An-Najah National University

Faculty of Engineering and Information Technology

Electrical Engineering Department

Presented in partial fulfilment of the requirements for Bachelor degree in (Electrical Engineering)

AIEN
Artificial Intelligence Electrical Nose

Prepared by:

Ameer Abo liel (11925683)
Malath Ghazal (11923666)

Supervised by:
Dr. Khadija Mayalah

بسم الله الرحمن الرحيم

"وَقُلِ ٱعْمَلُوا۟ فَسَيَرَى ٱللَّهُ عَمَلَكُمْ وَرَسُولُهُۥ وَٱلْمُؤْمِنُونَ ۖ وَسَتُرَدُّونَ إِلَىٰ عَٰلِمِ ٱلْغَيْبِ وَٱلشَّهَٰدَةِ فَيُنَبِّئُكُم بِمَا كُنتُمْ تَعْمَلُونَ"

[التوبة : 105]

"يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَات"

[المجادلة : 11]

صدق الله العظيم

Dedication

For Our Palestine …

For Our University …

For Our Teachers …

For Our Family …

We Present This Research …

Disclaimer

Acknowledgements

“We would like to express our heartfelt gratitude to Dr. Kadija Mayalah, our supervisor, for her dedication and continuous support throughout our project. Her valuable scientific guidance has been invaluable to our success. We would also like to thank the academics in the Department of Electrical Engineering for their expertise and assistance, as well as our friends and family for their belief in our abilities. We are deeply grateful for their contributions, support, and belief in our capabilities.”

Table of contents:

List of Figures:

List of Tables:

Abstract

Inspired by the high drug smuggling crime rates all over the world, and due to the fact that current mechanisms of drug detection in airports cause discomfort to many travelers worldwide, the need for new reformed detection mechanisms is constantly growing. Using trained K9 dogs could cause many fearful scenarios that could cause trauma to innocent patients. The new mechanism we are presenting to you shows instantaneous results about the material detected.

This project presents an artificial intelligence electronic nose (AIEN) for detecting and identifying various odors and substances. The AIEN utilizes MQ gas sensors and machine learning algorithms to analyze and classify different volatile organic compounds. A filtration system with ethanol is implemented to ensure accurate results between samples.

The device also incorporates DC and AC fans and motors controlled by a microcontroller to automate the sampling process. Extensive testing produced consistent characteristic odor profiles and plots for different substances like perfumes and alcoholic beverages. The fusion of gas sensor technology with artificial intelligence offers an innovative approach to processing complex olfactory data.

AIEN provides a proof of concept for the capabilities of intelligent odor detection systems in fields ranging from quality control to law enforcement

Chapter one: Introduction

1.0 General background

Smell is one of our five senses and is linked by the olfactory bulb to the limbic system that controls behavior, emotion, and even memory also it could save humans' life by detecting dangerous odors but unfortunately humans can't detect some dangerous odors like certain gasses and in general human can smell different odors but our brains couldn't recognize it and because there is some material are dangerous for humans to smell or touch like drugs from this point we create AIEN the electronic nose can detect any odor using an array of sensors that sent all information to microcontroller and create database that have more than 1000 odor and for sure can detect gas liquid solid materials .

1.1 Problem Statement

The main problem our team try to solve is to create a machine that could detect drugs this will help governments around the world to detect any sus material inter country entrances

The air contains an incalculable number of volatile compounds, which can be detected only by one sense: the smell. There are about 110,000 smells in nature. The human being perceives only about 100-200. Over 800 chemical odors can't be detected most of them are dangerous for the human's body

1.2 Objectives

The main object is to replicate, or even surpass, the olfactory capabilities of humans. By capturing and analyzing the volatile organic compounds (VOCs). We create AIEN to be easy to use and detect dangerous odors and material that is illegal.

1.3 Scope of work

Control system: responsible for controlling the motors Can be done through a touch screen and switches.

AI system: responsible for saving the values of MQ sensors and modifying the values, draws plots for each sample.

Air filtration system: responsible for getting rid of any odor inside AIEN in order to make good conditions for taking the next sample.

Sensing system: responsible for taking the sample's value and sending it to the microcontroller.

1.4 Importance of the work:

Here are some key points to explain the importance of electronic noses:

1. Applications in Industry: Electronic noses are essential in a number of sectors, including the food and beverage, cosmetics, agricultural, and pharmaceutical industries. They can promptly and precisely identify pollutants, product deterioration, or quality problems, preserving product integrity and assuring consumer safety.

2. E-noses are essential to the processes involved in quality control. They are able to detect minute variations in product fragrance profiles, assisting manufacturers in maintaining a consistent level of quality and making sure that goods adhere to predetermined criteria.

3. Early Hazard Detection: In industrial settings, electronic noses can identify potentially harmful substances or gases to workers' health. They offer a technique for early detection of probable leaks or harmful material exposure.

4. E-noses have uses in the field of medical diagnostics, particularly in the detection of diseases by the study of breath. They are able to recognize volatile organic chemicals in exhaled breath that are a sign of specific illnesses like diabetes, lung cancer, or digestive problems.

5. Environmental Monitoring: Another crucial use of electronic noses is for recognizing contaminants and air quality monitoring. They can aid in determining the origins of offensive emissions, monitoring indoor air quality, and determining pollution levels.

6. Waste Management: By identifying scents coming from landfills or waste treatment facilities, e-noses can help with waste management. This aid government agencies in addressing odor-related complaints and enhancing environmental conditions generally.

7. Security and safety: These tools can be used to detect the presence of explosives, dangerous drugs, or illegal substances in a variety of settings, including airports and public areas.

8. Agriculture: Electronic noses are used in agriculture to monitor crop health and evaluate the freshness of product. Based on the volatile molecules released, they may spot disease or physiological changes in plants as well as insect infestations.

9. E-noses are used in the food and beverage industries to examine the fragrance profiles of foods and drinks. This is crucial for the creation of new products, since it ensures consistency and helps to spot any off tastes or spoilage.

10. Research and development: To analyse odor profiles and comprehend the intricate interactions of volatile substances, researchers employ electronic noses. Innovations based on this knowledge might impact everything from materials science to neurology.

Chapter two: Theoretical Background and previous work

It has been proposed for many years to develop artificial systems that can replicate human senses. The human olfactory system, which is extremely sensitive and capable of detecting a wide range of aromas, served as inspiration for researchers when developing electronic noses. Multiple businesses, including food, beverage, environmental monitoring, healthcare, quality control, and even security, could benefit from creating a system that could imitate this skill.

Over the years, there has been a lot of research and development into electronic noses. Some noteworthy developments and uses include:

1. Electronic noses have been used in the food and beverage industries to evaluate the quality and freshness of food products, spot deterioration, and locate contaminants. They may contribute to preserving consistent product quality and assuring security.

2. Environmental Monitoring: By identifying pollutants and dangerous gases, e-noses are used to keep an eye on the quality of the air. To make sure that emission requirements are being followed, they can be utilized in industrial settings.

3. Medical diagnosis: Based on the identification of particular volatile organic compounds (VOCs) linked to ailments like lung cancer, diabetes, and bacterial infections, electronic noses have showed promise in the diagnosis of several medical problems.

4. Wine industry: To aid in quality control and authenticity, e-noses are used to examine and categorize various types of wines based on their scents and odors.

5. E-noses are useful tools in security and law enforcement since they may be used to detect bombs, drugs, and other dangerous chemicals.

6. Biotechnology and Research: Electronic noses are used in scientific research to examine complex combinations, monitor chemical reactions, and comprehend animal communication by fragrance.

Researchers are always investigating new sensor technologies, data analysis methods, and uses for electronic noses as technology develops. The objective is to develop more precise and adaptable tools that, in some situations, can mimic and even outperform the human olfactory system.

Chapter three: Methodology

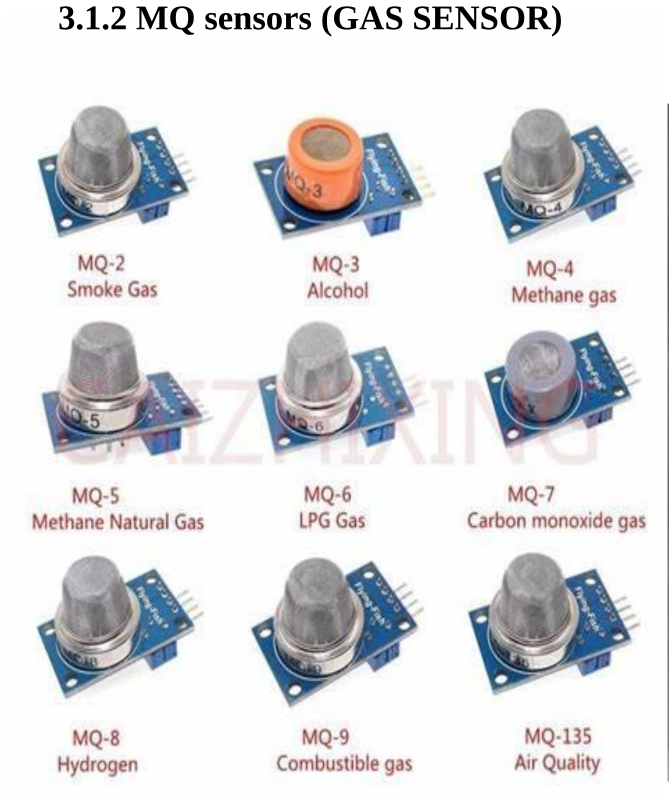
3.1 Project components

3.1.1 Plastic bottle

Figure 0-1 plastic bottle

We choose a plastic bottle with cylindrical shape this type of plastic can handle the drill force when we make holes for MQ sensors and this material is strong enough to handle the weight of 9 MQ sensors, and the main advantage is that because it’s cylindrical shape we can minimize the odors that stuck inside the bottle while we are taking the data and to ensure that the last odor will not affect the new odor after finishing the test.

3.1.2 MQ sensors (GAS SENSOR)



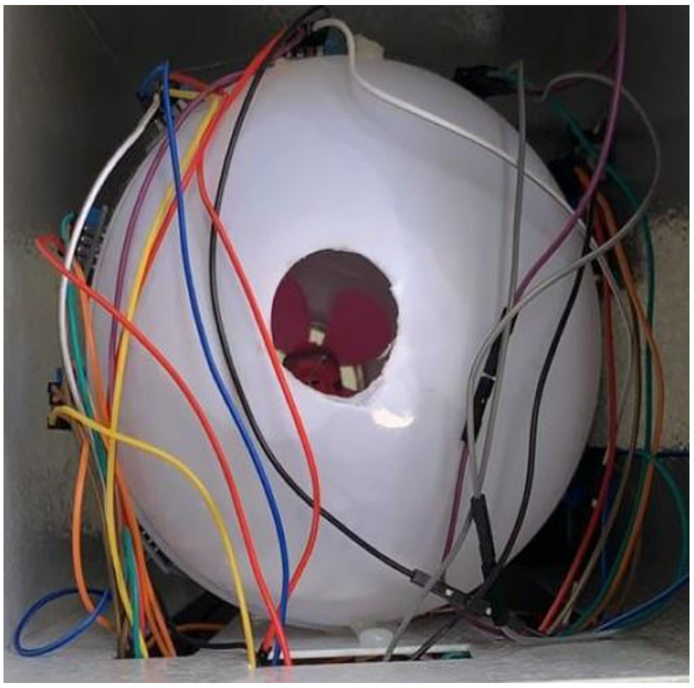


Figure 3-0-2 MQ Sensors

The MQ2 sensor is one of the most widely used in the MQ sensor series. It is a MOS (Metal Oxide Semiconductor) sensor. Metal oxide sensors are also known as Chemoreceptors because sensing is based on the change in resistance of the sensing material when exposed to gasses.

The MQ2 gas sensor operates on 5V DC and consumes approximately 800mW. It can detect LPG, Smoke, Alcohol, Propane, Hydrogen, Methane and Carbon Monoxide concentrations ranging from 200 to 10000 ppm.

The MQ2 is a **heater-driven sensor**. It is therefore covered with two layers of fine stainless-steel mesh known as an “anti-explosion network”. It ensures that the heater element inside the sensor does not cause an explosion because we are sensing flammable gasses by allowing only gaseous elements to pass through the chamber.

