PROJECT-BASED LEARNING REPORT

GUI APPLICATION DEVELOPMENT FOR ELECTRONIC NOSE DATA VISUALIZATION

Lecturer: Ahmad Radhy S.Si, M.Si

Group 4:

Evan Javier Firdausi M.	2042241010
Rahma Aulia	2042241056
M.Farel Alberto Firest	2042241058

DEPARTEMEN TEKNIK INSTRUMENTASI
FAKULTAS VOKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
2025

Table of Contents

Chapter 1

Introduction

1.1 Background

The electronic nose (eNose) has been proven effective for analyzing chocolate quality through the detection of volatile compounds that determine aroma, such as cocoa fermentation and roasting [Gunaratne et al., 2023]. In the chocolate industry, aroma is a key indicator of production batch quality, where differences in cocoa type or fermentation level produce unique gas sensor response patterns, enabling non-destructive classification [Malaysian Cocoa Journal, 2023]. This project chose chocolate samples as the focus of testing due to their complex aroma rich in volatile compounds, suitable for testing the integration of gas sensors with Arduino Uno R4 WiFi, Rust backend, and Qt Python GUI visualization.

The development of the eNose system for chocolate requires real-time signal processing from sensors such as the MICS-5524 or multichannel gas sensors, which detect gases such as CO, NH3, and roasting-specific compounds, and then visualizing them via GNUPLOT for response pattern analysis [Haiying et al., 2021]. Recent research has shown that eNose achieves up to 90 classification accuracy for cocoa fermentation levels using simple machine learning, supporting integration with Edge Impulse [Zeppa et al., 2021]. Thus, this system trains the competencies of sensor data acquisition, signal processing, and visualization for food industry applications.

Recent studies show that electronic nose systems are increasingly used to evaluate food quality because they can capture complex volatile patterns more quickly and consistently than human panels. In cocoa and chocolate applications, e-nose devices have achieved high classification accuracy for fermentation degree and bean quality by combining sensor arrays with supervised learning methods, demonstrating their potential as objective tools for process monitoring. Other works in food products also highlight that e-nose measurements can distinguish subtle differences in formulation or processing, supporting quality control and adulteration detection tasks in industrial environments. Building on these findings, this project develops an e-nose platform dedicated to chocolate aroma analysis, integrating gas sensors, Arduino Uno R4 WiFi, a high-performance Rust backend, and a Qt Python GUI to acquire, process, and visualize sensor signals in real time.

Several cocoa-related studies report that e-nose systems can successfully discriminate cocoa bean quality, fermentation level, and storage conditions by analyzing volatile profiles generated during processing. In these works, sensor arrays combined with pattern-recognition or neural-network models were able to separate good-quality beans from defective ones, indicating that aroma fingerprints carry enough information to support objective grading. This evidence motivates the use of an e-nose in chocolate applications, where variations in fermentation, roasting, and formulation similarly produce distinctive volatile compositions that can be captured as sensor response patterns. Beyond cocoa, recent reviews on food quality monitoring highlight that electronic nose technology provides fast, non-destructive analysis and can be integrated with electronic tongue or vision systems for more comprehensive evaluation. These reviews also emphasize the need for robust signal processing pipelines and machine-learning-based classifiers to handle sensor drift, environmental influences, and complex mixtures of volatile organic compounds. Consequently, there is a strong demand for e-nose platforms that not only acquire sensor data reliably but also offer efficient backends and user-friendly interfaces to support practical deployment in food industries.

1.2 Formulation of the Problem

The problem formulation in this project is:

1. How to design and implement an electronic nose (eNose) system based on a gas sensor and Arduino Uno R4 WiFi to acquire aroma data on chocolate samples in real-time?
2. How to integrate a Rust-based backend with a Qt Python-based frontend so that eNose sensor signal data on chocolate samples can be displayed in an informative and user-friendly visualization?
3. How to store and process eNose measurement data on chocolate samples in dataset form (CSV/JSON), then visualize the sensor response pattern using GNUPLOT, and analyze the differences in aroma characteristics between chocolate samples?

1.3 Objective

The objectives of this project are:

1. Creating a GUI application for visualizing eNose data on chocolate testing that can help chocolate manufacturers in controlling the quality of their products.
2. Integrate a Rust backend with a Qt Python frontend to produce a fast, efficient, and easy-to-use application.
3. Providing a system that can analyze aroma data from eNose and visualize the results to help decision making in the chocolate production process.

1.4 Benefit

The development of this application is expected to provide the following benefits:

1. For the chocolate industry: Provides an effective tool to monitor and analyze the quality of chocolate products based on aroma, which is one of the determining factors of quality.
2. For consumers: Helps producers ensure that the chocolate products they produce have consistent quality, especially in terms of aroma.
3. For technology development: Contribute to the development of eNose technology and GUI-based applications that can be used in various other industries.

Chapter 2

Basic Theory

2.1 Electronic Nose for Chocolate Analysis

An electronic nose (e-nose) is a system that mimics the human sense of smell by using a gas sensor array and signal processing algorithms to identify volatile compound patterns in a sample. In food applications, e-noses are widely used for aroma-based product quality assessment because they provide a fast and non-destructive response compared to conventional chemical methods.

In chocolate products, aroma quality is strongly influenced by cocoa variety, fermentation, drying, and roasting processes, resulting in a complex volatile compound profile. The e-nose can record the sensor's response pattern to chocolate vapor and generate a multi-channel time-series signal representing changes in volatile compound concentration during the measurement process. This signal pattern can then be analyzed and visualized to differentiate cocoa quality categories or fermentation levels.

2.2 Gas Sensor for E-Nose System

E-nose systems typically use semiconductor, electrochemical, or conductive polymer gas sensor arrays, each with varying sensitivity to specific volatile compounds. Metal oxide semiconductor (MOS) sensors are widely used due to their relatively low cost, fast response, and ease of integration with simple electronic circuits.

Sensors such as the MICS-5524 and the Multichannel Gas Sensor v2 module are designed to detect a variety of gases, such as carbon monoxide (CO), nitrogen dioxide (NO2), ammonia (NH3), and other volatile organic compounds at low concentrations. In the context of chocolate samples, compounds from fermentation and roasting modify the gas composition around the sample, thereby affecting the conductivity of the sensor's sensitive element, which is then translated into changes in voltage or resistance. Important sensor characteristics for e-nose systems include sensitivity, selectivity, response time, long-term stability (drift), and the effects of temperature and humidity, all of which impact the quality of the resulting signal and require further processing.

2.3 Signal Processing on E-Nose Data

The primary output of the e-nose system is a time-series signal from each sensor channel, indicating the dynamics of the response to the sample odorant during the exposure and recovery phases. This signal typically contains noise, baseline differences between channels, and potential drift over the measurement period, requiring initial signal processing before further analysis.

Basic signal processing steps can include smoothing, normalization, and baseline correction to allow for more consistent comparison of responses between sensors. Additionally, simple feature extraction is often performed, such as the peak response value, the difference between the initial value and the steady-state value, the time to steady-state value, and the area under the curve, which are numerical representations of the sensor response pattern. These features facilitate pattern visualization and the application of classification methods to distinguish between different chocolate samples.

2.4 Data Acquisition and Communication

Data acquisition in the e-nose system is performed by reading the voltage or resistance output from the gas sensor array using a microcontroller, then periodically converting it into digital data. In this project, an Arduino Uno R4 WiFi is used as the main acquisition unit, which reads the sensor channels, regulates the sampling rate, and sends data to the host computer.

Sensor readings are sent via a communication interface, typically a serial port or Wi-Fi connection, in data packets containing time and value information from each sensor channel. On the computer side, a backend written in Rust receives these data streams, parses them, and structures them for use by the processing and visualization modules. This approach supports a clear separation between the data acquisition layer and the user interface layer, while also fostering an understanding of the flow from sensors to the digital signal processing system.

2.5 Data Visualization and GNUPLOT

Data visualization is a crucial step in understanding the e-nose sensor's response patterns to various samples and measurement conditions. Time-series graphs displaying the relationship between time and the response value of each sensor channel can help observe the response rise phase, steady state, and recovery process after sample removal. Different curve shapes between samples are an early indication of differences in aroma characteristics.

GNUPLOT is a widely used open-source software for visualizing scientific and engineering data, including sensor measurement data. By utilizing datasets stored in CSV or JSON format, GNUPLOT can be used to plot sensor response curves for multiple chocolate types in a single graph, allowing for visual comparison of response patterns between samples. This visualization supports advanced analysis, both qualitatively by the user and as a basis for feature exploration for machine learning modeling.

2.6 Rust Backend and Qt Python GUI Integration

In modern engineering application development, separating the backend and frontend is a common approach to improve modularity, performance, and ease of interface development. The backend is responsible for tasks that require high efficiency and tight resource management, such as communicating with hardware, processing data, and managing connections, while the frontend focuses on presenting information and interacting with users.

Rust programming language was chosen for this project to build the backend because it offers performance close to low-level languages with strong memory safety guarantees, making it suitable for handling sensor data processing and real-time communication. On the other hand, Qt Python provides a productive framework for designing interactive graphical interfaces, including real-time graphical displays, numeric indicators, and input forms for the names and types of chocolate samples. Integration between Rust and Qt Python allows data received by the backend from the Arduino Uno R4 WiFi to be passed in a structured manner to the GUI application, so users can monitor the e-nose response directly and control the sampling process through an easy-to-use interface.

Chapter 3

System Design

3.1 Architecture and Data Flow

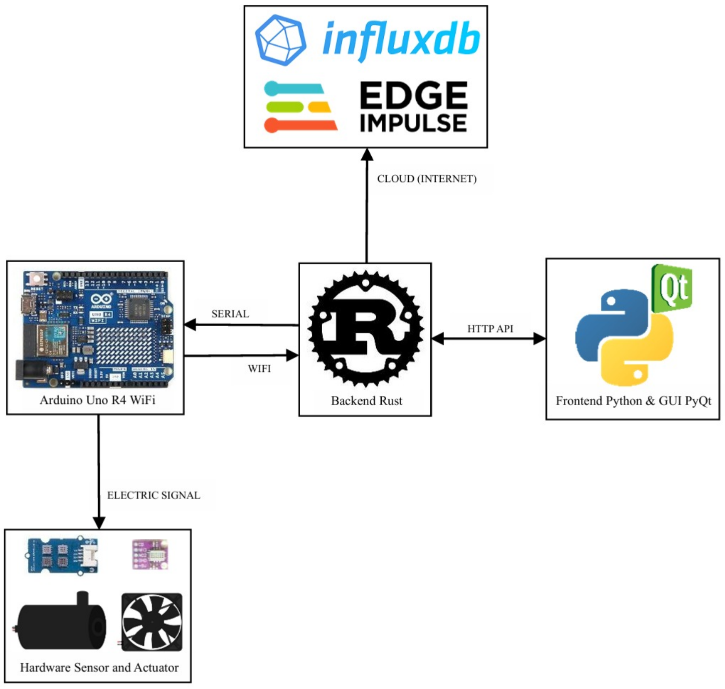


Figure 3.1 Arsitektur System

The eNose system architecture for chocolate samples is built from several main components: eNose hardware (sample chamber, sensors, pump, fan), Arduino Uno R4 WiFi, Rust-based backend, InfluxDB database, Qt Python GUI application, and integration with Edge Impulse and GNUPLOT. The aroma signal from the chocolate sample is first flowed by the pump and fan through the gas sensor array, resulting in an analog electrical signal representing the concentration of volatile compounds. The Arduino Uno R4 WiFi reads this analog signal, performs analog-to-digital conversion, and periodically sends the data via serial communication or WiFi to the Rust backend on the host computer.

The Rust backend receives data packets, performs parsing and format checking, and then saves the data to InfluxDB as a time-series database labeled with the name and type of the chocolate sample. At the same time, the backend provides an HTTP/REST endpoint accessible by a Qt Python GUI to display real-time graphs of sensor responses and numerical values. The stored dataset can then be exported to CSV/JSON files for visualization using GNUPLOT and uploaded to Edge Impulse as training data for machine learning models, completing the complete data flow from sensors to advanced analysis.

3.2 Sampling Procedure and Sample Input

The sampling procedure begins with preparing the chocolate sample according to the variety to be tested (e.g., brand, composition, or specific processing level) and placing the sample in the eNose sample chamber. Before the measurement begins, the user opens the Qt Python GUI application and fills in an input form containing the sample name, chocolate sample type, and other necessary measurement session information. Then, the user presses the start button to begin the sampling process.

During the session, the pump and fan are activated to circulate chocolate-scented air past the sensor, increasing the sensor response to a steady state, while the Arduino sends data to the backend and the GUI displays a real-time graph of the signal-time for each sensor channel. Measurements are maintained for a specified duration to establish a clear pattern of the sensor's response to the chocolate samples, then the session is stopped via the GUI and the dataset is automatically or manually saved to InfluxDB and exported to a CSV/JSON file with a filename containing the sample ID and measurement time.

3.3 3D Design of the eNose System

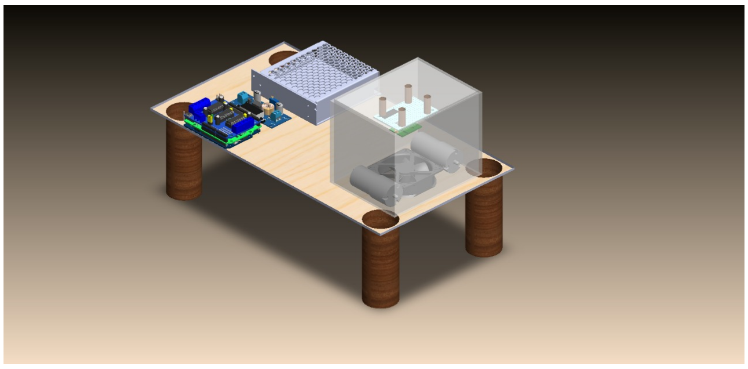


Figure 3.2 Side View of the 3D Design of the eNose system

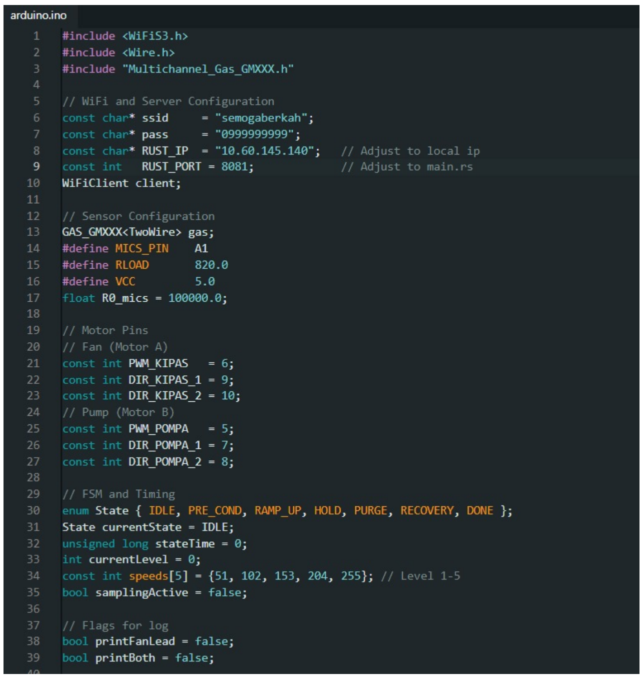
The 3D design of the eNose system was created to realize the mechanical layout of the sample chamber, gas sensor array, pump, fan, and placement of the Arduino Uno R4 WiFi and supporting circuits in a compact case. The 3D model was designed using CAD software so that the dimensions of the sample chamber and airflow path could be adjusted so that the gas flow from the chocolate sample passes through the sensor evenly with minimal leakage, while remaining easy to open for sample placement and replacement.

The 3D design also considered the position of the sensor relative to the airflow, the mounting for the electronic module, and the cable routing and connection ports to facilitate maintenance and further development. The pump and fan were positioned to reduce vibration and noise that could affect sensor readings, while the outer casing was designed to be stable on a laboratory bench and provide convenient access to the sample chamber and user interface.

Chapter 4

System Implementation

4.1 Arduino Implementation



arduino.ino

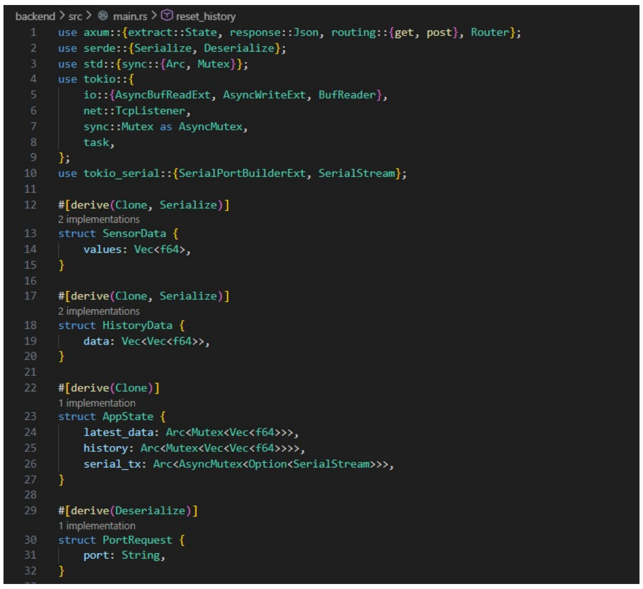
```
#include <WiFiS3.h>
#include <Wire.h>
#include "Multichannel_Gas_GMXXX.h"

// WiFi and Server Configuration
const char* ssid     = "semogaberkah";
const char* pass     = "0999999999";
const char* RUST_IP  = "10.60.145.140";   // Adjust to local ip
const int   RUST_PORT = 8081;             // Adjust to main.rs
WiFiClient client;

// Sensor Configuration
GAS_GMXXX<TwoWire> gas;
#define MICS_PIN    A1
#define RLOAD       820.0
#define VCC         5.0
float R0_mics = 100000.0;

// Motor Pins
// Fan (Motor A)
const int PWM_KIPAS   = 6;
const int DIR_KIPAS_1 = 9;
const int DIR_KIPAS_2 = 10;
// Pump (Motor B)
const int PWM_POMPA   = 5;
const int DIR_POMPA_1 = 7;
const int DIR_POMPA_2 = 8;

// FSM and Timing
enum State { IDLE, PRE_COND, RAMP_UP, HOLD, PURGE, RECOVERY, DONE };
State currentState = IDLE;
unsigned long stateTime = 0;
int currentLevel = 0;
const int speeds[5] = {51, 102, 153, 204, 255}; // Level 1-5
bool samplingActive = false;

// Flags for log
bool printFanLead = false;
bool printBoth = false;
```

Figure 4.1 implementation arduino

At this stage, the Arduino Uno R4 WiFi serves as the primary data acquisition unit, connecting the gas sensor array to the computer system. Each gas sensor channel is connected to an Arduino analog input pin, so changes in the concentration of volatile compounds in the chocolate sample are read as changes in ADC values. The Arduino program is developed to perform periodic readings on several sensor channels at a specific sampling rate, then package the readings and time information into a consistent data packet format.

Data is sent to the host computer via the selected interface (serial or WiFi) using a simple protocol consisting of a string of numeric values separated by a specific delimiter, making it easy for the Rust backend to parse. The program also configures control signals to activate or deactivate the pump and fan during the measurement session, allowing the airflow cycle in the sample chamber to be repeated consistently for each experiment.

4.2 Rust Backend Implementation



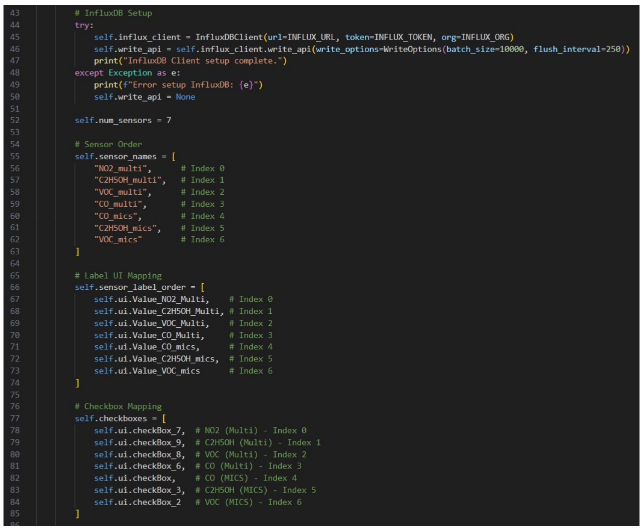
backend > src > main.rs > reset_history

```rust
use axum::{extract::State, response::Json, routing::{get, post}, Router};
use serde::{Serialize, Deserialize};
use std::{sync::{Arc, Mutex}};
use tokio::{
    io::{AsyncBufReadExt, AsyncWriteExt, BufReader},
    net::TcpListener,
    sync::Mutex as AsyncMutex,
    task,
};
use tokio_serial::{SerialPortBuilderExt, SerialStream};

#[derive(Clone, Serialize)]
2 implementations
struct SensorData {
    values: Vec<f64>,
}

#[derive(Clone, Serialize)]
2 implementations
struct HistoryData {
    data: Vec<Vec<f64>>,
}

#[derive(Clone)]
1 implementation
struct AppState {
    latest_data: Arc<Mutex<Vec<f64>>>,
    history: Arc<Mutex<Vec<Vec<f64>>>>,
    serial_tx: Arc<AsyncMutex<Option<SerialStream>>>,
}

#[derive(Deserialize)]
1 implementation
struct PortRequest {
    port: String,
}
```

Figure 4.2 implementation backend

The system backend is developed using Rust to handle communication with the Arduino, initial processing of sensor data, and integration with the InfluxDB time-series database. The serial communication module in Rust is configured to open the port used by the Arduino, read the incoming data stream continuously, and parse it according to the packet format, for example a timestamp followed by the values of each sensor channel. Values that are successfully parsed are stored in an internal data structure and then written to InfluxDB as time-series records with tags that include the name and type of the chocolate sample. In addition to storage functions, the backend provides HTTP/REST services that allow Qt Python GUI applications to retrieve data in real-time or based on specific timeframes. The API endpoint is designed to be lightweight so that it can return data in JSON format that is easy to process on the frontend, for example to draw live graphs and display the latest numeric values. The backend implementation also includes a function to export datasets from InfluxDB to CSV or JSON files for visualization in GNUPLOT and model training in Edge Impulse.

4.3 Qt Python GUI Implementation



```python
        # InfluxDB Setup
        try:
            self.influx_client = InfluxDBClient(url=INFLUX_URL, token=INFLUX_TOKEN, org=INFLUX_ORG)
            self.write_api = self.influx_client.write_api(write_options=WriteOptions(batch_size=10000, flush_interval=250))
            print("InfluxDB Client setup complete.")
        except Exception as e:
            print(f"Error setup InfluxDB: {e}")
            self.write_api = None

        self.num_sensors = 7

        # Sensor Order
        self.sensor_names = [
            "NO2_multi",      # Index 0
            "C2H5OH_multi",   # Index 1
            "VOC_multi",      # Index 2
            "CO_multi",       # Index 3
            "CO_mics",        # Index 4
            "C2H5OH_mics",    # Index 5
            "VOC_mics"        # Index 6
        ]

        # Label UI Mapping
        self.sensor_label_order = [
            self.ui.Value_NO2_Multi,    # Index 0
            self.ui.Value_C2H5OH_Multi, # Index 1
            self.ui.Value_VOC_Multi,    # Index 2
            self.ui.Value_CO_Multi,     # Index 3
            self.ui.Value_CO_mics,      # Index 4
            self.ui.Value_C2H5OH_mics,  # Index 5
            self.ui.Value_VOC_mics      # Index 6
        ]

        # Checkbox Mapping
        self.checkboxes = [
            self.ui.checkBox_7,  # NO2 (Multi) - Index 0
            self.ui.checkBox_9,  # C2H5OH (Multi) - Index 1
            self.ui.checkBox_8,  # VOC (Multi) - Index 2
            self.ui.checkBox_6,  # CO (Multi) - Index 3
            self.ui.checkBox,    # CO (MICS) - Index 4
            self.ui.checkBox_3,  # C2H5OH (MICS) - Index 5
            self.ui.checkBox_2   # VOC (MICS) - Index 6
        ]
```

Figure 4.3 implementation frontend

The system graphical interface is implemented using Qt Python, allowing users to interactively control the measurement process and monitor the e-nose response to chocolate samples. The main GUI provides input forms for the sample name and sample type, buttons to start and stop a measurement session, and a graph area that displays the sensor response curve in real time. The graph is updated by sending periodic requests to the backend HTTP endpoint and plotting the latest values for each sensor channel.

In addition to graphs, the GUI displays numeric indicators to show the current value of each sensor channel and the system connection status (Arduino, backend, and database). Additional features are provided to save the measurement session dataset to a CSV/JSON file with a filename containing the sample ID and measurement time, as well as a button to export the data to a format ready for use in GNUPLOT or upload to Edge Impulse. Thus, the Qt Python GUI implementation completes the sensor–backend–visualization integration so that the entire workflow from data acquisition to further analysis can be done through a single application.

