EAS SPS eNose

By Kelompok 3

Aldi Rochmad Romdoni 2042241050
Adrianta Fajar Surapramana 2042241052
Hanif Raditya Pratama 2042241065

Penjelasan Proyek

electronic nose (eNose) berperan dalam mendeteksi dan mengklasifikasikan aroma serta komposisi gas dari berbagai sampel uji. Untuk memahami integrasi mendalam antara sistem akuisisi data, pemrosesan sinyal, dan visualisasi hasil secara real-time, proyek ini mengembangkan sebuah aplikasi desktop interaktif yang mampu menerima dan memproses data dari multiple sensor gas yang terhubung dengan Arduino Uno R4 WiFi.

Pada kelompok kami, sampel uji yang diambil adalah 4-5 jenis daun, yaitu daun kemangi, seledri, jeruk, kari, dan pandan

Penjelasan Proyek

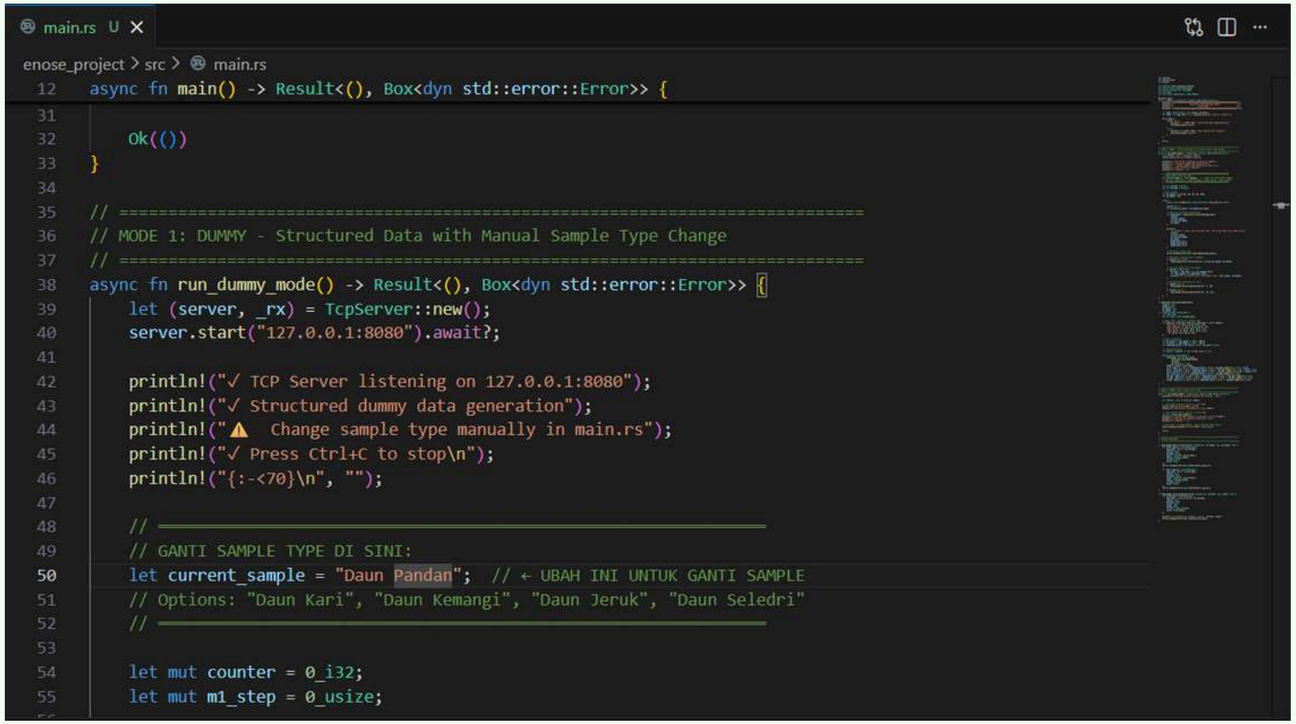
Aplikasi ini dirancang dengan arsitektur yang memisahkan tugas antara lapisan backend dan frontend untuk memastikan skalabilitas, maintainability, dan performa optimal. Stack teknologi yang digunakan mencakup:

