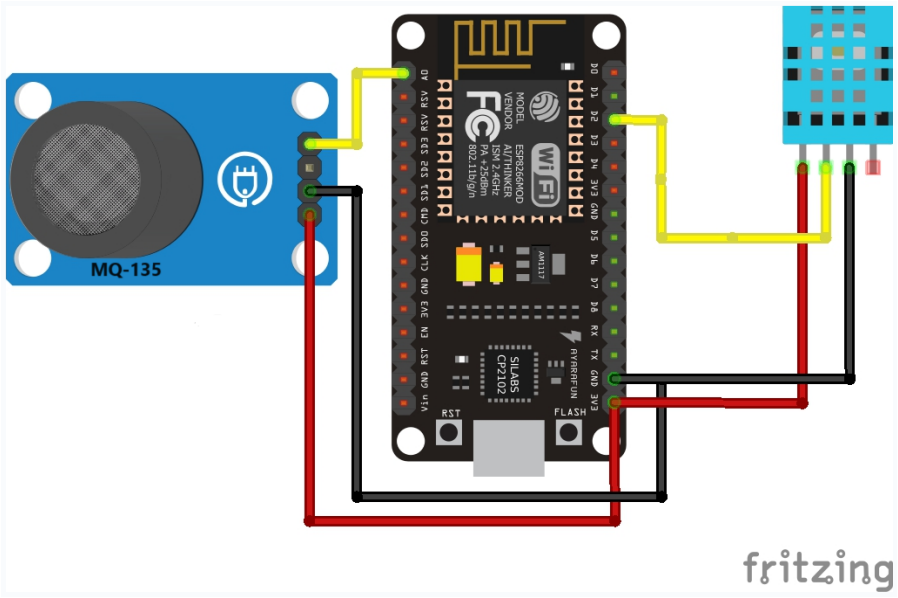


Figure 4.1 : AirAware circuit simulation

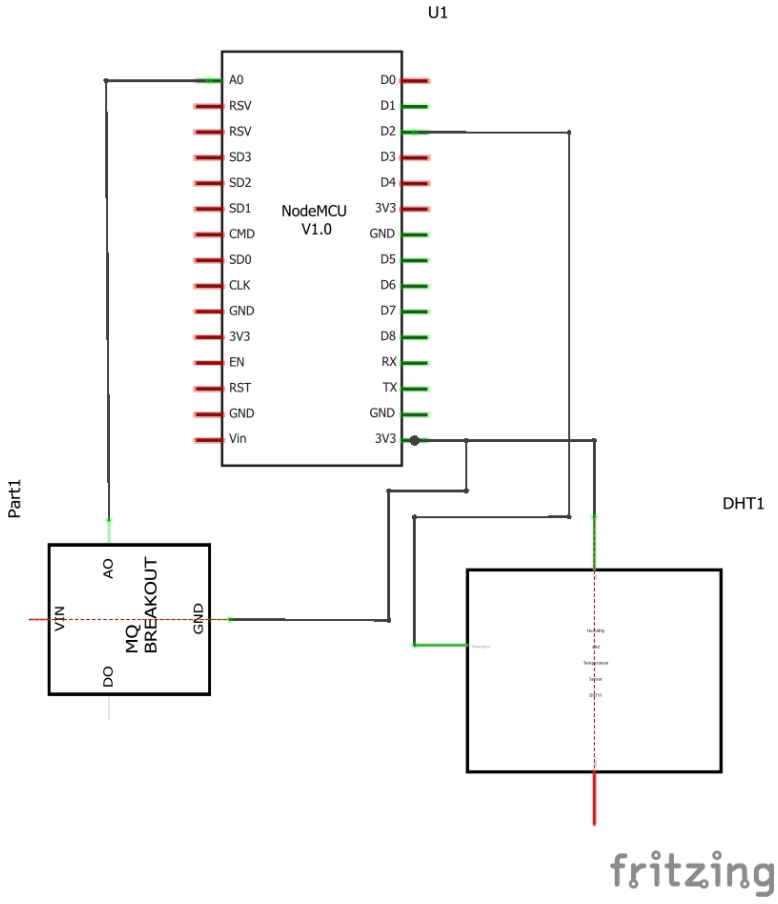


Figure 4.2: AirAware Schematic diagram

Utilizing Fritzing, an open-source electronic simulator. The developers gain access to a wide range of features. This enables them to make schematic designs and experiment with circuit simulations. These features are essential to the design process since they enable developers to see projects in greater detail and identify any problems with the design before the prototype is put into production.