4.4 Edge Impulse Implementation

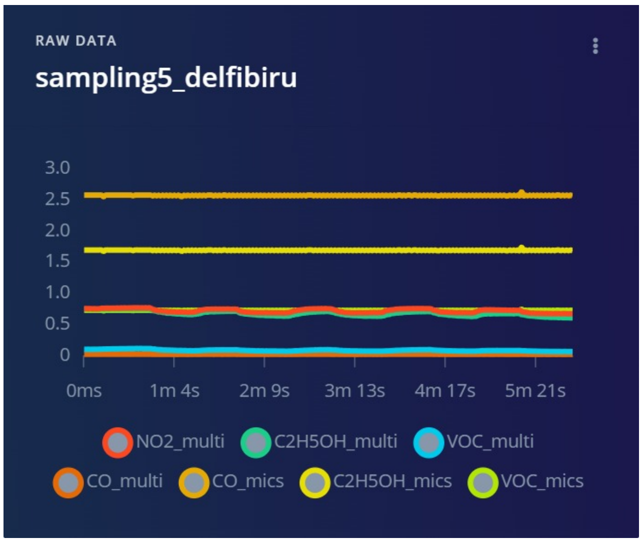


Figure 4.4 implementation Edge Impulse

The integration with Edge Impulse begins with preparing a dataset of eNose measurements saved in CSV or JSON format from a Qt Python GUI application. Each file contains information on the measurement time, the response value of each sensor channel, and a class label representing the type or quality of the sample being tested, for example, several categories of chocolate. The dataset is then uploaded to the Edge Impulse project via a web interface, and each sample is given a consistent label to ensure proper classification model training. Within Edge Impulse, signal processing and machine learning blocks are configured to suit the characteristics of the eNose data, such as selecting a time window, extracting statistical features, and selecting a classification algorithm. After the training and validation processes are complete, Edge Impulse generates performance metrics such as accuracy and a confusion matrix, which are used to evaluate the model's ability to distinguish sensor response patterns between sample classes. The best model is then exported in a format provided by Edge Impulse so that in the next development phase it can be integrated into the eNose system to automatically predict class patterns for new sensor data.

4.5 Influx DB Implementation

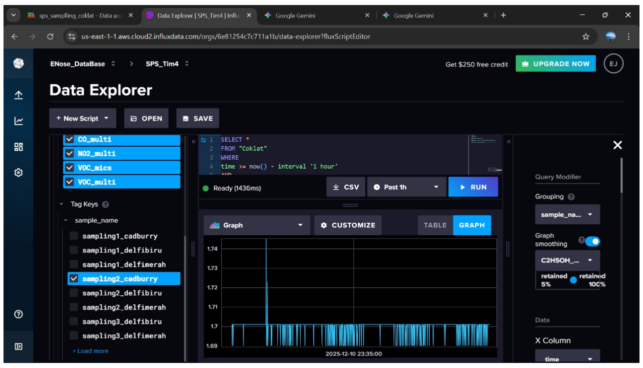

```
SELECT *
FROM "Coklat"
WHERE
time >= now() - interval '1 hour'
```

Figure 4.5 implementation Influx DB

In this project, InfluxDB is implemented as the main time-series database to store the multi-channel sensor signals generated by the eNose system. Each data point recorded by the Arduino Uno R4 WiFi and processed by the Rust backend is written to InfluxDB together with timestamps and tags that describe the chocolate sample name, type, and test level. This structure allows efficient querying of sensor responses over specific time windows and makes it easy to group or compare different chocolate samples based on their aroma profiles. InfluxDB also supports fast data export to CSV or JSON, so the stored datasets can be reused for visualization with GNUPLOT or for training machine-learning models on platforms such as Edge Impulse.

Chapter 5

Results and Discussion

5.1 Test Results

Here is some of the results that obtained after conducting the test. The following graph results can be seen in the attachment section and on the Author's GitHub. GitHub can be accessed via the following link: `https://github.com/jaaviier11/E-Nose-System-with-Integration-of-Rust-and-Python`

5.1.1 Cadbury Chocolate Test Results

Figure 5.1 shows a sensor response graph for the Cadbury chocolate sample, showing a relatively stable pattern with moderate fluctuations in several key channels. This indicates that the volatile compound composition in Cadbury is quite homogeneous, with moderate aroma intensity, so the curve does not show extreme spikes.

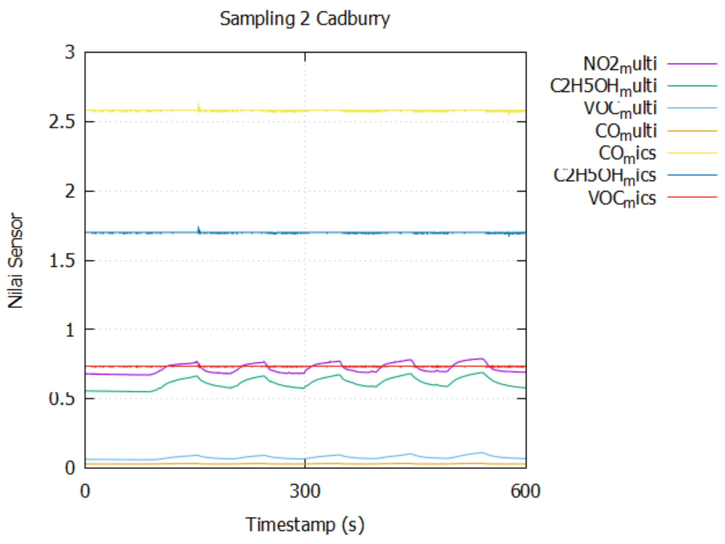


Figure 5.1.2 Cadburry Chocolate Test Results

5.1.2 Delfi Red Chocolate Test Results

Figure 5.2 shows the sensor response. In the Red Delphi chocolate sample, the sensor response amplitude tends to be lower but the fluctuations between cycles are tighter. This pattern depicts a scent that is not as strong as Cadburry, but still contains a fairly complex mix of volatiles so that the graph appears more noisy and wavy.

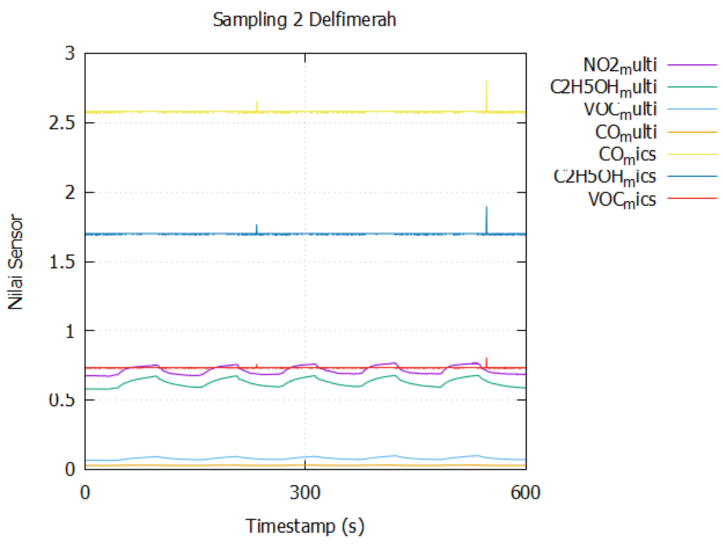


Figure 5.1.2 Delfi Red Chocolate Test Results

5.1.3 Delfi Blue Chocolate Test Results

Figure 5.3 shows the sensor response for Delfi Blue chocolate, which exhibits several pulses with more pronounced peaks in certain channels. These peaks indicate the presence of more dominant aroma components, making its aroma fingerprint more contrasting than Delfi Red and potentially more easily distinguished by classification algorithms.

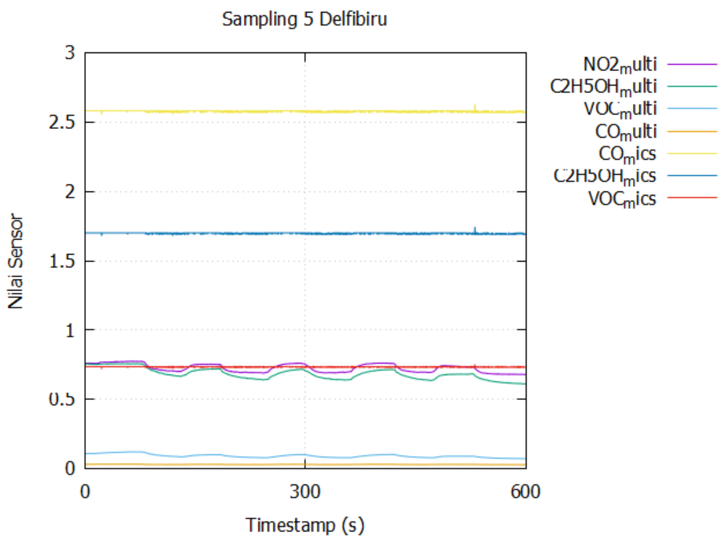


Figure 5.1.3 Delfi Blue Chocolate Test Results

5.1.4 Silverqueen Chocolate Test Results

Figure 5.4 shows the sensor response for the Silverqueen chocolate sample. The graph shows a curve with the smallest amplitude and a relatively flat shape. This condition indicates that the concentration of volatile compounds detected by the sensor is lower, resulting in a lighter measured aroma and a narrower aroma fingerprint response range compared to the other three chocolates.