MQ from inside

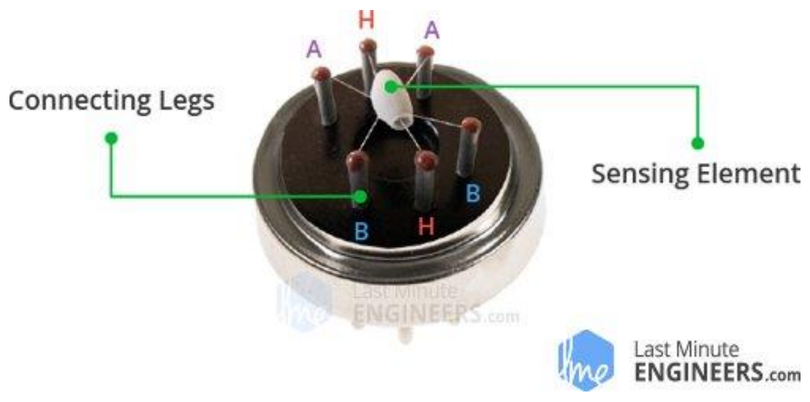


Figure 3-0-3 MQ Sensor inside

As we can see in this photo the MQ sensor only has legs and sensing elements with the six leads are in charge of heating the sensing element and are linked together by a Nickel-Chromium coil (a well-known conductive alloy).

The remaining four signal-carrying leads (A and B) are connected with platinum wires. These wires are connected to the body of the sensing element and convey slight variations in the current flowing through the sensing element.

The tubular sensing element is made of Aluminum Oxide (AL2O3) based ceramic with a Tin Dioxide coating (SnO2). Tin Dioxide is the most important material because it is sensitive to combustible gasses. The ceramic substrate, on the other hand, improves heating efficiency and ensures that the sensor area is continuously heated to the working temperature.

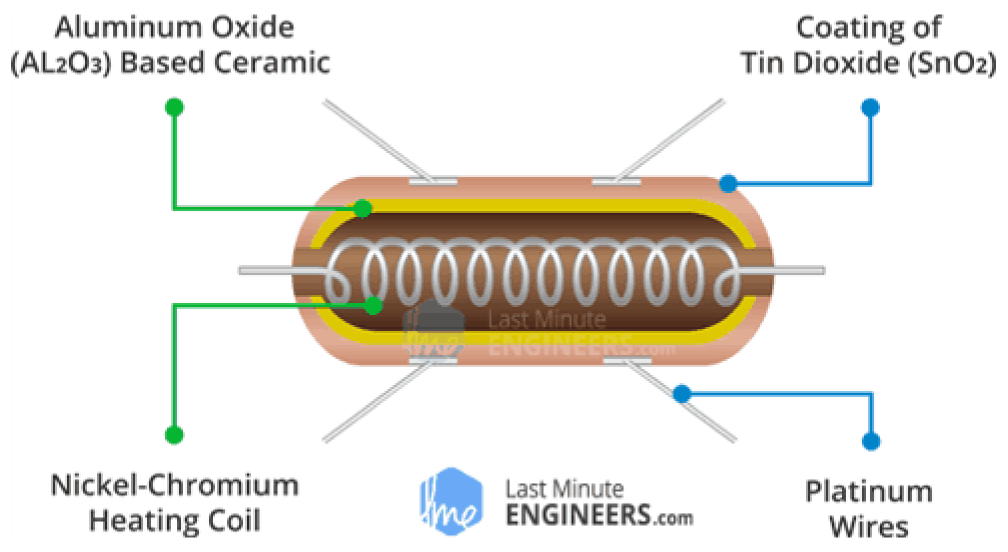

Aluminum Oxide (AL2O3) Based Ceramic
Coating of Tin Dioxide (SnO2)
Nickel-Chromium Heating Coil
Last Minute ENGINEERS.com
Platinum Wires

Figure 3-0-4 Sensing element

The CCT of MQ sensor basically the MQ sensor connected with comparator and potentiometer with for sure starting led and status led (when the value exceeds threshold value)

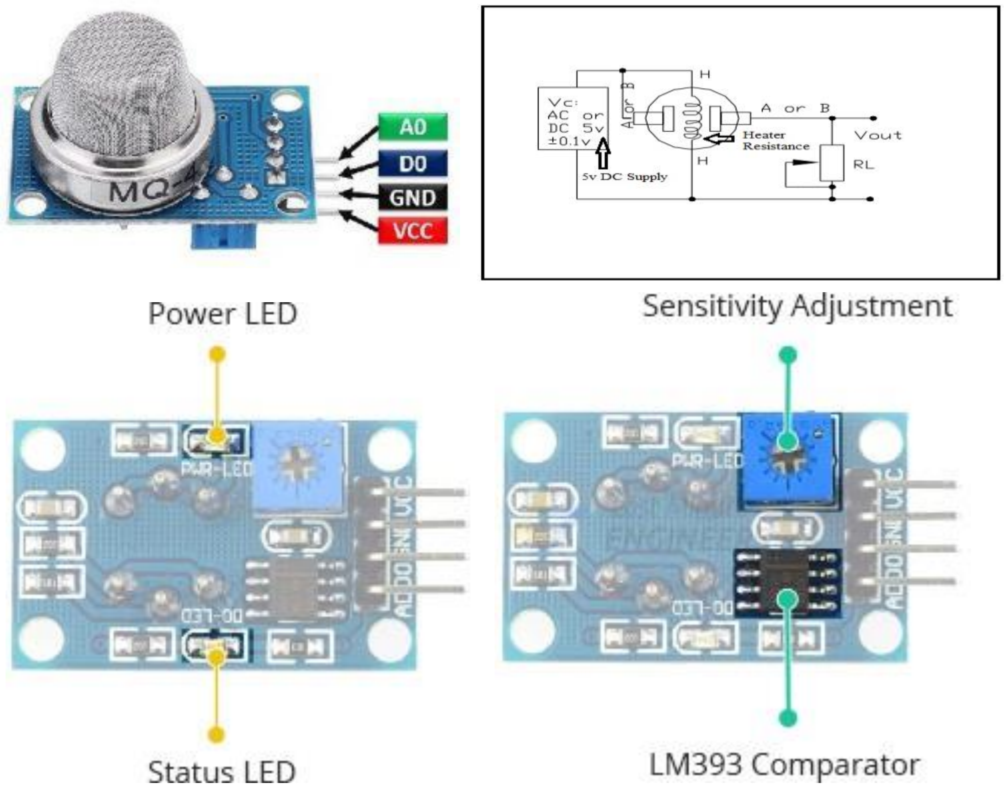


Figure 3-0-5 MQ Contents

How Does a Gas Sensor Work?

When a SnO2 semiconductor layer is heated to a high temperature, oxygen is adsorbed on the surface. When the air is clean, electrons from the conduction band of tin dioxide are attracted to oxygen molecules. This creates an electron depletion layer just beneath the surface of the SnO2 particles, forming a potential barrier. As a result, the SnO2 film becomes highly resistive and prevents electric current flow.

In the presence of reducing gasses, however, the surface density of adsorbed oxygen decreases as it reacts with the reducing glasses, lowering the potential barrier. As a result, electrons are released into the tin dioxide, allowing current to freely flow through the sensor

The MQ sensor are the most important part in the project because the MQ sensors are responsible to take the value of the sample in this project we use 9 MQ sensor each sensor responsible to measure specific value as shown in the table:

MQ-2	methane, butane, smoke
MQ-3	Alcohol. ethanol
MQ-4	methane, CNG gas
MQ-5	Natural gas
MQ-6	LPG, butane gas
MQ-7	Carbon monoxide
MQ-8	hydrogen
MQ-9	Carbon monoxide, flammable gasses
MQ-135	Ammonia, air quality

Table 1 Each MQ sensor specific usage

As we said MQ-sensor is the most important tool in this project so in order to make MQ sensor read the same value every time we need to ensure that:

- Having enough time to warm up
- Stable environment
- Stability in power supply
- Sensor having good placement

3.1.3 Arduino mega

Figure 3-0-6 Arduino mega

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560[1]. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analogs inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller. We need Arduino mega because it has more analog pins to connect 9 MQ sensor which the ordinary sensor will not provide that number of pins.

3.1.4 Arduino Uno

Figure 3-0-7 Arduino Uno

Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller. Simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again. Basically, we use all the pins in the Arduino mega and there are still Touch screen need to get control with for that we use extra Arduino Uno microcontroller to use these pins to control Touch screen, so we use Arduino Uno for nextion resistive touch screen which need a whole Arduino to start working.

3.1.5 Nextion Resistive Touch Screen

Figure 3-0-8 Touch screen

This type of screen, the Nextion resistive touch screen, facilitates touch interaction by applying pressure to multiple layers. It supports input from fingers, stylus, or any object capable of applying pressure, The Nextion board integrates a microcontroller. By utilizing a straightforward SD Card interface, we can execute our GUI design and program it using Arduino. This is achieved through serial communication between the Nextion device and our microcontroller, allowing for event registration and handling. In this project we use this type of screen in order to represent the values and the name of the odor and control some motors by touch. This type of screen needs a fully Arduino board to work.

3.1.6 Lm298 Driver

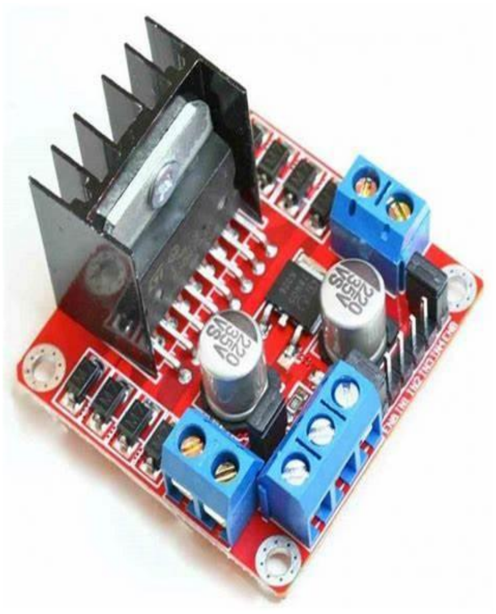


Figure 3-0-9 LM298 Driver

L298N module is a high voltage, high current dual full-bridge motor driver module for controlling DC motor and stepper motor. It can control both the speed and rotation direction of two DC motors.

This module consists of an L298 dual-channel H-Bridge motor driver IC. This module uses two techniques for the control speed and rotation direction of the DC motors.

These are PWM – For controlling the speed and H-Bridge – For controlling rotation direction. These modules can control DC fan motor, so we use LM298N driver to control the voltage applied into the DC fan motor to ensure that its work without high voltage or current to ensure that will not be an odor for this DC fan motor (odor of wires over heat because of high voltage or current).

3.1.7 ethanol

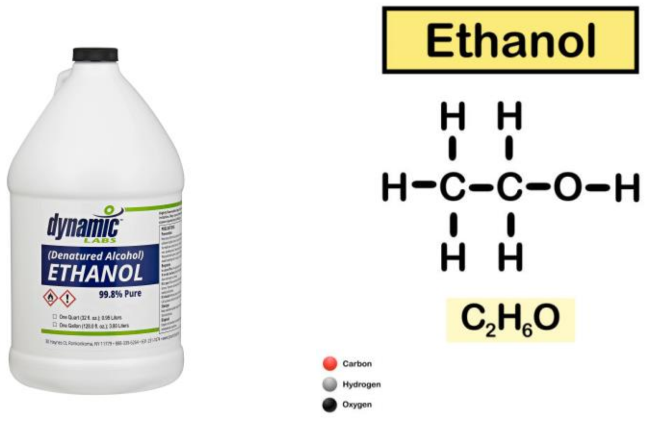


Figure 3-0-10 Ethanol

Ethanol (abbr. EtOH; also called ethyl alcohol, grain alcohol, drinking alcohol, or simply alcohol) is an organic component. It is an alcohol with the chemical formula C2H6O. Its formula can also be written as CH3−CH2−OH or C2H5OH (an ethyl group linked to a hydroxyl group). Ethanol is a volatile, flammable, colorless liquid with a characteristic wine-like odor and pungent, In our project, we face so many problems but the main problem was the air filtration system after we get the sample from any perfume its smell stuck in the MQ sensor box which will cause an error with the next sample value for solving this issue and make sure there isn't any smell left from any perfume we need a chemical element that can get rid of any odor the ethanol and acetone could do that but the acetone case plastic melts so we use ethanol which is much safer for the equipment .

3.1.8 Compressor Nebulizer 230V

A Compressor Nebulizer is an electrical device that converts liquid materiality into a fine spray, allowing it to be easier for detected by MQ sensors. The compressor generates pressurized air, typically at pressures ranging from 8 to 12 pounds per square inch (psi). This compressed air rushes through a narrow jet outlet in the nebulizer cup. As the air rushes through the narrow jet, its velocity increases significantly. According to the Bernoulli principle, this increase in velocity creates a decrease in pressure around the jet outlet. This pressure drop creates a force that pulls the liquid medication from the nebulizer cup into the airstream. Compressor nebulizers are generally more powerful and efficient than other types of nebulizers.

Figure 3-0-11 Compressor Nebulizer 230V

3.1.9 220V fan

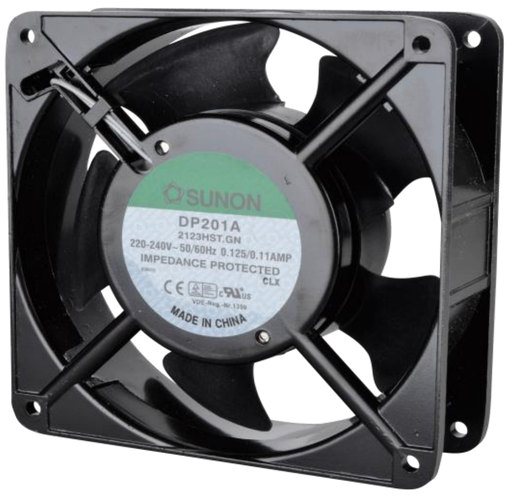


Figure 3-0-12 220V Fan

We use a mini 220V fan for the Air filtration system its responsible to clean the MQ sensor box of any odor after the MQ sensors take the values of the sample the microcontroller will send the signal to the relay to convert into a close CCT It is connected from the supply directly to the common leg in the relay it will work for 15 SEC after that the microcontroller will send another signal to relay to get back the original case.

3.1.10 DC fan (12V)

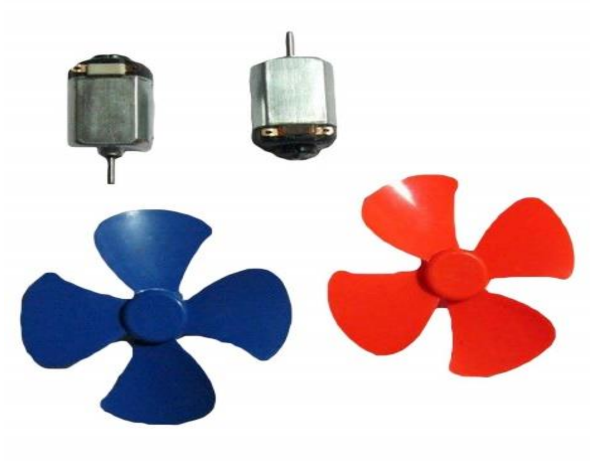


Figure 3-0-13 DC fan (12V)

A DC motor working on 12 volts with fan blades connected into his shaft and fed from the LM298 driver. As we explain recently ethanol work as chemical material that can rid of any smell but unfortunately ethanol has a strong smell that will affect the MQ sensor's results the good thing is we can get rid of ethanol smell by exposing it to extra air intake for this we use an extra fan inside the box.

3.1.13 Power Supplies

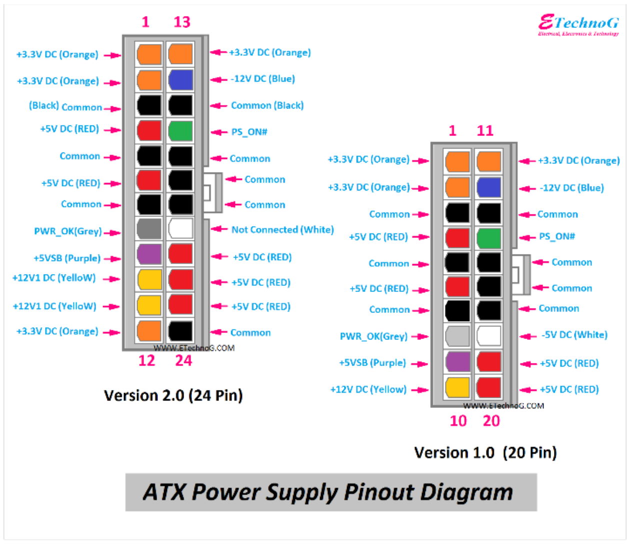


Figure 3-0-14 Power supply

A rectifier is an electrical device that converts alternating current (AC), which periodically reverses direction, to direct current (DC), which flows in only one direction. The reverse operation (converting DC to AC) is performed by an inverter. In this project we use computer power supply to get our needed of DC voltages (12V, 5V, 3.3V) with enough current that can run the MQ sensors and the DC fan and also, we utilize the 230V input power to feed the compressor nebulizer and the 230V fan.

3.1.15 Double Relay

A relay is an electricity operated switch. It consists of a set of input terminals for a single or multiple control signals, and a set of operating contact terminals. The switch may have any number of contacts in multiple contact form, such as make contacts, break contacts, or combinations thereof.

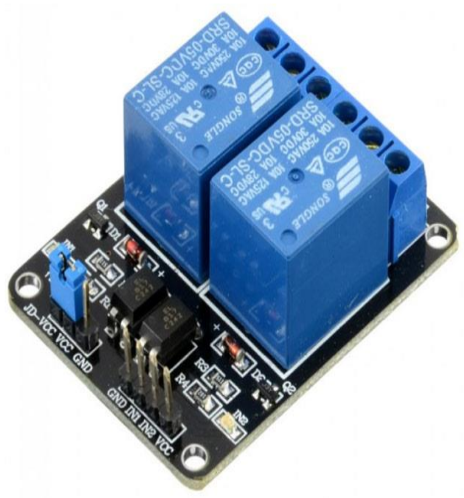


Figure 3-0-15 Double Relay

Relays are used where it is necessary to control a circuit by an independent low-power signal, or where several circuits must be controlled by one signal. Relays were first used in long-distance telegraph circuits as signal repeaters: they refresh the signal coming in from one circuit by transmitting it on another circuit. Relays were used extensively in compressor nebulizer and in 230V fan to control the operation for them.

3.1.16 Funnel

Figure 3-0-16 Funnel

When we apply the samples, the samples goes throw the bottle and MQ sensors then we need to get red of the odor so the 12V fan will work to push the odor onto plastic hole, then and to ensure get the odor to outside the device and because the fan blade take a large circular shape and the hole is small, we use the funnel to maximize the ingesting of air from the hole to outside.

3.1.15 Outside Case

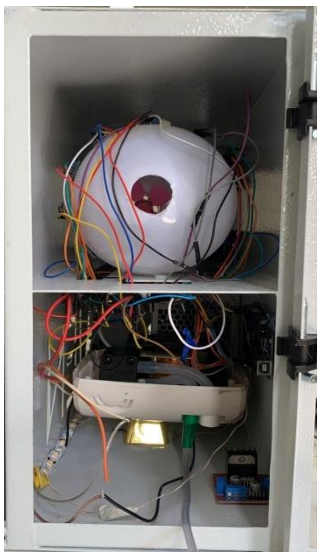


Figure 3-0-17 Outside Case

We notice another problem that we had to arrange all the element that we used in this project in one large case and gives an electric character so we made this case that was designed to help us present our project in beautiful shape and to let us make changes on our project whenever we want to do that.

3.2 block diagram

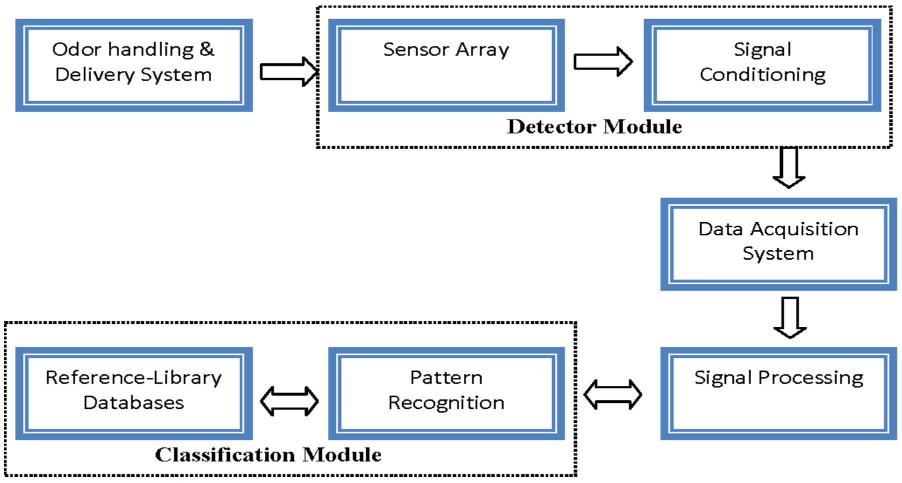


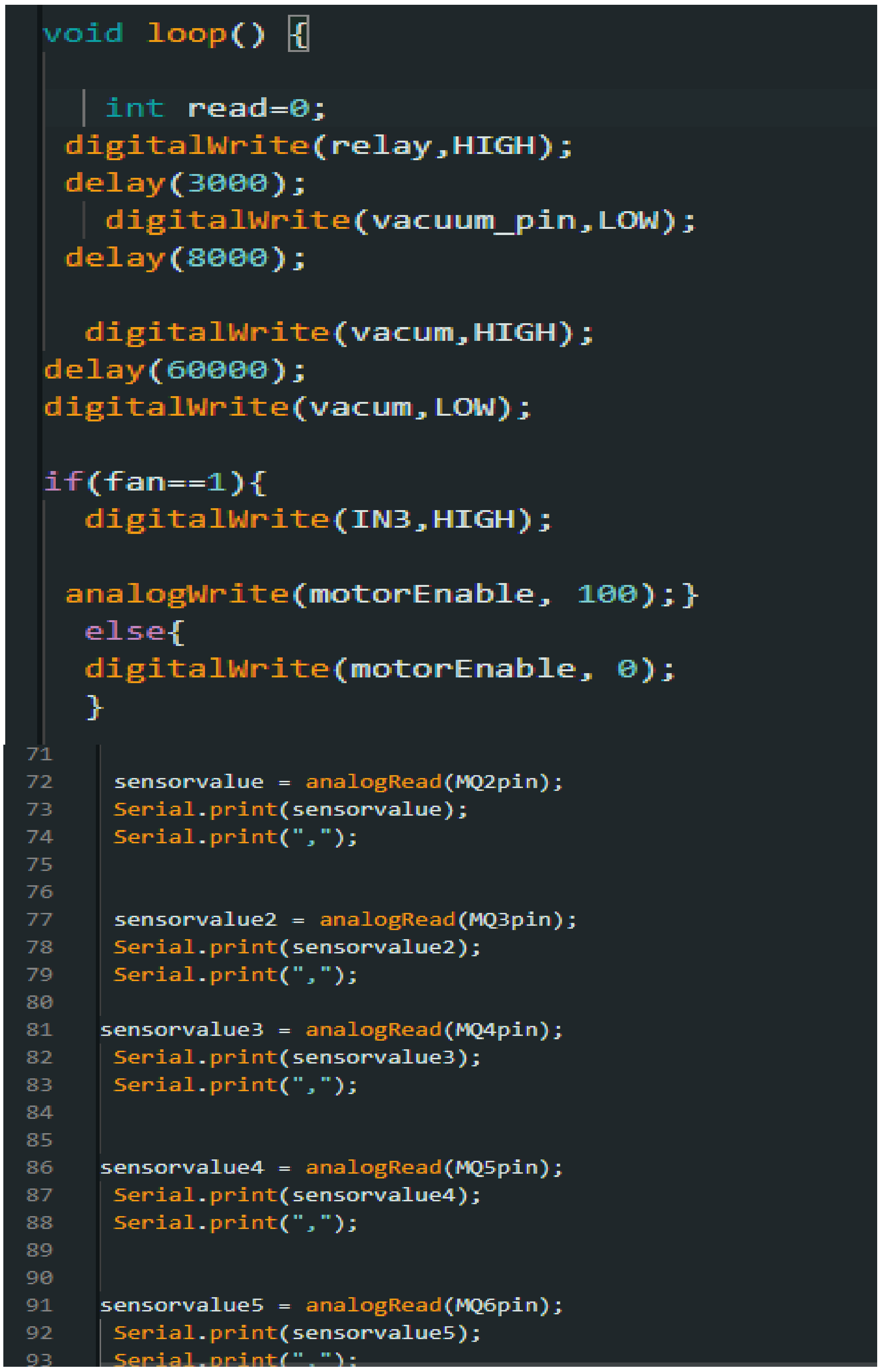
Figure 3-0-18 E-NOSE Block diagram

Block Description:

E-nose operates when volatile organic compounds samples are placed on the odor and handling mechanism simulating a signal carrying the characteristic of the odor. The sampling technique such as headspace, diffusion methods, bubblers or pre-concentrators are being used to draw Theodor molecules into the E-nose [14-15]. Odor sample particles that attracted to the sensor array will go through chemical reactions. Incrementally-different sensors are chosen to make up the cross-reactive sensor array, purposely to act in response to a vast range of chemical classes. The diverse mixtures of possible analyses are then discriminated. At the signal conditioning stage, a distinct digital response pattern is produced, using the assembled and integrated output collected from each individual sensor. In an E-nose, the pattern recognition and data processing techniques played a very important role in the classification module. At this point, recognition of the unique aroma identity (electronic fingerprint) of collective sensor responses will accomplish the identification and classification of an analyzed mixture. The unique aroma identity pattern represents the characteristic of a simple or complex mixture that can be determined without separating the mixture into its individual components before or throughout the analysis. Preceding the analysis of unknown, digital aroma identity reference library for known samples must be constructed. Training is necessary for the pattern classifier with a database of known samples prior to E-nose commercialization in order to predict the response for an unknown sample. To complete the system, computer central processing unit (CPU) as interface, recognition library and recognition software as brain and graphic user interface to process input data from the sensor array for succeeding data analysis must be provided. Included in the library are the pattern recognition algorithms that search the differences between all analyses type patterns which are used to configure the ANN through learning process (neural net training. This ongoing process will only stop when a level of discrimination selected earlier is met. Unknown samples then can be compared with the validated and assembled results in the reference library. Identification of unknown can be done by comparing the similarity of the pattern present in the library databases with the aroma attributes distribution or components that the analyses pattern has. The system is completed by providing an interface using computer central processing unit (CPU), recognition library and recognition software that serve as brain and graphic user interface to process input data from the sensor array for subsequent data analysis.

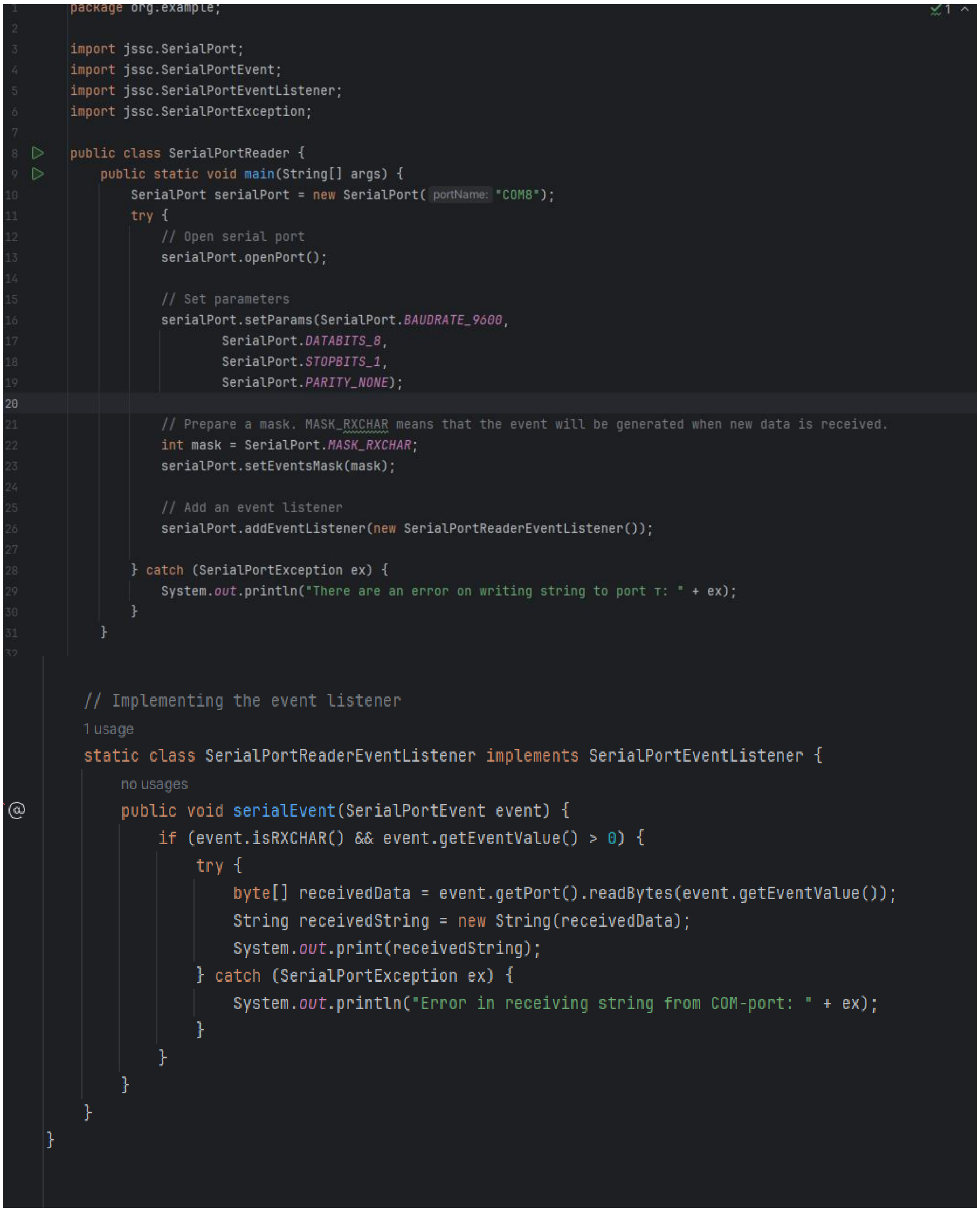
3.3 Software programs and codes

This part analyzes the Arduino code for MQ sensors with controlled 220V fan and DC-mot fan the system design to monitor gas level using multiple sensors the code defines 8 MQ sensors and one 220V fan and vacuum however the code includes a section in the loop function for controlling a motor and fan based on gas sensor readings



```
void loop() {

    int read=0;
  digitalWrite(relay,HIGH);
  delay(3000);
    digitalWrite(vacuum_pin,LOW);
  delay(8000);

   digitalWrite(vacum,HIGH);
delay(60000);
digitalWrite(vacum,LOW);

if(fan==1){
   digitalWrite(IN3,HIGH);

  analogWrite(motorEnable, 100);}
   else{
   digitalWrite(motorEnable, 0);
   }

     sensorvalue = analogRead(MQ2pin);
     Serial.print(sensorvalue);
     Serial.print(",");


     sensorvalue2 = analogRead(MQ3pin);
     Serial.print(sensorvalue2);
     Serial.print(",");

    sensorvalue3 = analogRead(MQ4pin);
     Serial.print(sensorvalue3);
     Serial.print(",");


    sensorvalue4 = analogRead(MQ5pin);
     Serial.print(sensorvalue4);
     Serial.print(",");


    sensorvalue5 = analogRead(MQ6pin);
     Serial.print(sensorvalue5);
     Serial.print(",");
```

By use these codes we can read MQ sensors value part 2: This report builds upon the previously discussed Java program designed to read data from a serial port. The program is now extended to fulfill the requirement of reading MQ sensor values from the Arduino serial monitor and copying them to a database using Java. However, it’s possible to take reading from the Arduino serial monitor but 1000 values are generated every 30 seconds, it is crucial to implement a mechanism for periodic data collection.



```java
package org.example;

import jssc.SerialPort;
import jssc.SerialPortEvent;
import jssc.SerialPortEventListener;
import jssc.SerialPortException;

public class SerialPortReader {
    public static void main(String[] args) {
        SerialPort serialPort = new SerialPort( portName: "COM8");
        try {
            // Open serial port
            serialPort.openPort();

            // Set parameters
            serialPort.setParams(SerialPort.BAUDRATE_9600,
                    SerialPort.DATABITS_8,
                    SerialPort.STOPBITS_1,
                    SerialPort.PARITY_NONE);

            // Prepare a mask. MASK_RXCHAR means that the event will be generated when new data is received.
            int mask = SerialPort.MASK_RXCHAR;
            serialPort.setEventsMask(mask);

            // Add an event listener
            serialPort.addEventListener(new SerialPortReaderEventListener());

        } catch (SerialPortException ex) {
            System.out.println("There are an error on writing string to port т: " + ex);
        }
    }

    // Implementing the event listener
    1 usage
    static class SerialPortReaderEventListener implements SerialPortEventListener {
        no usages
        public void serialEvent(SerialPortEvent event) {
            if (event.isRXCHAR() && event.getEventValue() > 0) {
                try {
                    byte[] receivedData = event.getPort().readBytes(event.getEventValue());
                    String receivedString = new String(receivedData);
                    System.out.print(receivedString);
                } catch (SerialPortException ex) {
                    System.out.println("Error in receiving string from COM-port: " + ex);
                }
            }
        }
    }
}
```

This code will open serial monitor and receive the values coming from Arduino

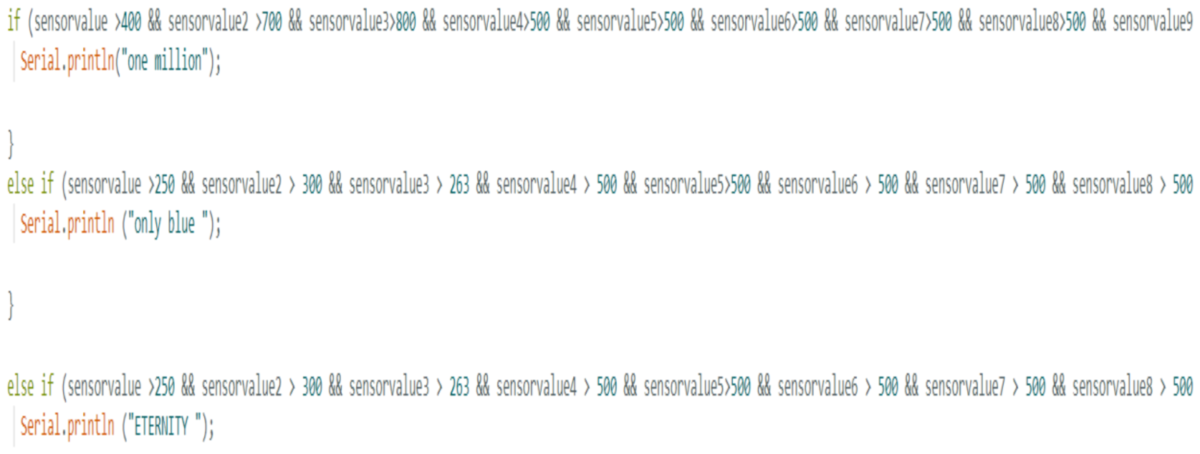
3.4 AI system

Artificial Intelligence, refers to the development of computer systems and machines that can perform tasks that typically require human intelligence. AI enables machines to perceive, reason, learn, and make decisions or take actions based on data and algorithms. It aims to mimic certain aspects of human intelligence and problem-solving capabilities

In general, the electronic nose idea existed and has so many papers talking about it but AIEN is the first project applying AI in electronic nose, so why do we use AI and how do we use it?

We face problem in the beginning of the project it was really hard to collect data when MQ sensor gave the values we were write it in papers every sample take about 2 papers

After that we was manually put the values in Arduino code we built the database from the scratch which is taking much time:



```
if (sensorvalue >400 && sensorvalue2 >700 && sensorvalue3>800 && sensorvalue4>500 && sensorvalue5>500 && sensorvalue6>500 && sensorvalue7>500 && sensorvalue8>500 && sensorvalue9
 | Serial.println("one million");


}
else if (sensorvalue >250 && sensorvalue2 > 300 && sensorvalue3 > 263 && sensorvalue4 > 500 && sensorvalue5>500 && sensorvalue6 > 500 && sensorvalue7 > 500 && sensorvalue8 > 500
 | Serial.println ("only blue ");


}


else if (sensorvalue >250 && sensorvalue2 > 300 && sensorvalue3 > 263 && sensorvalue4 > 500 && sensorvalue5>500 && sensorvalue6 > 500 && sensorvalue7 > 500 && sensorvalue8 > 500
 | Serial.println ("ETERNITY ");
```

Figure 3-0-19 Old detection code

Imagen every sample we need to make it. We must write Arduino code that has the values of each MQ sensor so in that case we need AI to work as data collection. AIEN now can make a database that should include a wide range of odors you want the electronic nose to detect and identify.