5. Source code & Output

API:

```
const express = require('express');

[humidity, gas, temperature, currentTime], (err, result) => {
if (err) {
console.error("Error inserting into sensor: ", err); res.sendStatus(500);
return;
}

// New row successfully inserted
res.json({ message: "New sensor data inserted successfully", id: result.insertId });
}
);
});

app.get('/', (req, res) => { res.send('API is running');
});


app.get('/get_data', (req, res) => {
const sql = `SELECT * FROM system_control`; pool.query(sql, (err, result) => {
if (err) {
console.error('Error fetching data from database:', err); res.status(500).send('Error fetching
data from database');
} else {
console.log('Data fetched successfully from system_control

table');

}
});
});


res.send(result);


app.get('/TSValue', async (req, res) => { try {
const response = await
```

```
axios.get('https://api.thingspeak.com/channels/2376161/feeds.json?results= 2');
const feeds = response.data.feeds;

const temp = feeds[0].field1; const humidity = feeds[0].field2; const gas = feeds[0].field3;
console.log("Filtered: ",temp, humidity, gas) const currentTime = Date.now()

pool.query(
"INSERT INTO system_control (Humidity, Gas, Temperature, Time) VALUES (?, ?, ?, ?)",
[humidity, gas, temp, currentTime], (err, result) => {
if (err) {
console.error("Error inserting into sensor: ", err); res.sendStatus(500);
return;
}
res.json({ message: "New sensor data inserted successfully", id: result.insertId });
}
);
} catch (error) {
console.error(`Error fetching data: ${error}`); res.status(500).send('Error fetching data');
}
});


cron.schedule('10 * * * * *', async () => { try {
const response = await
axios.get('https://api.thingspeak.com/channels/2376161/feeds.json?results= 2');
const feeds = response.data.feeds; const temp = feeds[0].field1; const humidity =
feeds[0].field2; const gas = feeds[0].field3;
console.log("Filtered: ",temp, humidity, gas) pool.query(

"INSERT INTO system_control (Humidity, Gas, Temperature) VALUES (?, ?, ?)",
[humidity, gas, temp], (err, result) => {
if (err) {
console.error("Error inserting into sensor: ", err); return;
}
```

Source Code:

Arduino IDE:

```
#include <ESP8266HTTPClient.h>


//Air Quality Monitoring

#include <DHT.h>       // Including library for dht #include "MQ135.h"
#include <ESP8266WiFi.h> #include <LiquidCrystal_I2C.h>

String apiKey = "6BCNMCT2TOOKVNFE";   //          Enter your Write API key from
```

ThingSpeak

```
const char *ssid = "reinier";    // replace with your wifi ssid and wpa2 key
const char *pass =      "andotdot";
const char* server = "api.thingspeak.com";

const int sensorPin= 0; int air_quality;

LiquidCrystal_I2C lcd(0x27, 16, 2);

#define DHTPIN 2

DHT dht(DHTPIN, DHT11);
WiFiClient client; void setup()
{
Serial.begin(9600);

delay(10); dht.begin();
lcd.begin(); // Initialize the LCD lcd.backlight();
Serial.println("Connecting to "); Serial.println(ssid);


WiFi.begin(ssid, pass);

while (WiFi.status() != WL_CONNECTED)
{
delay(500); Serial.print(".");
}
Serial.println(""); Serial.println("WiFi connected");


}




void loop()
{

float h = dht.readHumidity(); float t = dht.readTemperature();
//int gasValue = analogRead(gas); MQ135 gasSensor = MQ135(A0); air_quality =
gasSensor.getPPM();


lcd.setCursor(0, 0); // Set the cursor to the top left of the LCD lcd.print("Temp: ");
lcd.print(t);

lcd.setCursor(0, 1); // Set the cursor to the bottom left of the LCD lcd.print("Humid: ");
lcd.print(h);
```

```
lcd.setCursor(10, 0); // Set the cursor to the right side of the LCD lcd.print("PPM: ");
lcd.print(air_quality);
if (client.connect(server,80))   //          "184.106.153.149" or api.thingspeak.com
{

String postStr = apiKey; postStr +="&field1="; postStr += String(t); postStr +="&field2=";
postStr += String(h); postStr +="&field3=";
postStr += String(air_quality); postStr += "\r\n\r\n";









"+apiKey+"\n");

client.print("POST /update HTTP/1.1\n"); client.print("Host: api.thingspeak.com\n");
client.print("Connection: close\n"); client.print("X-THINGSPEAKAPIKEY:

client.print("Content-Type:

application/x-www-form-urlencoded\n");
client.print("Content-Length: "); client.print(postStr.length()); client.print("\n\n");
client.print(postStr);

}
client.stop();

Serial.print("Temperature: "); Serial.print(t);
Serial.print(" degrees Celcius, Humidity: "); Serial.print(h);
//Serial.print("%, Gas Value: ");
//Serial.print(h); Serial.print("%, Air Quality: "); Serial.print(air_quality);
Serial.println("PPM. Send to Thingspeak.");


Serial.println("Waiting...");
```

```
// thingspeak needs minimum 15 sec delay between updates delay(1000);
}
```

Final Output: