GUI APPLICATION

DEVELOPMENT FOR ELECTRONIC NOSE DATA VISUALIZATION

SISTEM PENGOLAHAN SINYAL

OUR TEAM

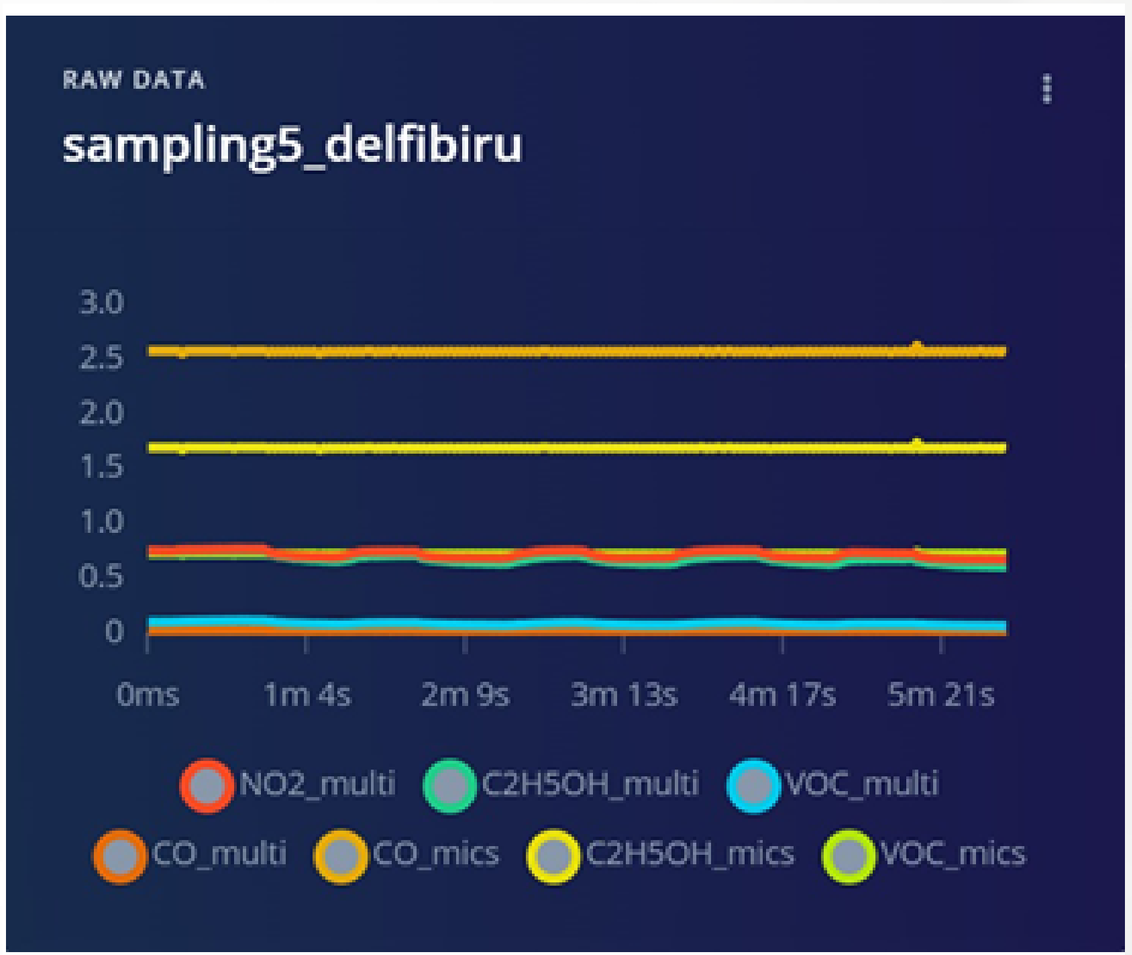
Evan Javier Firdausi Malik
2042241010

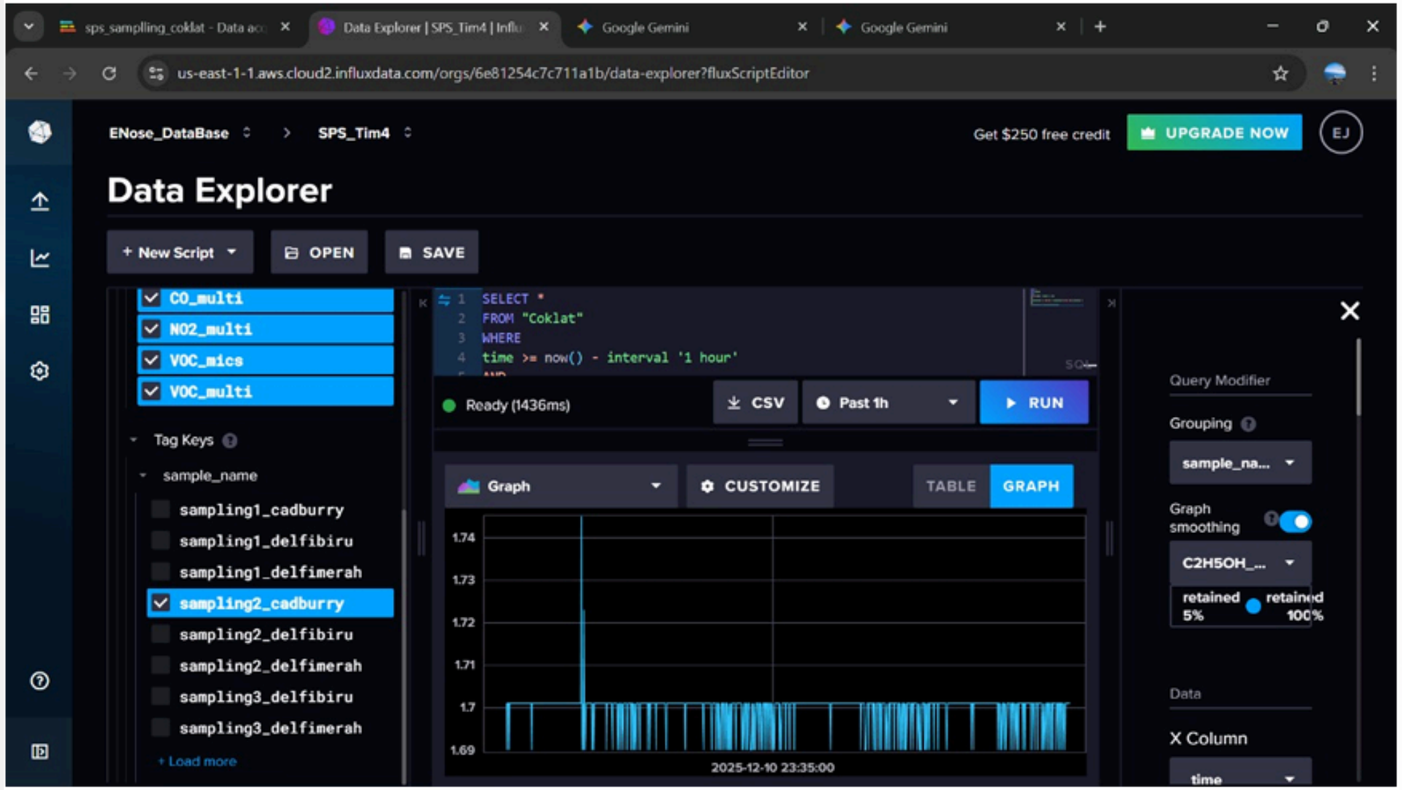
Rahma Aulia
2042241056

Farel Alberto Firest
2042241058

BACKGROUND AND OBJECTIVES

In the chocolate industry, **aroma is a critical indicator of quality, determined by factors such as fermentation and roasting**. Traditional methods relying on human panels can be inconsistent and slow. Therefore, this project aims to design an **electronic nose system capable of acquiring real-time aroma data from chocolate samples**. The primary objective is to integrate a high-performance Rust-based backend with a user-friendly Qt Python frontend to visualize sensor response patterns. This system is designed to help **manufacturers monitor product consistency and control quality through objective data analysis.**

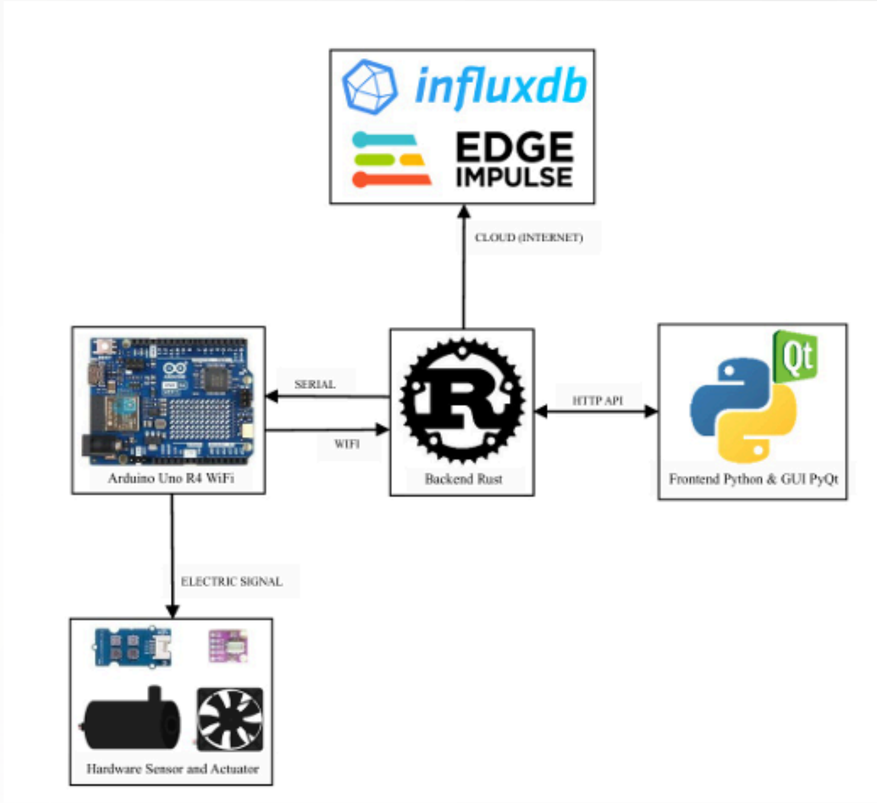




THEORETICAL **FRAMEWORK**

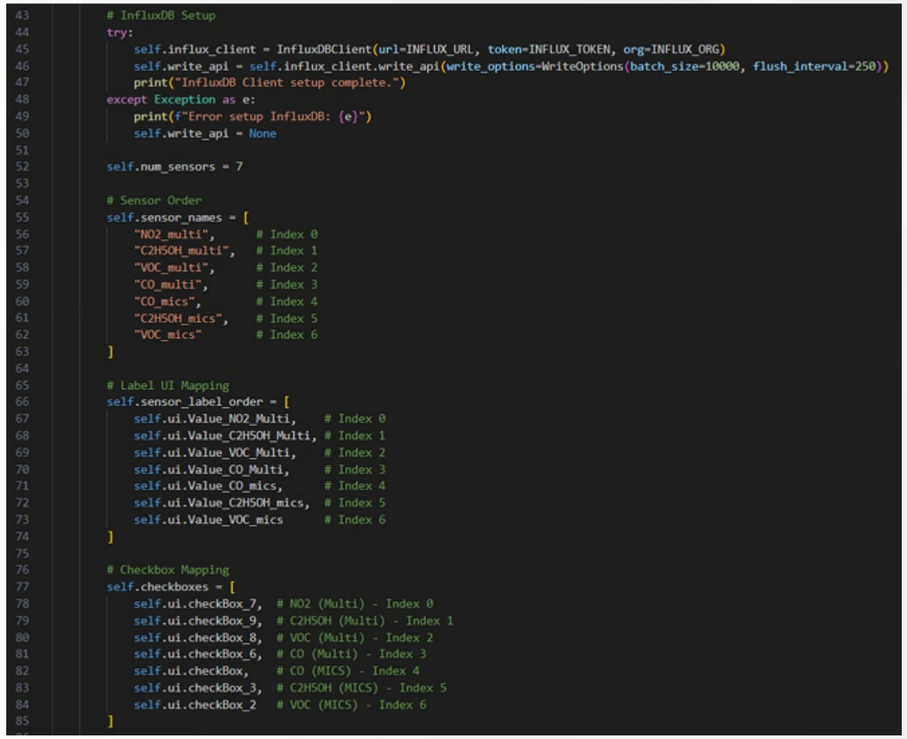
The electronic nose mimics the human sense of smell using an array of gas sensors, such as the **MICS-5524** and **Multichannel Gas Sensor v2**, which utilize Metal Oxide Semiconductor (MOS) technology. When exposed to volatile compounds from chocolate, the conductivity of these sensors changes, creating a time-series signal. To process this data efficiently, the project utilizes the Rust programming language for the backend due to its memory safety and high performance in handling serial communication. For the visualization layer, GNUPLOT and Qt Python are employed to render the complex multi-channel sensor data into understandable graphs for analysis.