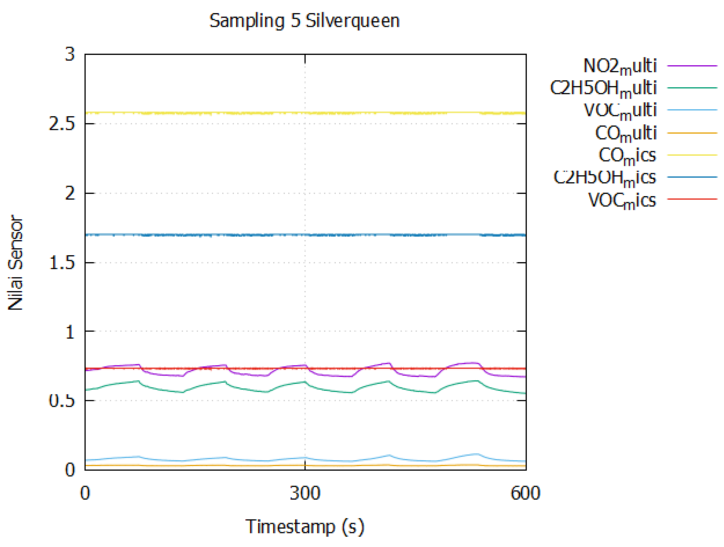


Figure 5.1.4 Silverqueen Chocolate Test Results

5.2 Discussion

The comparative analysis of the sensor responses highlights distinct “aroma fingerprints” for each chocolate brand, governed by the concentration and complexity of their volatile organic compounds (VOCs). The results indicate a clear hierarchy in signal intensity and pattern stability.

Delfi Blue exhibits the most distinguishable profile, characterized by pronounced peaks (Figure 5.1.3) that suggest the presence of dominant specific aroma components. This high-contrast signature implies that Delfi Blue provides the strongest feature set for classification, making it potentially the easiest sample to distinguish using pattern recognition algorithms. Conversely, Silverqueen (Figure 5.1.4) presents the minimal boundary of detection; its low amplitude and flat response reflect a low concentration of detectable volatiles, resulting in a significantly lighter aroma profile.

Between these extremes, Cadbury and Delfi Red demonstrate moderate responses but differ significantly in signal topology. Cadbury displays a stable, homogeneous composition, whereas Delfi Red—despite having a lower amplitude than Cadbury—reveals a complex, “noisy” volatile mix characterized by tight fluctuations. This variation confirms that the sensor array is sensitive not only to the overall intensity (amplitude) but also to the qualitative complexity (fluctuation patterns) of the chocolate samples.

Chapter 6

Closing

6.1 Conclusion

This project successfully developed an electronic nose system for chocolate aroma analysis by integrating a gas sensor, Arduino Uno R4 WiFi, a Rust-based backend, and a Qt Python interface capable of real-time data acquisition and visualization. The system was able to read multi-channel signals, perform preprocessing such as filtering to reduce noise, and display sensor responses in the form of stable time-series graphs, making it easier to monitor aroma dynamics during the testing process. Tests on several chocolate samples showed that each sample produced a different sensor response pattern, both in terms of peak amplitude and waveform, thus revealing a unique aroma fingerprint for each type of chocolate. These results prove that the developed eNose system not only functions well in terms of hardware and software integration, but also has adequate discrimination capabilities against differences in chocolate aroma characteristics, making it suitable as a basis for further development using machine learning-based classification methods.

6.2 Recommendation

Based on the analysis results and the limitations still identified in the e-nose system for chocolate, several further development steps are needed to improve system performance and prepare it for wider application. The following recommendations are designed to strengthen the data, algorithms, and hardware and software design.

- Increase the number and variety of chocolate samples (different brands, cocoa content, roasting/fermentation levels) to enrich the dataset so that the chocolate quality classification model that will be trained on the machine learning platform becomes more accurate and robust.
- Fully integrate a pipeline from chocolate eNose data to Edge Impulse or a similar platform to build and test automated chocolate grade or type classification models.
- Adding environmental sensors such as temperature and humidity and applying corrections or normalization, so that the influence of room conditions on the sensor's response to chocolate can be compensated for and measurement results are more consistent.
- Developing a mechanical/3D design of the sample chamber so that the air flow around the chocolate is more uniform and the purge process is faster, so that repeating measurements between chocolate samples becomes more repeatable and the experiment time is more efficient.

Bibliografi

[1] N. Saiz-Rodríguez, M. Marín-Sáez, A. Aparicio *et al.*, “Applications of electronic nose (e-nose) and electronic tongue (e-tongue) in food quality-related properties determination: A review,” *Computers and Electronics in Agriculture*, vol. 183, p. 106074, 2021.

[2] M. Roy, S. Bhattacharyya, and S. R. Chandra, “Electronic nose for detection of food adulteration: A review,” *Critical Reviews in Food Science and Nutrition*, vol. 61, no. 7, pp. 1234–1252, 2021.

[3] E. A. de Araújo, J. F. de Oliveira, and M. A. Lopes, “Electronic Nose Coupled with Linear and Nonlinear Supervised Classification Methods for Cocoa Bean Quality Assessment,” *International Journal of Intelligent Engineering and Systems*, vol. 12, no. 6, pp. 176–187, 2019.

[4] M. S. Haiying, L. Zhang, and H. Wang, “Sensing fermentation degree of cocoa (Theobroma cacao L.) beans by machine learning classification models based electronic nose system,” *Journal of Food Process Engineering*, vol. 44, no. 10, e13875, 2021.

[5] A. F. M. Noor, M. F. Abdullah, and M. I. Ibrahim, “Intelligent Classification of Cocoa Bean using E-nose,” *Mekatronika*, vol. 5, no. 2, pp. 12–20, 2023.

[6] Malaysian Cocoa Board, “Malaysian Cocoa Journal 2023, Vol. 15(2),” *Malaysian Cocoa Journal*, vol. 15, no. 2, 2023.

[7] I. A. Hidayat, A. P. Putra, and A. A. Asyhar, “Klasifikasi Mutu Biji Kakao berbasis Data Electronic Nose menggunakan Metode Jaringan Syaraf Tiruan,” *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 5, no. 7, pp. 2760–2769, 2021.

[8] A. Nugroho, R. Suryani, and F. Rahma, “Electronic nose usage on pattern recognition of volatile organic compounds in food products,” *Food Research*, vol. 8, no. 4, pp. 100–110, 2024.

[9] A. R. de Oliveira, F. Silva, and R. Costa, “Applications of Electronic Nose, Electronic Eye and Electronic Tongue in Food Quality Evaluation,” *Frontiers in Nutrition*, vol. 10, 1080401, 2023.

[10] Rust Foundation, “Best Practices for Integrating Rust and Qt in Embedded Systems,” Rust Foundation Blog, 2023.