Also we have other problem that AI can help with it we invent AIEN to be product the MQ sensor as we know it measure how much PPM in the sample so every time we increase the amount of the sample the MQ sensor will increase the result value so we use AI this time to draw plots for each sample we insert the result of MQ sensor for each amount like in the perfumes the result of first spray of perfume and included in AI database after that we include the value of two sprays of perfume sample and included in the database and the third etc.

The AI code start with importing CSV file into java application, the CSV Reader from open CSV library first thin it gone read CSV store it in List<String[]> rows constructor read all rows from CSV file and initialize variables as num Rown Numcols get Training Data and get Testing Data methods return an In Out object, indicating input and output data post-processing. Exception handling in the constructor addresses potential issues like IO errors, CSV parsing errors, and number format errors, encapsulating them in a Run time Exception for simplicity.

```
package com.example.cnn_java.datapackage;

import com.example.cnn_java.datapackage.dataprocessor.Processor;
import com.opencsv.CSVReader;
import com.opencsv.exceptions.CsvException;
import lombok.Data;

import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.List;

@Data
public class CSVData {
    private List<String[]> rows;
    private int numRows;
    private int numCols;

    public CSVData(File file) {
        try (CSVReader csvReader = new CSVReader(new FileReader(file))) {
            List<String[]> rows = csvReader.readAll();
            this.rows = rows;
            this.numRows = rows.size();
            this.numCols = rows.get(0).length;
        } catch (IOException | CsvException | NumberFormatException e) {
            throw new RuntimeException(e);
        }
    }

    public InOut getTrainingData(Processor processor) {
        return new InOut(processor.getTrainInputs(),
processor.getTrainOutputs());
    }

    public InOut getTestingData(Processor processor) {
        return new InOut(processor.getTestInputs(),
processor.getTestOutputs());
    }
}
```

Part 2 (activation function)

This code generate activation function, it include 3 different activation function

1. Sigmoid
2. RELU
3. TANH

Starting with first activation function Sigmoid (also called logistic function) take any real value as input and output a value in the range (0,1) it is calculated as:

Sigmoid= $\frac{1}{1+e^{-x},}$

Where x is output value of the neuron

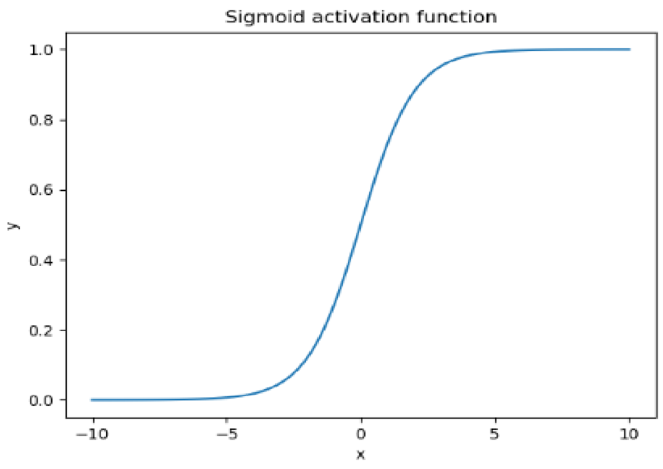


Figure 0-20 Sigmoid activation function

When the output value is close to 1 the neuron is active and enables the flow of information while close to 0 corresponds to an inactive neuron

```
package com.example.cnn_java.neuralnetworkpackage.activationfunctions;

public class Sigmoid implements Activation {
    @Override
    public float activationFunction(float in) {
        return (float) (1 / (1 + Math.exp(-in)));
    }

    @Override
    public float activationFunctionDerivative(float in) {
        float sigmoid = activationFunction(in);
        return sigmoid * (1 - sigmoid);
    }

    @Override
    public String toCpp() {
        return """
                float activationFunction(float in) {
                  return (float) (1 / (1 + exp(-in)));
                }
                """;
    }

    @Override
    public String toH() {
        return """
                #include<math.h>
                float activationFunction(float);
                """;
    }
}
```

2. RELU (rectifies linear unit)
the function returns x if x is positive, and 0 otherwise. Graphically, it looks like a ramp, where the output is zero for all negative inputs and increases linearly for positive inputs.

$$f(x) = x^+ = \max(0, x) = \frac{x + |x|}{2} = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise,} \end{cases}$$

Where x is input neural network

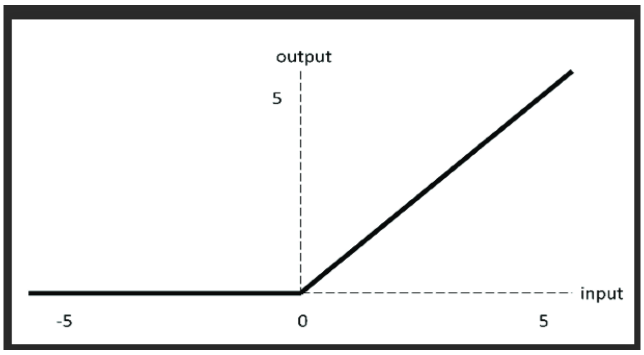


Figure 0-21 RELU activation function

RELU code:

```
package com.example.cnn_java.neuralnetworkpackage.activationfunctions;

public class ReLU implements Activation {
  @Override
  public float activationFunction(float in) {
    return in < 0 ? -0.001f : in;
  }

  @Override
  public float activationFunctionDerivative(float in) {
    return in <= 0 ? (float) 0.01 : 1;
  }

  @Override
  public String toCpp() {
    return """
            float activationFunction(float in) {
                    return in < 0 ? -0.001f : in;
                }
                """;
  }

  @Override
  public String toH() {
    return """
            float activationFunction(float);
            """;
  }
}
```

3. TANH

We can calculate TANH using

Tanh= $\frac{2}{1+e^{-2x}} - 1$

after that we create activation function in the begging because we don’t have clear idea about data we create RELU activation function which is the MAX (0, X) where X = the positive input (same number)
and = 0 when its negative input value.

after we apply it we create 2 other activation function to see if it will give us better results we use Tanh and sigmoid:

Tanh= $\frac{2}{1+e^{-2x}} - 1$

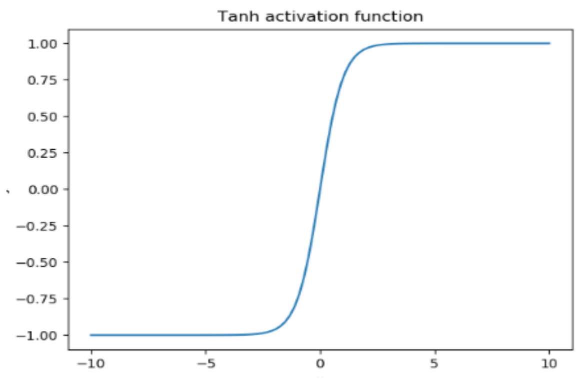


Figure 0-22 Tanh activation function

This activation function is similar to sigmoid the deferent it gives values between 1 to -1

Tanh code:

```java
package com.example.cnn_java.neuralnetworkpackage.activationfunctions;

public class Tanh implements Activation {
  @Override
  public float activationFunction(float in) {
    return (float) Math.tanh(in);
  }

  @Override
  public float activationFunctionDerivative(float in) {
    return (float) (1 - (Math.tanh(in) * Math.tanh(in)));
  }

  @Override
  public String toCpp() {
    return """
            float activationFunction(float in) {
              return (float) tanh(in);
            }
            """;
  }

  @Override
  public String toH() {
    return """
            #include<math.h>
            float activationFunction(float);
            """;
  }
}
```

Part 3 (neural network)

Neural network class this class initialize prediction training testing and code g create input output hidden layers it gives list of layers generate C++ environments to send it to Arduino so this class is important because it represent the essential operations for using the neural network to make predictions, train on data, evaluate performance, and seamlessly integrate it into C++ systems.

```
package com.example.cnn_java.neuralnetworkpackage;

import com.example.cnn_java.arduino.Ciable;
import com.example.cnn_java.datapackage.InOut;
import com.example.cnn_java.datapackage.dataprocessor.ClassificationProcessor;
import com.example.cnn_java.neuralnetworkpackage.activationfunctions.Activation;
import lombok.Builder;
import lombok.Getter;
import lombok.Setter;
import lombok.Singular;

import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

@Getter
@Setter
public class NeuralNetwork implements Ciable {
  private final List<NeuralNetworkLayer> layers;
  private final Activation function;

  @Builder
  public NeuralNetwork(@Singular List<Integer> layers, Activation function) {
    this.function = function;
    this.layers = new ArrayList<>();
    for (int i = 0; i < layers.size() - 1; i++) {
      NeuralNetworkLayer layer = new NeuralNetworkLayer(layers.get(i), layers.get(i + 1));
      this.layers.add(layer);

      if (i > 0) {
        layer.setPreviousLayer(this.layers.get(i - 1));
        this.layers.get(i - 1).setNextLayer(layer);
      }

    }
  }

  public float[] predict(float[] input) {
    NeuralNetworkLayer inputLayer = layers.get(0);
    return inputLayer.forward(input, function);
  }
```

```
public float[] train(InOut data, float learningRate, int epochs) {
    float[][] inputs = data.getInputs();
    float[][] targets = data.getOutputs();
    if (inputs.length  != targets.length) {
      throw new IllegalArgumentException("Number of input samples mus   t match the number of
target samples.");
    }
    float[] epochsGradiantAverageSum =  new float[epochs];

    for (int epoch = 0; epoch < epochs; epoch++) {
      epochsGradiantAverageSum[epoch] =  0;
      for (int i = 0; i < inputs.length; i++) {
        float[] input = inputs[i];
        float[] target = targets[i];

        float[] predictedOutput = predict(input);


        epochsGradiantAverageSum[epoch] += backpropagate(target, predictedOutput, learningRate);
      }
      epochsGradiantAverageSum[epoch] /= inputs. length;
    }
    return epochsGradiantAverageSum;
  }

  private float backpropagate(float[] target, float[] predictedOutput,  float learningRate) {
    int lastLayerIndex = layers.size() - 1;
    NeuralNetworkLayer outputLayer =  layers.get(lastLayerIndex);


    float[] outputGradients =  new float[target.length];
    for (int i = 0; i < target.length; i++) {
      outputGradients[i] = target[i]  - predictedOutput[i];
    }

    float[] gradiantSum = {0};
    outputLayer.backward(outputGradients,  function, learningRate, gradiantSum);
    return gradiantSum[0];
  }


  public float classificationTest(InOut testingData, ClassificationProcessor classificationProcessor) {
    float[][] inputs = testingData.getInputs();
    float[][] outputs = testingData.getOutputs();
    float right = 0;
    float length = inputs. length;
    for (int i = 0; i < length; i++) {
      if (Objects.equals(classificationProcessor.classifyOutInverse(predict(inputs[i])),
@Override
  public String  toCpp() {
    StringBuilder stringBuilder =   new StringBuilder("float* predict(float* in){ \n  return 
l").append(layers.get(0).getLayerNumber()).append(  "forward(in);\n}\n");
    stringBuilder.append(function.toCpp());
    for (NeuralNetworkLayer layer :  layers) {
      stringBuilder.append(layer.toCpp());
    }
    return stringBuilder.toString();
  }

  @Override
  public String  toH() {
    StringBuilder stringBuilder =   new StringBuilder("float* predict(float*); \n");
    stringBuilder.append(function.toH());
    for (NeuralNetworkLayer layer :  layers) {
      stringBuilder.append(layer.toH());
    }
    return stringBuilder.toString();
  }
}
```