SYSTEM ARCHITECTURE AND DATA FLOW



The system architecture is designed as a streamlined pipeline **starting with the hardware and ending with advanced data analysis.** The aroma from the chocolate sample is **circulated through a sensor array using a pump and fan**, where the **Arduino Uno R4 WiFi acquires the analog signals**. This data is **transmitted** to a host computer where a Rust-based backend parses the information and stores it in an InfluxDB time-series database. Simultaneously, **the data is served via an HTTP API to the Qt Python GUI application, allowing for real-time monitoring, while also being formatted for export to Edge Impulse for machine learning classification.**

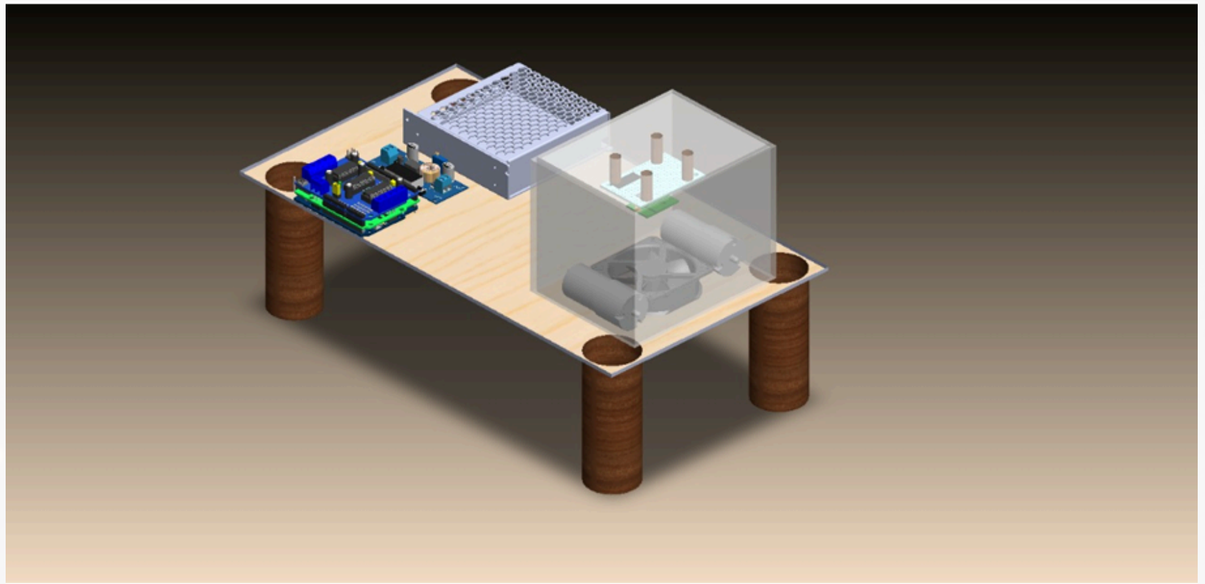
SOFTWARE
IMPLEMENTATION (FRONTEND & BACKEND)



```
# 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
]
```

The software implementation is divided into **two distinct** but integrated layers to maximize efficiency. The **backend, written in Rust, handles the heavy lifting of continuous serial communication with the Arduino, data parsing, and database management with InfluxDB**. It exposes a lightweight REST endpoint for data retrieval. On the user side, **the Qt Python GUI provides an interactive interface where users can input sample details, control the sampling process, and view live response curves.** This separation ensures that the data acquisition remains stable and fast, even while the user interacts with the graphical interface.