Attachment

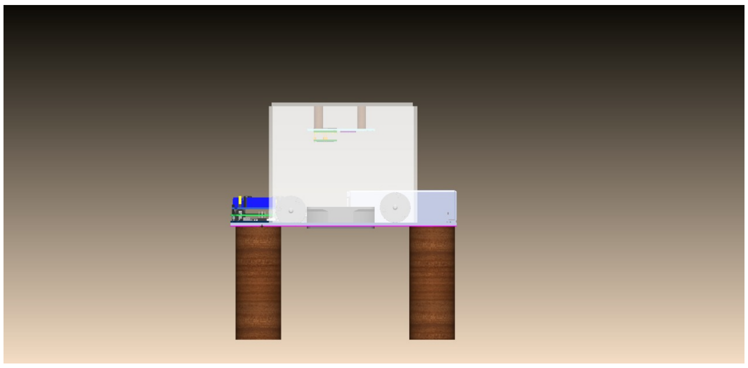


Figure 1: Side View of the 3D Design of the eNose system

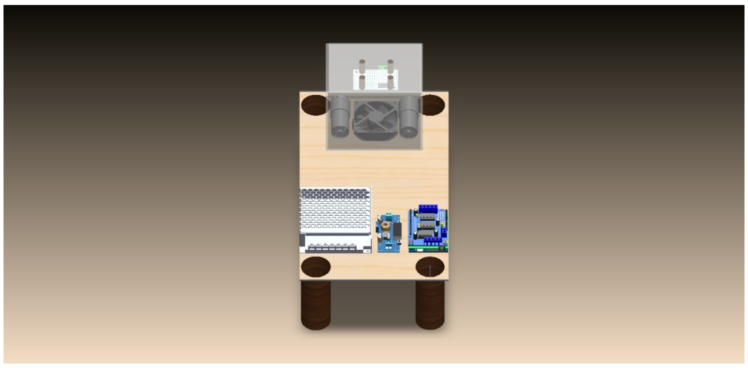


Figure 2: Front View of the 3D Design of the eNose system

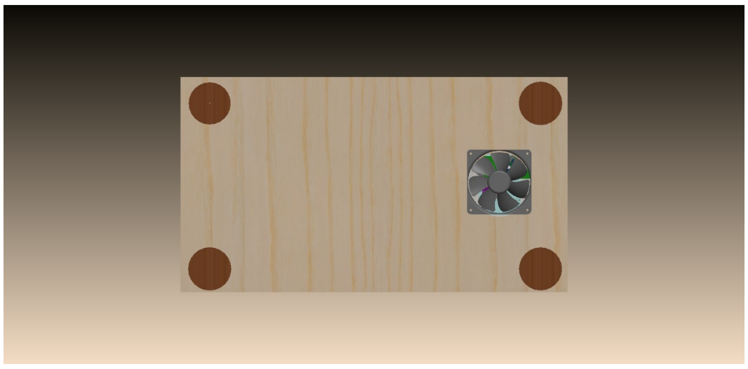


Figure 3: Bottom View 3D Design of the eNose system.

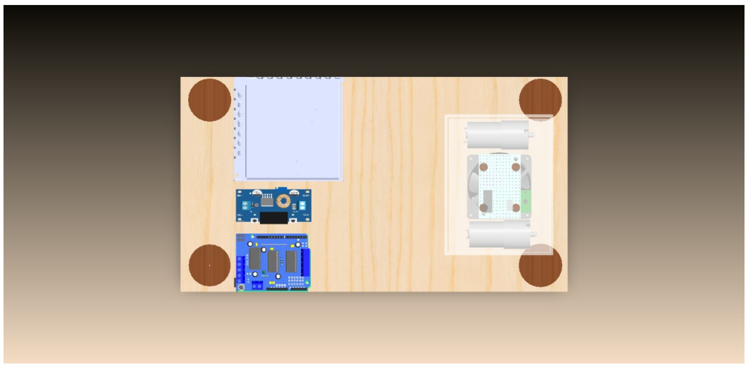


Figure 4: Top View 3D Design of the eNose system.