- Rust sebagai backend untuk pemrosesan sinyal, manajemen data sensor, dan logika komputasi yang demand tinggi terhadap performa
- Qt Python (PySide6/PyQt6) sebagai framework GUI untuk menghadirkan antarmuka pengguna yang responsif dan intuitif
- Edge Impulse sebagai platform untuk pengembangan dan deployment model machine learning yang melakukan analisis dan klasifikasi data sensor secara real-time

Model Rust (backend)

main.rs

- Menentukan mode operasi: dummy atau normal (Arduino via Wi-Fi).
- Inisialisasi:
  - TcpServer
  - Receiver Arduino
  - Loop utama async untuk menangani aliran data.
- Menjembatani alur data antara Arduino → DataProcessor → Server → GUI.



Model Rust (backend)

server.rs

- Mengimplementasikan server TCP asinkron untuk koneksi ke GUI.
- Menangani beberapa klien sekaligus (multiclient).
- Melakukan broadcast pesan berformat Message:
  - SensorData
  - Status
  - Command
- Menyediakan channel internal untuk menerima data dari modul lain.

server.rs U

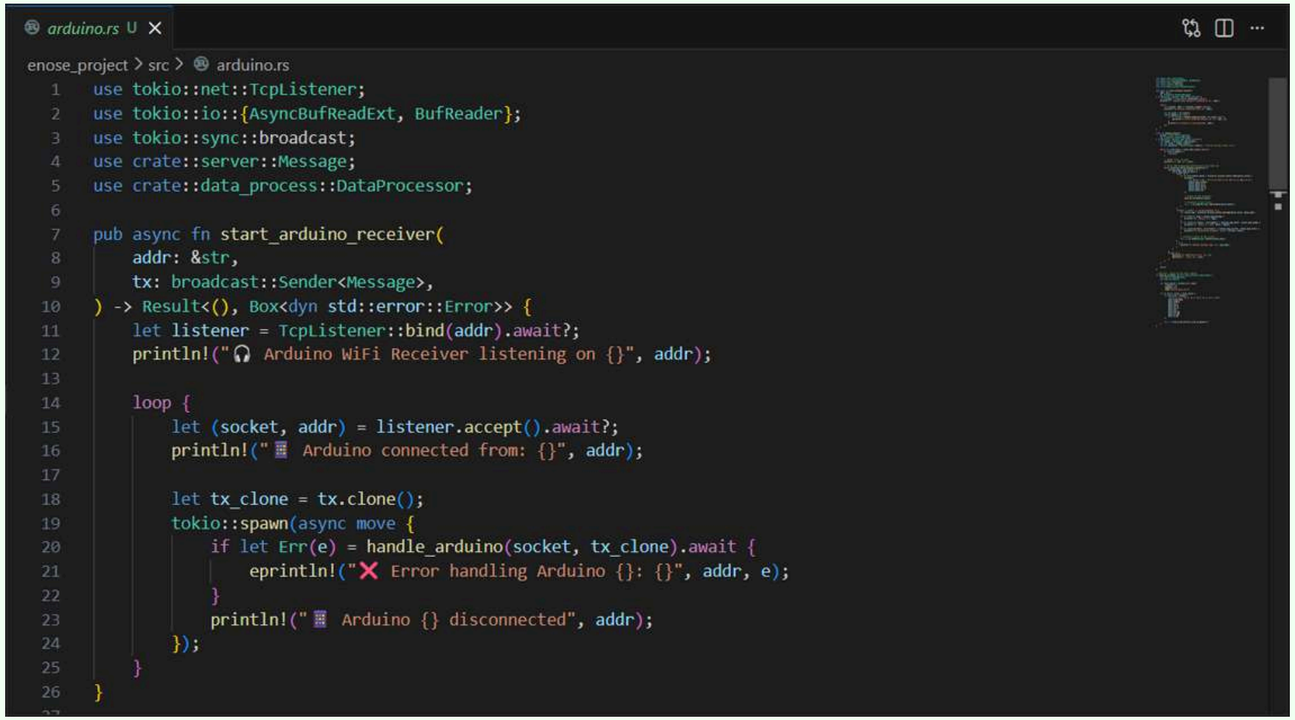
enose_project > src > server.rs

```rust
// src/server.rs
use crate::data_process::{SensorData, StatusMessage};
use serde_json;
use tokio::io::{AsyncBufReadExt, AsyncWriteExt, BufReader};
use tokio::net::{TcpListener, TcpStream};
use tokio::sync::broadcast;

#[derive(Clone, Debug)]
pub enum Message {
    SensorData(SensorData),
    Status(StatusMessage),
    Command(String),
}

pub struct TcpServer {
    tx: broadcast::Sender<Message>,
}

impl TcpServer {
    pub fn new() -> (Self, broadcast::Receiver<Message>) {
        let (tx, rx) = broadcast::channel(100);
        (Self { tx }, rx)
    }

    pub async fn start(&self, addr: &str) -> Result<(), Box<dyn std::error::Error>> {
        let listener = TcpListener::bind(addr).await?;
```