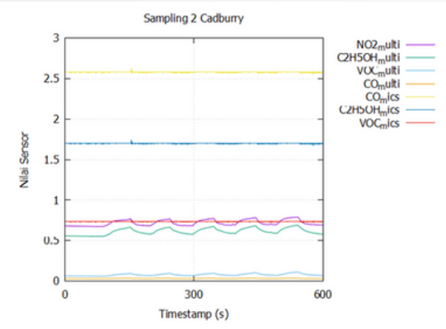
HARDWARE
INTEGRATION AND 3D DESIGN

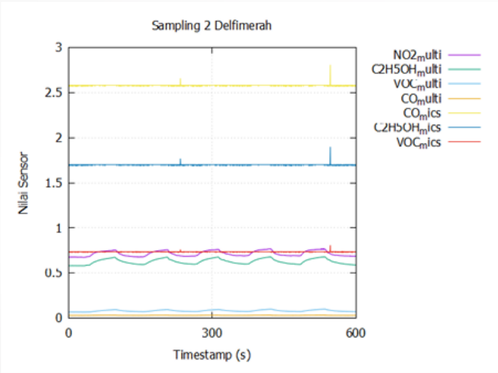


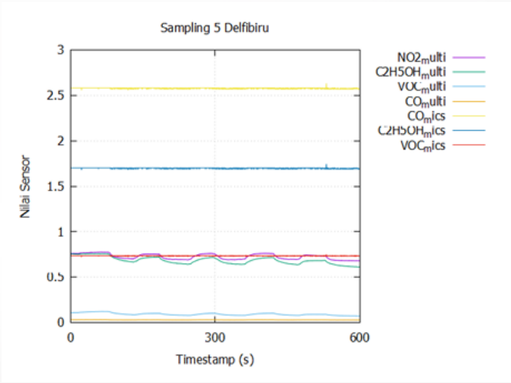
To ensure reliable sensor readings, the hardware is housed in a **custom-designed 3D-printed enclosure. This design optimizes the mechanical layout of the sample chamber, gas sensors, pump, and Arduino Uno R4 WiFi to ensure even airflow and minimize leakage.** The compact case includes specific mounting for electronic modules and cable routing to facilitate maintenance. Furthermore, **the system integrates with Edge Impulse, where the collected datasets (in CSV/JSON format) are uploaded to train machine learning models.** This integration allows the system not just to visualize data, but to eventually classify different chocolate types based on their aroma fingerprints.

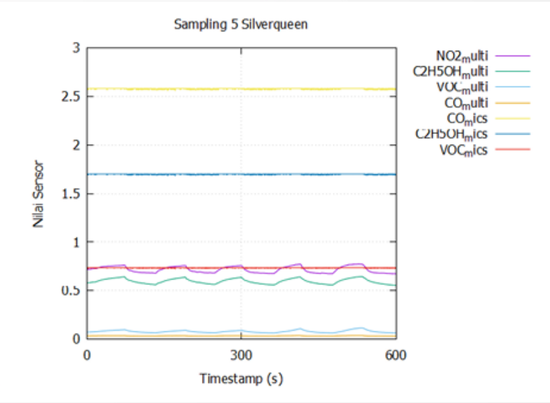
RESULTS
AND DISCUSSION

The system was tested using four different chocolate samples: **Cadbury**, **Delfi Red, Delfi Blue, and Silverqueen.** The resulting sensor response graphs revealed unique "fingerprints" for each sample. **Cadbury showed a stable pattern with moderate fluctuations, indicating a homogeneous composition. Delfi Blue exhibited pronounced peaks in certain channels, suggesting dominant aroma components**, whereas **Delfi Red appeared noisier**. **Silverqueen displayed the smallest amplitude and a flat curve**, indicating a lighter aroma intensity. These results demonstrate that the system can successfully differentiate between chocolate varieties based on their volatile compound profiles.









THANK YOU

KELOMPOK 4