class neuralnetworklayer represents a layer in a neural network, it includes properties for weights, biases, inputs, outputs, and other parameters necessary for the functioning of a neural network layer.
The initializeWeightsAndBiases method initializes the weights and biases of the layer using random values.
The forward method performs the forward pass of the neural network layer. It calculates the weighted sum of inputs, applies an activation function, and passes the result to the next layer if it exists.
The backward method performs the backward pass, which is crucial for training the neural network. It calculates gradients, updates weights and biases, and propagates gradients backward to the previous layers

```
package com.example.cnn_java.neuralnetworkpackage;

import com.example.cnn_java.arduino.Ciable;
import com.example.cnn_java.neuralnetworkpackage.activationfunctions.Activation;
import lombok.Builder;
import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

import java.util.Random;

@ToString
public class NeuralNetworkLayer implements  Ciable {
 private static int layersCount  = 0;
 @Getter
 private final int  layerNumber;
 private final int  inputSize;
 private final int  outputSize;
 private float[][] weights;
 private float[] biases;
 private float[] inputs;
 @Getter
 private float[] outputs;
 private float[] gradients;
 @Setter
 private NeuralNetworkLayer nextLayer;
 @Setter
 @ToString.Exclude
 private NeuralNetworkLayer previousLayer;

 @Builder
 public NeuralNetworkLayer(int  inputSize,  int outputSize) {
  this.inputSize  = inputSize;
  this.outputSize  = outputSize;
  layerNumber = layersCount;
  layersCount++;


  initializeWeightsAndBiases();
 }
```

```
private void initializeWeightsAndBiases() {
    Random random = new Random();


    weights = new float[outputSize][inputSize];
    for (int i = 0; i < outputSize; i++) {
      for (int j = 0; j < inputSize; j++) {
        weights[i][j] = (float) random.nextGaussian();
      }
    }


    biases = new float[outputSize];
    for (int i = 0; i < outputSize; i++) {
      biases[i] = (float) random.nextGaussian();
    }
  }

  public float[] forward(float[] input, Activation activation) {
    if (input.length != inputSize) {
      throw new IllegalArgumentException("Input size does not match the expected size.");
    }

    this.inputs = input;


    outputs = new float[outputSize];
    for (int i = 0; i < outputSize; i++) {
      float sum = 0;
      for (int j = 0; j < inputSize; j++) {
        sum += weights[i][j] * input[j];
      }
      outputs[i] = activation.activationFunction(sum + biases[i]);
    }


    if (nextLayer != null) {
      return nextLayer.forward(outputs, activation);
    }
    return outputs;
  }

  public void backward(float[] outputGradients, Activation activation, float learningRate, float[]
gradientSum) {
    if (outputGradients.length != outputSize) {
      throw new IllegalArgumentException("Output gradients size does not match the expected size.");
    }
```

```
gradients = new float[inputSize];
    float gradient = 0;
    for (int i = 0; i < outputSize; i++) {
      gradient = outputGradients[i] *
activation.activationFunctionDerivative(outputs[i]);
      for (int j = 0; j < inputSize; j++) {
        gradients[j] += gradient * weights[i][j];
        weights[i][j] += learningRate * gradient * inputs[j];
      }
      biases[i] += learningRate * gradient;
    }
    gradientSum[0] += Math.abs(gradient);


    if (previousLayer != null) {
      previousLayer.backward(gradients, activation, learningRate,
gradientSum);
    }
  }

  @Override
  public String toCpp() {
    StringBuilder stringBuilder = new StringBuilder();
    stringBuilder.append("float*
").append("l").append(layerNumber).append("forward(float* in){\n")
            .append("   float* outputs = new
float[").append(outputSize).append("];\n");
    for (int i = 0; i < outputSize; i++) {
      stringBuilder.append((i == 0) ? "   float sum = 0;\n" : "   sum =
0;\n");
      for (int j = 0; j < inputSize; j++) {
        stringBuilder.append("    sum +=
").append("l").append(layerNumber).append("w").append(i).append(j).append("
* in[").append(j).append("];\n");
      }
      stringBuilder.append("    outputs[").append(i).append("] =
").append("activationFunction(sum +
l").append(layerNumber).append("b").append(i).append(");\n");
    }
    stringBuilder.append("    delete[] in;\n");
    if (nextLayer != null) {
      stringBuilder.append("    return l").append(layerNumber +
1).append("forward(outputs);\n");
    } else stringBuilder.append("   return outputs;\n");
    stringBuilder.append("}\n");
    return stringBuilder.toString();
  }

  @Override
  public String toH() {
    StringBuilder stringBuilder = new StringBuilder();
    for (int i = 0; i < weights.length; i++) {
      float[] weight = weights[i];
      for (int j = 0; j < weight.length; j++) {
        float w = weight[j];
        stringBuilder.append("#define
l").append(layerNumber).append("w").append(i).append(j).append("
").append(w).append("f\n");
      }
    }
    for (int i = 0; i < biases.length; i++) {
```

part4 (frontend creation)
class CNN Application is important to create frontend Fxml describe what is in this frontend as #of hidden layer training rate epochs etc.

```
package com.example.cnn_java;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import java.io.IOException;
import java.util.Objects;


public class CNNApplication extends Application {

    @Override
    public void start(Stage primaryStage)  throws IOException {

        Parent load =
FXMLLoader.load(Objects.requireNonNull(CNNApplication. class.getResource("view.fxml")));

        Scene scene = new Scene(load,1100,700);
        primaryStage.setScene(scene);
        primaryStage.setTitle("App");
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch();

    }


}
```

part5 (link between back and front)

code constitutes an application for implementing and visualizing a neural network with classification capabilities. The program allows users to load datasets, configure neural network parameters, and visualize the training process through JavaFX charts. It incorporates features for training the neural network, exporting the model to C++ code, and interacting with the user interface. While the code successfully integrates various functionalities, potential improvements include adopting consistent naming conventions, enhancing modularity through encapsulation, and improving code readability through comments and reduced method length. Additionally, the user interface could be refined for a more user-friendly experience, and error handling mechanisms could be strengthened. The overall structure and design suggest a versatile tool for neural network experimentation, with the potential for further refinement in terms of code organization and user interaction.

```java
package com.example.cnn_java;

import com.example.cnn_java.datapackage.CSVData;
import com.example.cnn_java.datapackage.InOut;
import com.example.cnn_java.datapackage.dataprocessor.ClassificationProcessor;
import com.example.cnn_java.neuralnetworkpackage.NeuralNetwork;
import com.example.cnn_java.neuralnetworkpackage.activationfunctions.Activation;
import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.Node;
import javafx.scene.chart.*;
import javafx.scene.control.*;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.FileChooser;
import javafx.util.StringConverter;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.lang.reflect.InvocationTargetException;
import java.util.*;
import java.util.concurrent.atomic.AtomicReference;

import static com.example.cnn_java.InterfaceImplementingClassesFinder.findImplementingClasses ;
```

```
public class Controller {
 @FXML
 public ScatterChart testChart;
 @FXML
 public ScatterChart predictedChart;
 private static ClassificationProcessor classificationProcessor;
 private static LinkedHashMap<String, XYChart.Series>  seriesList;

 private static InOut trainingData;

 private static InOut testingData;
 private static NeuralNetwork neuralNetwork;
 private static float[][] inputs;
 private static float[][] outputs;
 private static CSVData csvData;
 private static int runNumber;
 @FXML
 public ChoiceBox functionSelection;
 @FXML
 public Spinner  epochsSpinner;
 @FXML
 public Spinner  learningRateSpinner;
 @FXML
 public Spinner  trainingDataSpinner;
 @FXML
 public AreaChart gradientGraph;
 @FXML
 public Label accuracy;
 @FXML
 public CheckBox goalCheckBox;
 @FXML
 public HBox parent;

 @FXML
 public Label loadedFileName;
 @FXML
 public Button resetButton;
 @FXML
 public Button startButton;
 @FXML
 public HBox testDataContainer;
 private boolean init = true;

 @FXML
 public Spinner  hiddenNeuronsSpinner ;
 private FileChooser fileChooser;
```

```
items.addAll(implementedActivationFunctions);
  functionSelection.setItems(items);
  functionSelection.setValue(items.get(0));
  functionSelection.setConverter(new StringConverter() {
   @Override
   public String toString(Object o) {
    return ((Class<?>) o).getSimpleName();
   }

   @Override
   public Object fromString(String s) {
    return null;
   }
  });

 }

 @FXML
 public void onStartButtonClick(ActionEvent actionEvent) throws NoSuchMethodException,
InvocationTargetException, InstantiationException, IllegalAccessException {
  Platform.runLater(() -> {
        if (init) {
         classificationProcessor = new ClassificationProcessor(csvData, ((Double)
trainingDataSpinner.getValue()).floatValue());
         trainingData = csvData.getTrainingData(classificationProcessor );
         testingData = csvData.getTestingData(classificationProcessor );
         try {
          neuralNetwork = NeuralNetwork.builder()
               .layer(classificationProcessor.classes.size() - 1)
               .layer((int) hiddenNeuronsSpinner.getValue())
               .layer(classificationProcessor.classes.get("class").size())
               .function((Activation) ((Class<?>)
functionSelection.getValue()).getDeclaredConstructor().newInstance())
               .build();
         } catch (InstantiationException | IllegalAccessException | InvocationTargetException |
               NoSuchMethodException e) {
          throw new RuntimeException(e);
         }
         inputs = testingData.getInputs();
         outputs = testingData.getOutputs();
         seriesList = new LinkedHashMap<>();
         runNumber = 0;
         fillGraph(testChart, inputs, outputs, "", false);
         gradientGraph.getData().add( new XYChart.Series<>());
         disableInputs();
         init = false;
        }
        Integer epochsSpinnerValue = (Integer) epochsSpinner.getValue();
float[] gradiantAverageSum = neuralNetwork.train(trainingData, ((Double)
learningRateSpinner.getValue()).floatValue(), goalCheckBox.isSelected() ? 1 : epochsSpinnerValue);
          List<float[]> inputs = new ArrayList<>();
          List<float[]> outputs = new ArrayList<>();
          for (int i = 0; i < Controller.inputs.length; i++) {
           float[] input = Controller.inputs[i];
           float[] predicted = neuralNetwork.predict(input);
           if (Objects.equals(classificationProcessor.classifyOutInverse(predicted),
classificationProcessor.classifyOutInverse(Controller.outputs[i]))) {
            inputs.add(input);
            outputs.add(predicted);
           }
          }
          if (outputs.size() > 0) {
           fillGraph(predictedChart, inputs.toArray(float[][]::new), outputs.toArray(float[][]::new),
"predicted", true);
          }
          float classificationTest = neuralNetwork.classificationTest(testingData,
classificationProcessor);
          accuracy.setText(String.format("%.2f %%", classificationTest));
          XYChart.Series series = (XYChart.Series) gradientGraph.getData().get(0);

          for (int i = 0; i < gradiantAverageSum.length; i++) {
           series.getData().add(new XYChart.Data<>((runNumber + i), gradiantAverageSum[i]));
          }
          runNumber += epochsSpinnerValue;
          if (goalCheckBox.isSelected() && classificationTest >= epochsSpinnerValue) break;
         } while (goalCheckBox.isSelected());
        });
       }
   );
  }

 private void fillGraph(ScatterChart scatterChart, float[][] inputs, float[][] outputs, String name,
boolean rebuild) {
   if (rebuild) {

    Set<Map.Entry<String, XYChart.Series>> entries = seriesList.entrySet();
    for (Map.Entry<String, XYChart.Series> next : entries) {
     if (next.getKey().contains(name))
      next.getValue().getData().clear();
    }
   }
   for (int i = 0; i < inputs.length; i++) {
    float color = inputs[i][1];
    StringBuilder combination = new
StringBuilder(classificationProcessor.classifyOutInverse(outputs[i])).append(color);
```