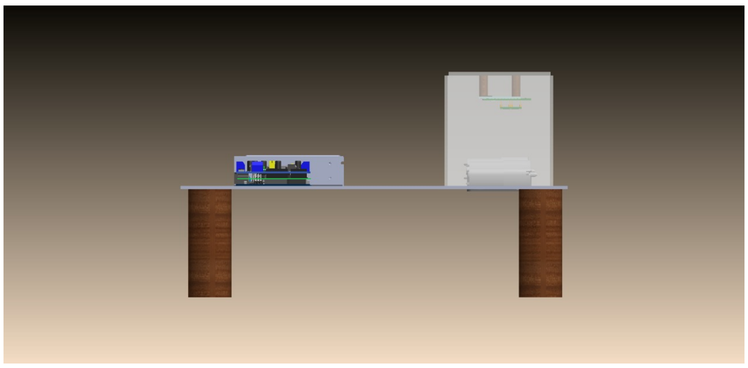


Figure 5: Side View of the 3D Design of the eNose system

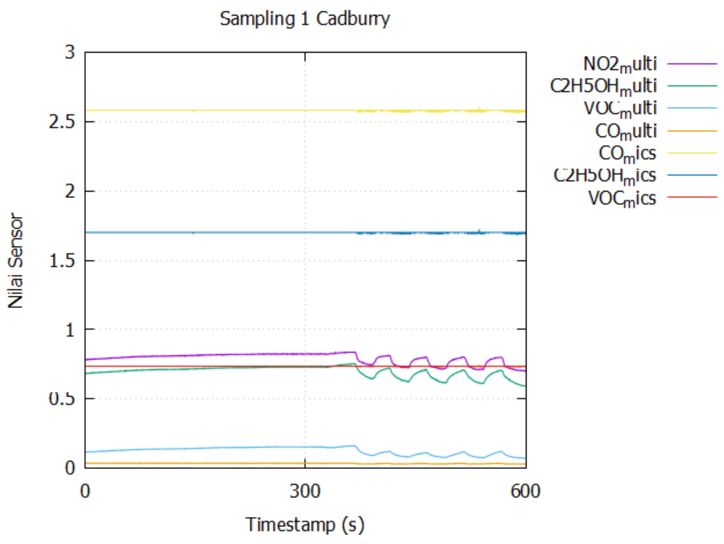


Figure 6: grafik sampling1 cadburry

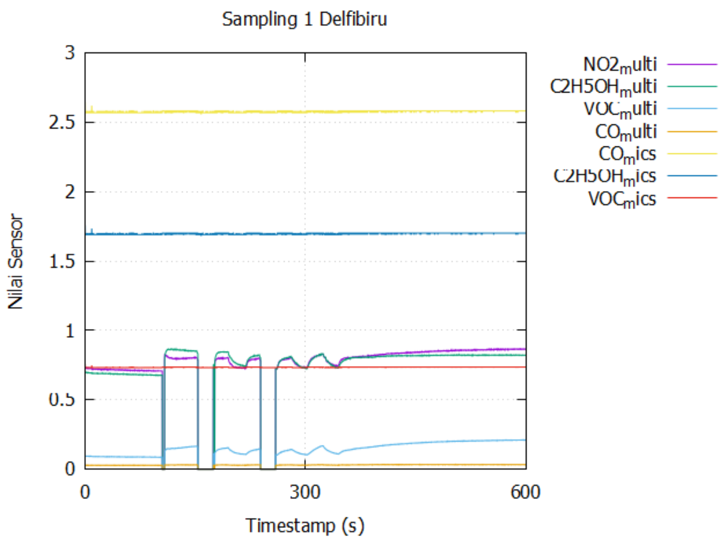


Figure 7: grafik sampling1 delfibiru

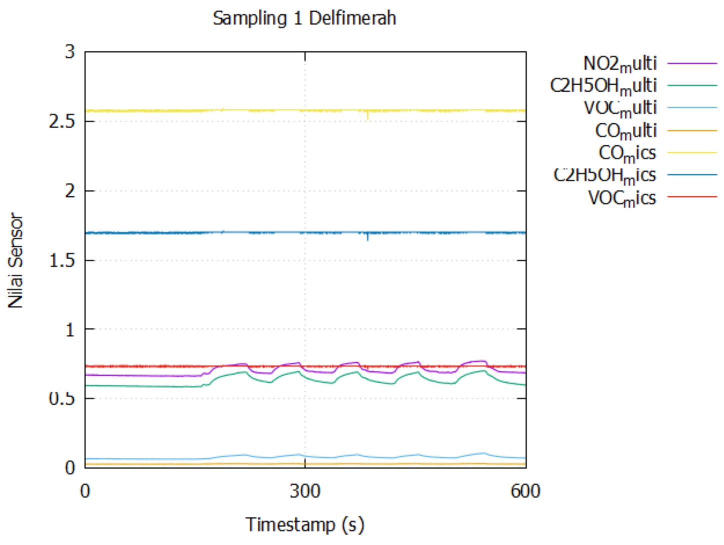


Figure 8: grafik sampling1 delfimerah

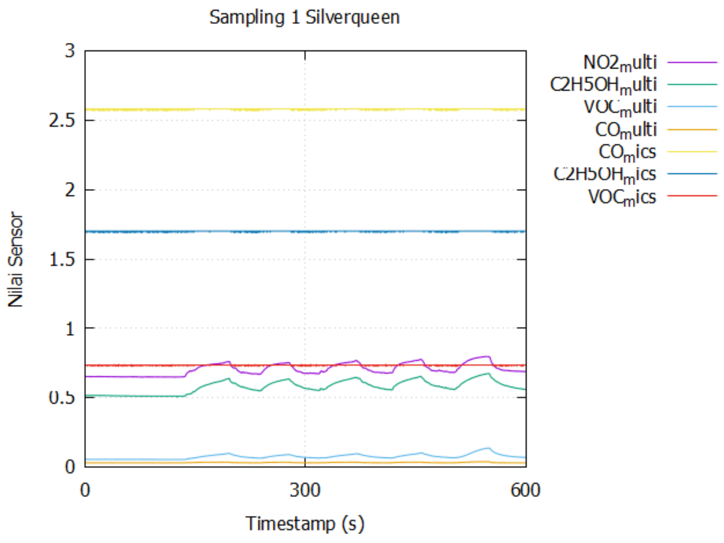


Figure 9: grafik sampling1 silverqueen

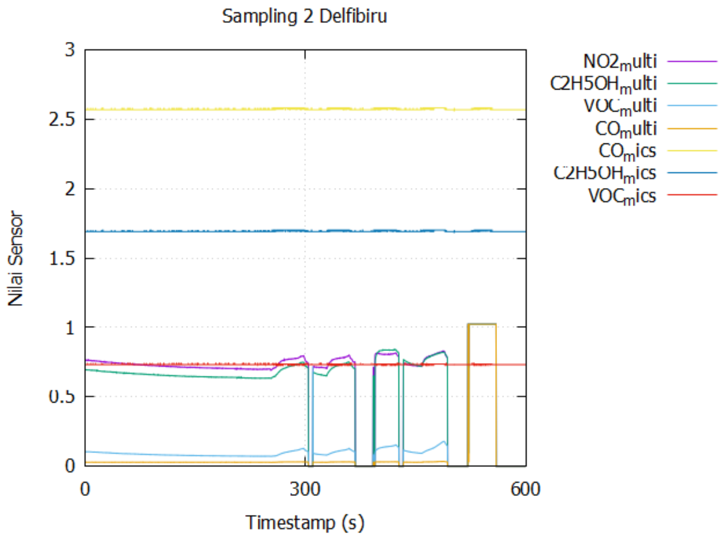


Figure 10: grafik sampling2 delfibiru

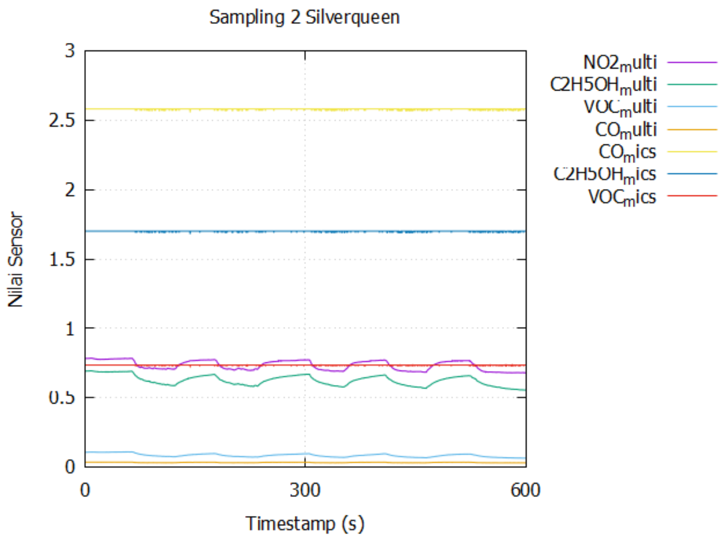


Figure 11: grafik sampling2 silverqueen

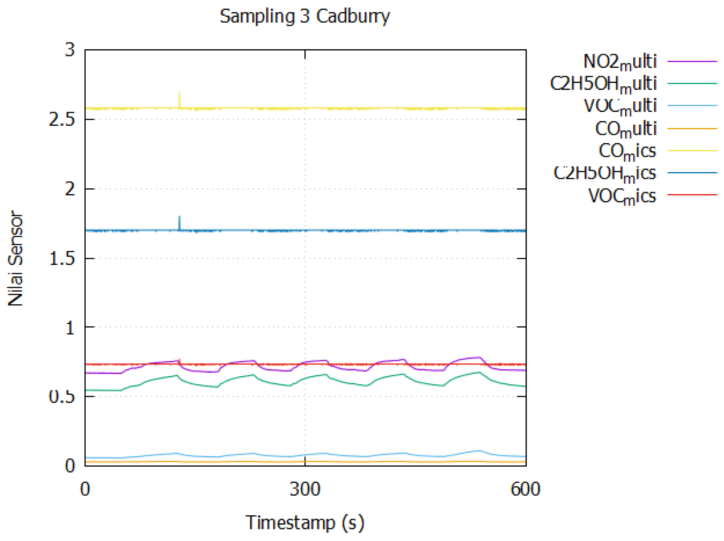


Figure 12: grafik sampling3 cadburry

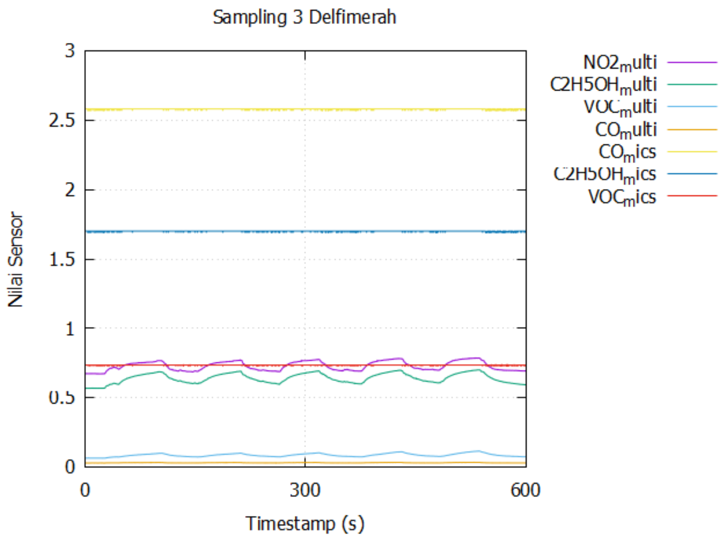