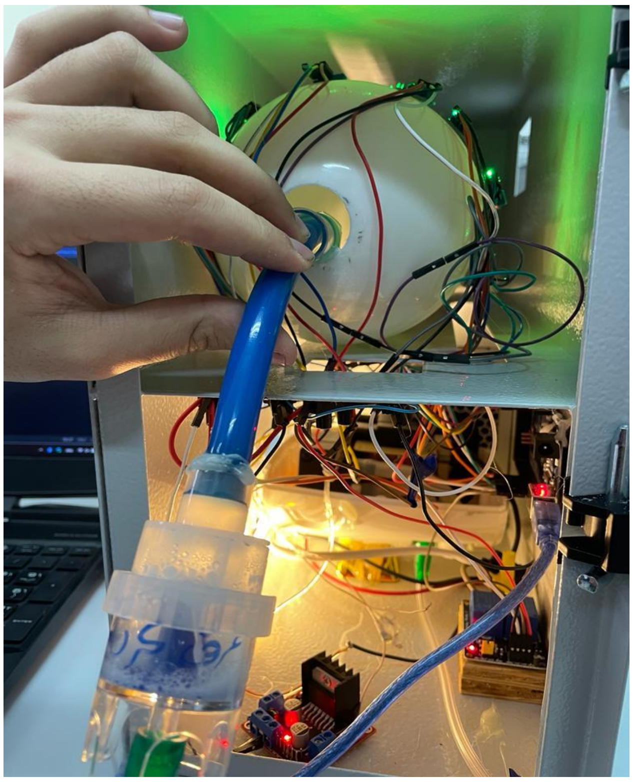
```
XYChart.Series<Object, Object> serieS =   new XYChart.Series<>();
        serieS.getData().add(new  XYChart.Data<>(inputs[i][1], inputs[i][ 0]));
        seriesList.put(combination + name, serieS);
        scatterChart.getData().add(serieS);
      } else
        series.getData().add(new  XYChart.Data<>(inputs[i][1], inputs[i][ 0]));
    }

    int size = 8 - scatterChart.getData().size();
    for (int i = 0; i < size; i++) {
      scatterChart.getData().add(new  XYChart.Series<>());
    }
  }

  @FXML
  public void  onResetButtonClick(ActionEvent actionEvent) {
    enableInputs();
    clearGraphs();
    accuracy.setText(String.format("%.2f %%" , 0f));
    init = true;
  }

  private void  disableInputs() {
    if (goalCheckBox.isSelected())
      epochsSpinner.setDisable(true );
    learningRateSpinner.setDisable( true );
    hiddenNeuronsSpinner.setDisable( true);
    trainingDataSpinner.setDisable( true);
  }

  private void  enableInputs() {
    goalCheckBox.setSelected(false);
    epochsSpinner.setDisable(false);
    learningRateSpinner.setDisable( false);
    hiddenNeuronsSpinner.setDisable( false);
    trainingDataSpinner.setDisable(false);
  }

  private void  clearGraphs() {
    if (!init) {
      for (XYChart.Series series :  seriesList.values()) {
        series.getData().clear();
      }
      gradientGraph.getData().clear();
    }
  }

  public void  onLoadFileClick(ActionEvent actionEvent) {
```

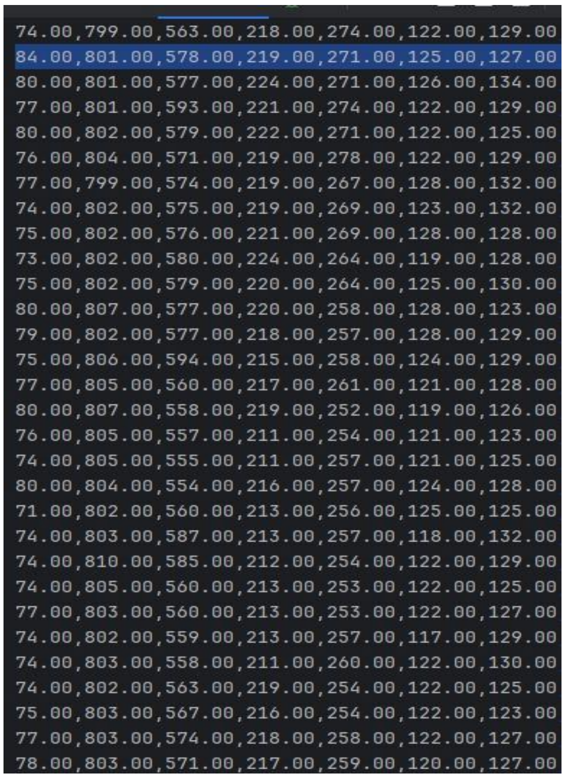
```
Label label = new Label();
        label.setText(csvData.getRows().get(0)[i]);
        VBox vBox = new VBox();
        vBox.getChildren()
                .addAll(label,  new TextField());
        testDataContainer.getChildren().add(vBox);
      }
      Button button = new Button();
      button.setText("Predict");
      testDataContainer.getChildren().add(button);
      Label label = new Label();
      AtomicReference<TextField> textField = new AtomicReference<>();
      testDataContainer.getChildren().add(label);
      button.setOnMouseClicked(mouseEvent  -> {
        label.setText("");
        List<String> inputs =  new ArrayList<>();
        List<String> labels =  new ArrayList<>();
        ObservableList<Node> children =  testDataContainer.getChildren();
        int lastBox = children.size()  - 3;
        for (int i = 0; i < children.size()  - 2; i++) {
          Node outerNode = children.get(i);
          ObservableList<Node>  nodes = ((VBox) outerNode).getChildren();
          for (Node node : nodes) {
            if (node instanceof TextField) {
              String text = ((TextField) node).getText();
              if (text == null  || text.isBlank() && outerNode != children.g et(lastBox))
                label.setText("Wierd Input" );
              if (outerNode == children.get(lastBox)) {
                textField.set((TextField) node);
                node.setDisable(true);
              } else
                inputs.add(text);
            }
            if (node instanceof Label) {
              String text = ((Label) node).getText();
              if (text == null  || text.isBlank())
                label.setText("Wierd Input" );
              else
                labels.add(text);
            }
          }
        }
        float[] inputsFloat = new float[inputs.size()];
        for (int i = 0; i < inputsFloat.length; i++) {
          inputsFloat[i] =  classificationProcessor.classifyIn(inputs.get(i), labels.get(i));
        }

textField.get().setText(classificationProcessor .classifyOutInverse( neuralNetwork .predict(inputsFloat))
IOException {
          // Creating nuralNetwork.h
          String headerFileName = "nuralNetwork.h";
          FileWriter writerH = new FileWriter(headerFileName);
          writerH.write("// Header file for nuralNetwork\n#include
\"Arduino.h\"\n");
          writerH.write(classificationProcessor.toH());
          writerH.write(neuralNetwork.toH());
          writerH.close();

          // Creating nuralNetwork.cpp
          FileWriter writerCpp = new FileWriter("nuralNetwork.cpp");
          writerCpp.write("// Implementation file for nuralNetwork\n#include \""
+ headerFileName + "\"\n");
          writerCpp.write(classificationProcessor.toCpp());
          writerCpp.write(neuralNetwork.toCpp());
          writerCpp.close();
        }
}
```

Chapter four: Result and discussion

4.1 implementation and Result

first, we fill the nebulizer cup with 6 centimeters cubic (cc) of the desired sample then we run the code so the nebulizer is run for 90s and show the result from 70-90s (steady state time).

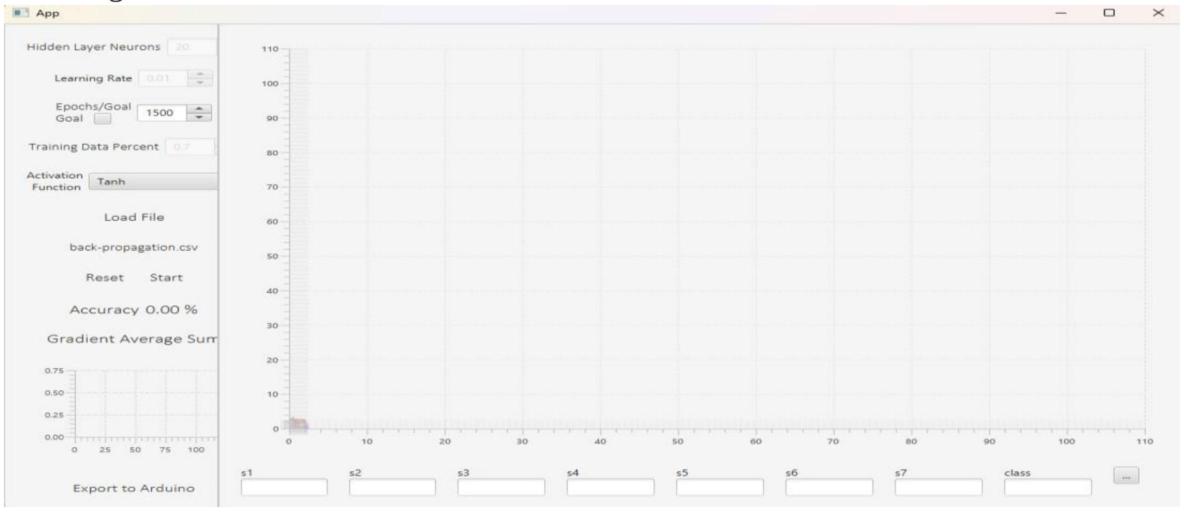




```
74.00,799.00,563.00,218.00,274.00,122.00,129.00
84.00,801.00,578.00,219.00,271.00,125.00,127.00
80.00,801.00,577.00,224.00,271.00,126.00,134.00
77.00,801.00,593.00,221.00,274.00,122.00,129.00
80.00,802.00,579.00,222.00,271.00,122.00,125.00
76.00,804.00,571.00,219.00,278.00,122.00,129.00
77.00,799.00,574.00,219.00,267.00,128.00,132.00
74.00,802.00,575.00,219.00,269.00,123.00,132.00
75.00,802.00,576.00,221.00,269.00,128.00,128.00
73.00,802.00,580.00,224.00,264.00,119.00,128.00
75.00,802.00,579.00,220.00,264.00,125.00,130.00
80.00,807.00,577.00,220.00,258.00,128.00,123.00
79.00,802.00,577.00,218.00,257.00,128.00,129.00
75.00,806.00,594.00,215.00,258.00,124.00,129.00
77.00,805.00,560.00,217.00,261.00,121.00,128.00
80.00,807.00,558.00,219.00,252.00,119.00,126.00
76.00,805.00,557.00,211.00,254.00,121.00,123.00
74.00,805.00,555.00,211.00,257.00,121.00,125.00
80.00,804.00,554.00,216.00,257.00,124.00,128.00
71.00,802.00,560.00,213.00,256.00,125.00,125.00
74.00,803.00,587.00,213.00,257.00,118.00,132.00
74.00,810.00,585.00,212.00,254.00,122.00,129.00
74.00,805.00,560.00,213.00,253.00,122.00,125.00
77.00,803.00,560.00,213.00,253.00,122.00,127.00
74.00,802.00,559.00,213.00,257.00,117.00,129.00
74.00,803.00,558.00,211.00,260.00,122.00,130.00
74.00,802.00,563.00,219.00,254.00,122.00,125.00
75.00,803.00,567.00,216.00,254.00,122.00,123.00
77.00,803.00,574.00,218.00,258.00,122.00,127.00
78.00,803.00,571.00,217.00,259.00,120.00,127.00
```

Figure 4-0-1 implementation and results

after that we take one random line from these and apply it to the front-end window, like the following:



App

Hidden Layer Neurons 20

Learning Rate 0.01

Epochs/Goal 1500

Goal

Training Data Percent 0.7

Activation Function Tanh

Load File

back-propagation.csv

Reset Start

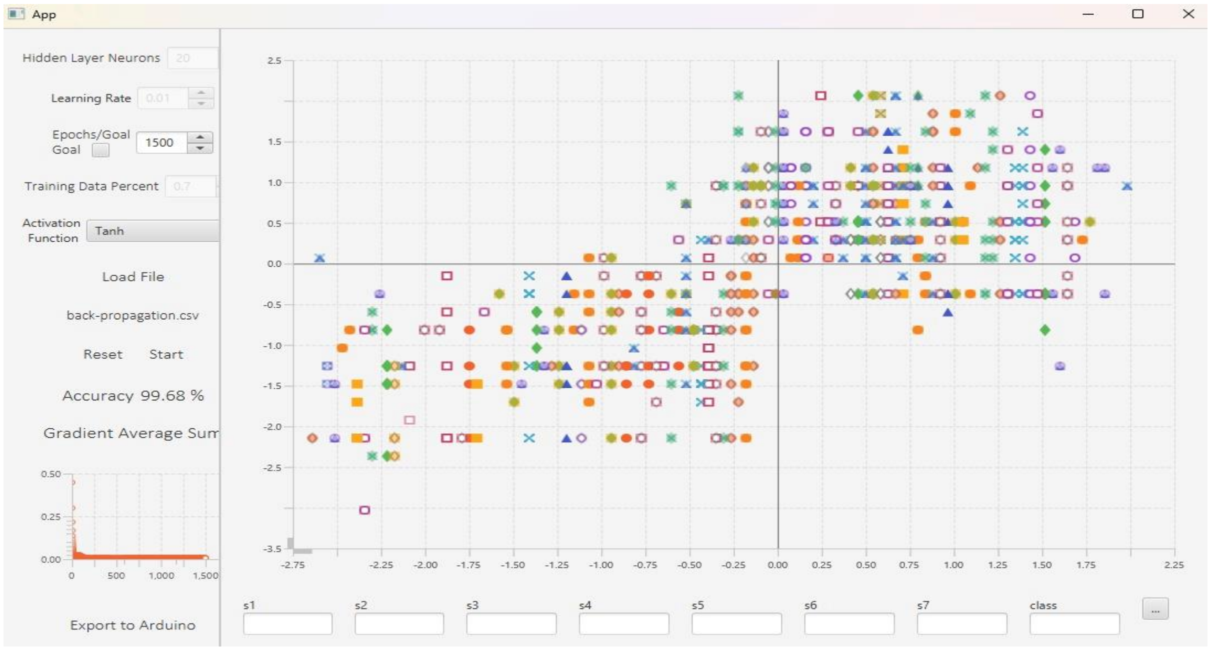
Accuracy 0.00 %

Gradient Average Sum

Export to Arduino

Figure 4-0-2 Front-end window

In our project we deal with four diffrent samples (two alcoholic and to nonalcoholic) that was Bavaria, BLU, Red-label and Araq. after we take 3 samples for each for training in the back-propagation, and then we apply the csv file to the front-end window and we got some result, the first one is for the Red-label and Araq (alcoholic samples) and we got these result in the graph:

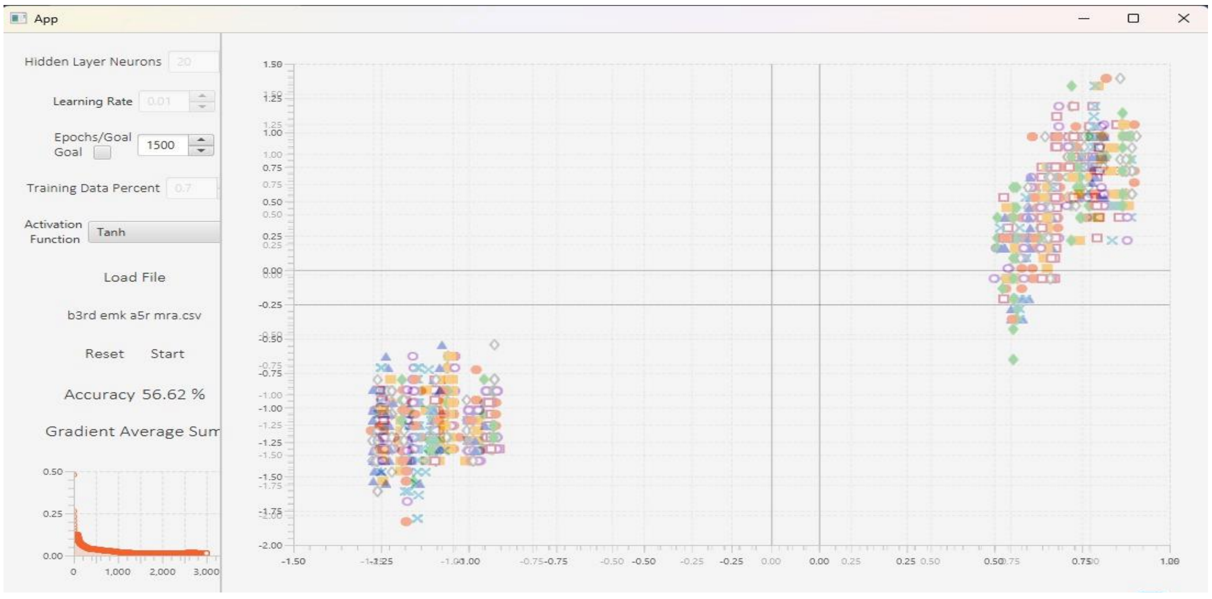


Hidden Layer Neurons 20
Learning Rate 0.01
Epochs/Goal 1500
Training Data Percent 0.7
Activation Function Tanh
back-propagation.csv
Accuracy 99.68 %

Figure 0-3 Alcoholic samples results

from this graph we can figure out what was done in the back-propagation operation (Tanh method) and we can see the neural network result and how the point are separate and clear to detect.

and here is the result after adding the BLU and Bavaria (non-alcoholic samples) and we got these results:



Hidden Layer Neurons 20
Learning Rate 0.01
Epochs/Goal 1500
Training Data Percent 0.7
Activation Function Tanh
b3rd emk a5r mra.csv
Accuracy 56.62 %

Figure 0-4 all samples results

we can see from the last graph that there is a wide difference between the alcoholic and non-alcoholic samples after we training the program on these samples.

4.2 Discussion

The results clearly demonstrate the capability of the AIEN system to reliably detect and discriminate between different volatile organic compounds (VOCs) using the array of MQ gas sensors. The consistent and distinct odor profile plots generated for various substances verify the effectiveness of the integrated hardware and software components.

The filtration system using fans to clear the residual VOCs between samples was crucial to obtaining accurate readings. Without properly cleaning the sampling chamber between tests, residual VOCs would create overlapping sensor signals and prevent the isolation of unique odor profiles.

The characteristic plots for the alcoholic and non-alcoholic beverages showcase the efficacy of the backpropagation algorithm and tanh activation function in extracting distinguishing features in the multivariate MQ sensor data. The clear separation between the two groups on the scatter plot matrix confirms the pattern recognition capability of the AI software based on the training data.

While the results are promising, further enhancements could improve the sensitivity, selectivity, and accuracy of the AIEN system. Additional sensor modalities utilizing different transduction mechanisms such as metal oxide semiconductors, infrared spectroscopy, or electrochemical cells could provide more orthogonal VOC measurements and enhance discrimination. Expanding the reference database with more VOC samples and concentrations would also augment the training data for the neural network and improve pattern recognition performance.

More complex neural network architectures beyond the simple multilayer perceptron used here could potentially extract non-linear relationships and interactions between sensor signals. Advanced techniques like convolutional neural networks may better uncover latent VOC features. However, increased model complexity risks overfitting and would require more substantial training data.

Overall, the results confirm the viability of interfacing an array of gas sensors with AI software for intelligent identification of VOCs. With incremental improvements to the sensors, sampling process, reference database, and machine learning approach, the performance of the AIEN system could likely match or exceed human olfactory capabilities.

4.3 Economical feasibility

In this section we analyze the economical part of our project, we will put the price of each component that we use in our project in the NIS shekel unit, and it is as the following:

component	price
Plastic bottle	20 ₪
MQ sensors	250 ₪
Arduino mega	90 ₪
Arduino Uno	40 ₪
Nextion touch screen	100 ₪
Lm298 driver	20 ₪
Ethanol	20 ₪
Compressor Nebulizer	50 ₪
220V fan	20 ₪
DC fan (12V)	10 ₪
Power supply	-
Double Relay	10 ₪
Funnel	5 ₪
Outside case	200 ₪

Table 2 components costs

Chapter five: Problem we face

In this project we face do many problems in this chapter we will talk about these problems and how we solve it

5.1 filtration

when we start the project we face the problem that when we apply some samples to the sensor the smell stuck in the box which will affect the result of the second sample commonly it is none we can get rid of any smell by Expose it to the sun and air that take much time that we try one sample and waiting for the day to apply another sample we solve this problem by making fraternization system which we explain it in the process using methanol to clean all parts of the system with heating element we solve this problem.

5.2 inaccurate values

we note that the values every time changes and it never is the same or increase with the same amount the problem that we found is the shape of the box distracting the smell inside the box so to solve this problem we add funnel making circular shape that the smell will not distracting also if any sensor do not take enough we add to the controlling system bridge that the pipe will move to any sensor.

5.3 data collection

the other problem we face is we need to take so many values for each sample in 3,6,9 SEC with different substance concentration to put it in database this take to many times so we solve this problem by adding AI system which help us to collect all the data we need and draw plot for each sample.

5.4 sample taker

first, we were but the perfumes using cartoons or directly to the MQ sensors this make the result not accurate at to solve this we try hard plastic which is easy to clean

5.5 high current needed

in this project we have 9 MQ sensor and MQ sensor need good current in order to operate correctly the microcontroller couldn't delivered the amount of current needed it could for two MQ sensor but when we apply 4 MQ sensor the Arduino mega shut down also we have many motors that need enough current to operate well so in order to solve this problem we connect inverters with different voltages that can handle the amount of current needed.

5.6 similarity of value

as we said every odor in this world has different VOCS but because we are using an MQ sensor which is the commercial sensor for sure it won't give us accurate values for this reason there is some smell that are really close to the values in order we see that in different type of odors to solve this problem we start to measure other important variable that the same of VOCS this value couldn't be the same values in the materials we are talking about mole wight we measure mole weight using piezoelectric sensor every value will gave us different value of voltage and we make other database for this reason.

5.7 acetone

the reason why we use ethanol instead of acetone is because of what happens in plastic pipes every time we put acetone it interacts with plastic and burns it after so many time putting acetones the pipes shrinks so we look for other material that could rid of smells and won’t interact with plastic and we choose ethanol.

5.8 powder damage MQ sensors

when we reach stage three we unfortunately change all the MQ sensors why? Because when we apply powder in the sensors it gets stuck in the stainless steel layer so the result for the next sample was the fault. In order to solve this problem we add gauze which helps us to prevent any powder from reaching MQ sensors.

Chapter six: Conclusion and future plans for AIEN

6.1 Conclusion

In conclusion, this project successfully implemented a proof-of-concept artificial intelligence electronic nose capable of discriminating between VOCs based on automated MQ sensor data and neural network analysis. The results highlight the prospects for fusing gas sensor arrays with AI techniques to electronically perceive and classify complex odors.

The consistent VOC profiles generated by the AIEN system prove the effectiveness of the hardware and software components, including the filtration process, fan controls, sensor array, data acquisition, and pattern recognition algorithms. Distinct separation between the various VOC samples was achieved, verifying the system's ability to reliably detect and identify different substances based on training data.

While the current results are promising, enhancements to the sensor technology, sampling process, reference database, and machine learning approach could improve the sensitivity, selectivity, and accuracy of the AIEN system. With sufficient development, such intelligent e-nose systems could match or exceed human olfactory capabilities in many applications.

This project establishes a solid foundation and development pathway for engineering practical artificial intelligence electronic nose implementations. With additional work, AIEN systems could enable transformative applications in areas ranging from healthcare diagnostics to industrial quality control, agriculture, security, and environmental monitoring. Intelligent electronic noses promise to provide rapid, sophisticated odor detection and analysis, augmenting human senses. This project confirms the significant potential of merging gas sensor arrays with artificial intelligence for next-generation smart odor detection systems.

6.2 Future plans for AIEN

In this project we have so many ideas that unfortunately we couldn't do it in this chapter we will represent all of these ideas that we hope in the future these feature AIEN will have them:

1) **AIEN can figure the percentage of each VOCS in the material**
   AIEN can recognize the VOCS inside the material but it couldn't know exactly the percentage of this VOCS we can do that by using specific sensor as
   1. electrochemical sensors
   2. metal oxide sensors
   3. infrared spectroscopy

Figure 0-1 Metal oxide sensor

Figure 0-3 Electrochemical sensor

Figure 0-2 Infrared sensor

We couldn’t add these sensors because of high cost of each one of them

2) **Face detection:**
   as we say AIEN can detect drugs so it could be used in the airports using raspberry pi AIEN can be connected to governments database with Criminal record holders or drug dealers to catch them.

3) **from one MQ sensor AIEN can detect the odor**:
   we hope in the future and after adding the three sensors above AIEN could detect the odor by using only MQ2 sensor

Chapter seven: References

(n.d.). From arduino: https://www.arduino.cc/

(PDF) Detection of fruits in warehouse using Electronic nose. (n.d.). From researchgate: researchgate.net

Harmouzi, M. (n.d.). *Conception and Simulation of an Electronic Nose Prototype for Olfactory Acquisition*. From ASTES: www.astesj.com

In-Depth: How MQ2 Gas/Smoke Sensor Works? & Interfacev it with Arduino. (n.d.). From lastminuteengineers: lastminuteengineers.com

Soh, A. C. (n.d.). *Development Of Neural Network-Based Electronic Nose For Herbs Recognition*. From Sciendo: https://sciendo.com/article/10.21307/ijssis-2017-671

THE ELECTRONIC NOSE. (n.d.). From uc: uc.edu

https://web.archive.org/web/20180513090020/http://www.maskau.dk/projects/electronic-nose