Figure 13: grafik sampling3 delfimerah

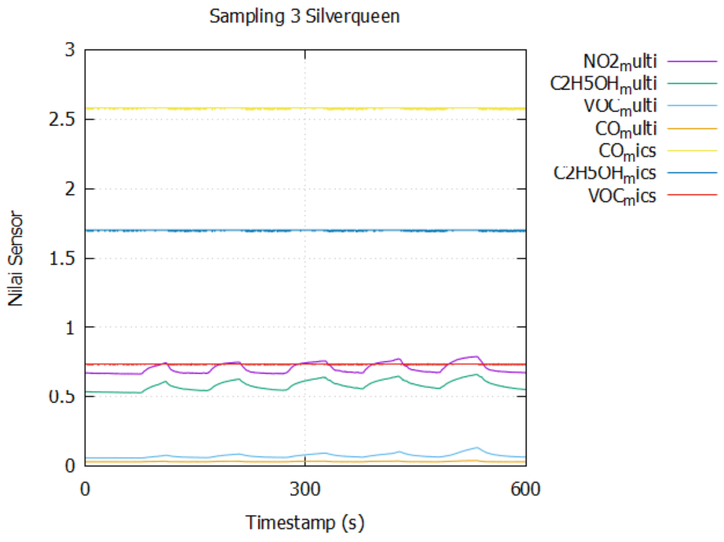


Figure 14: grafik sampling3 silverqueen

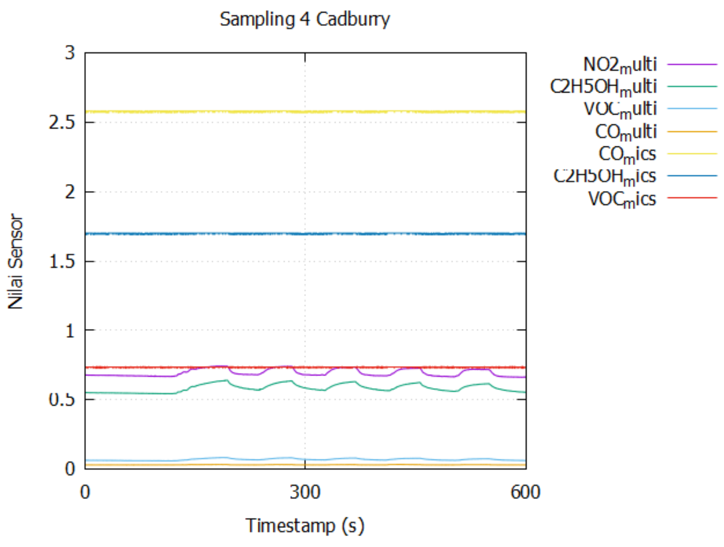


Figure 15: grafik sampling4 cadburry

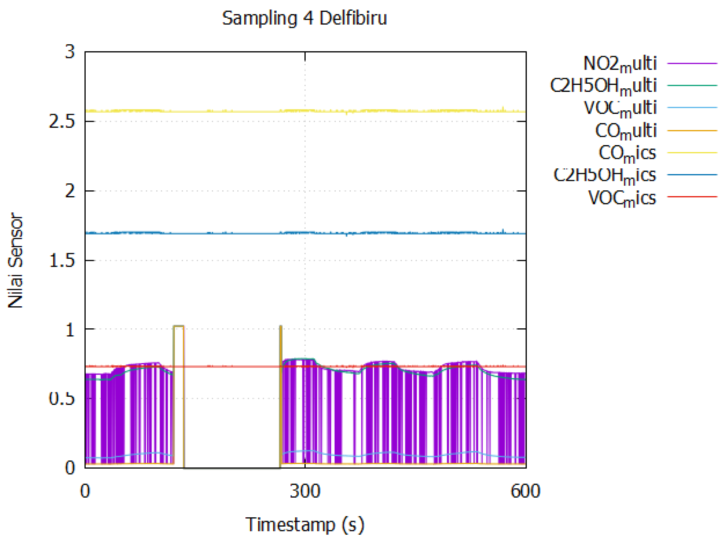


Figure 16: grafik sampling4 delfibiru

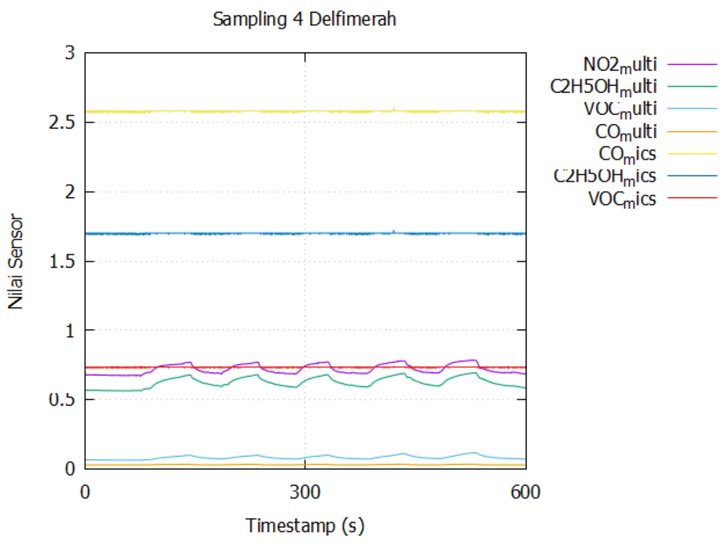


Figure 17: grafik sampling4 delfimerah

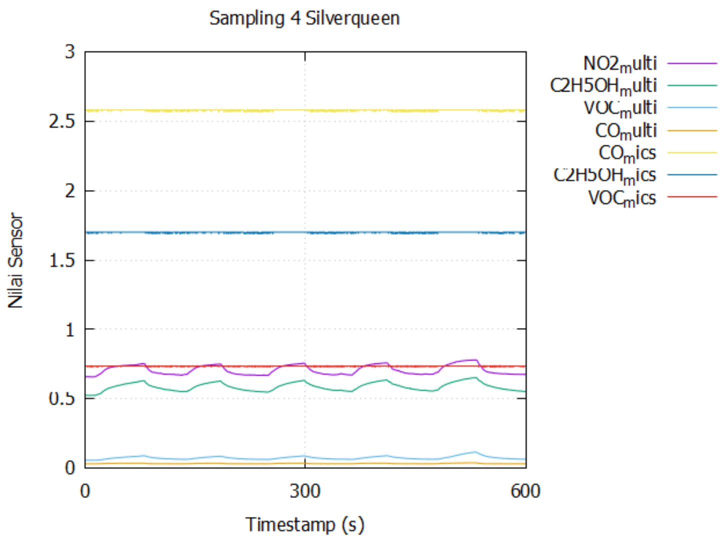


Figure 18: grafik sampling4 silverqueen

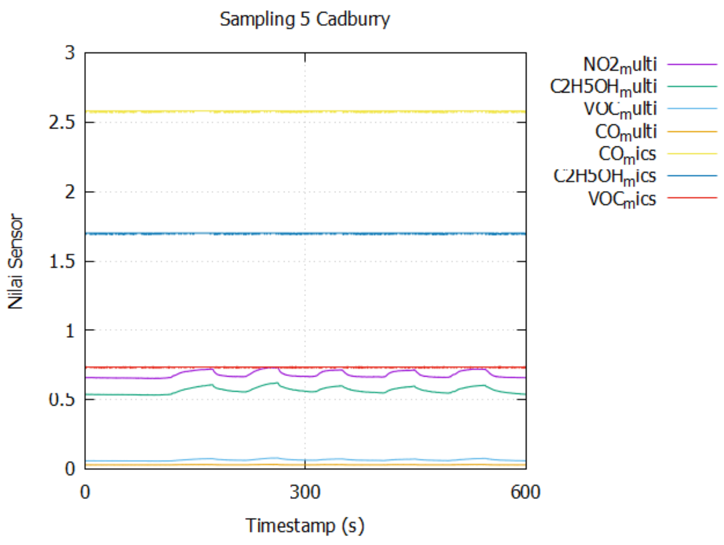


Figure 19: grafik sampling5 cadburry

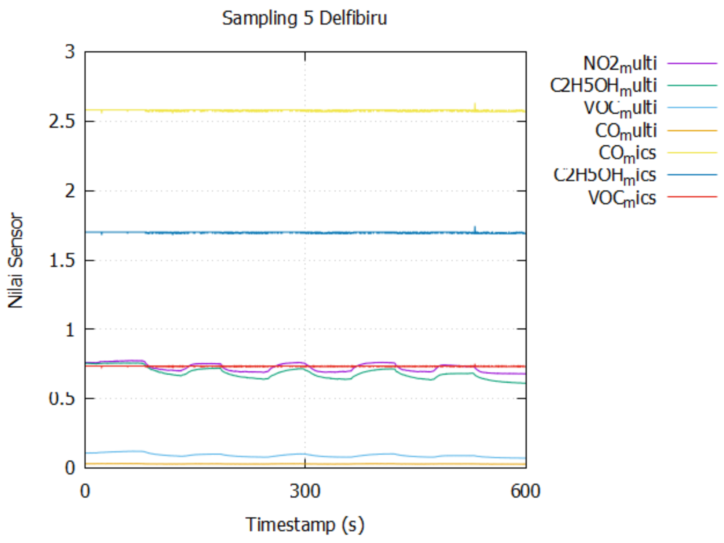


Figure 20: grafik sampling5 delfibiru

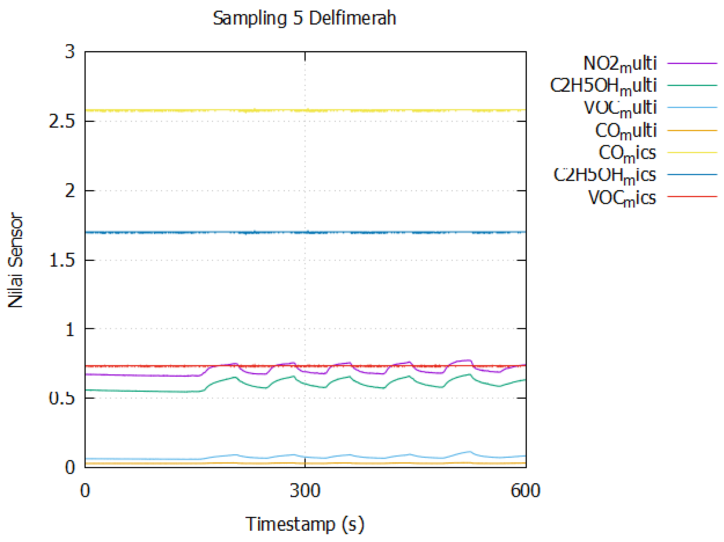


Figure 21: grafik sampling5 delfimerah