Model Rust (backend)

arduino.rs

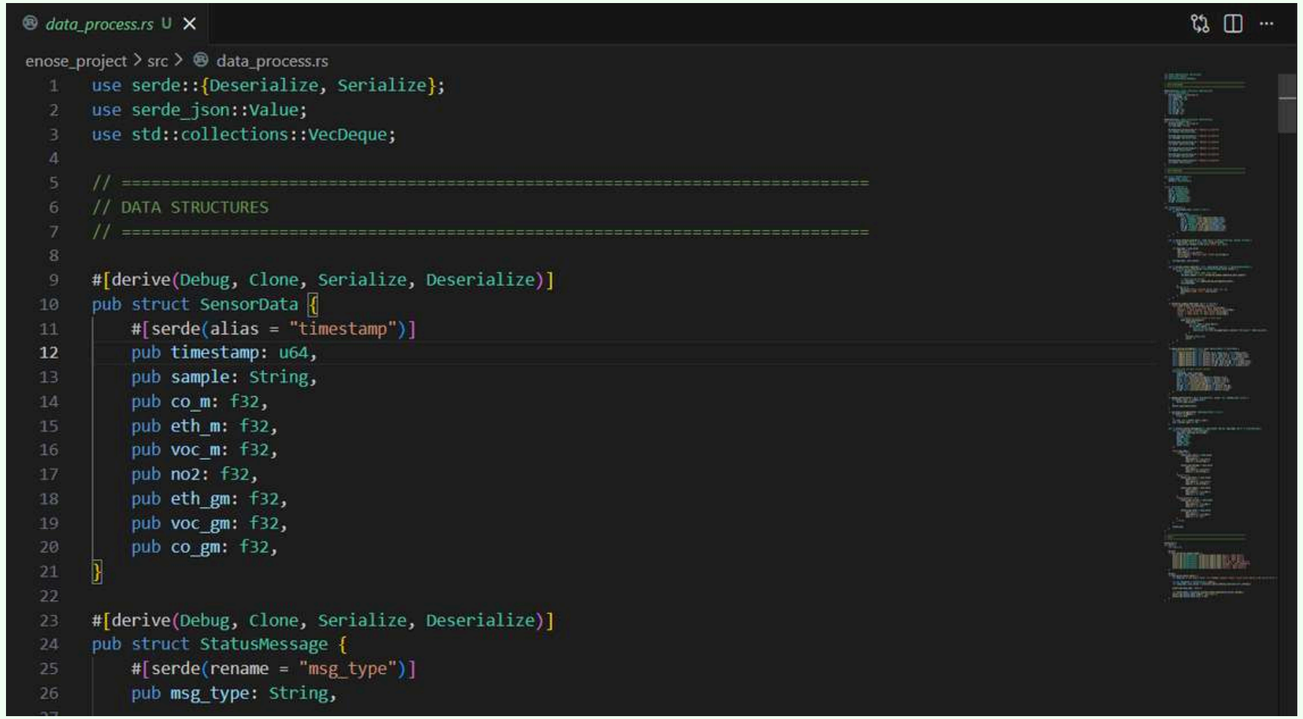
- Menerima data JSON dari Arduino melalui koneksi TCP/Wi-Fi.
- Memvalidasi dan meneruskan data mentah ke modul pemrosesan (DataProcessor).
- Mengirim hasil pemrosesan ke TcpServer untuk diteruskan ke GUI.
- Menangani reconnect, timeout, dan deteksi error data sederhana.



Model Rust (backend)

data_process.rs

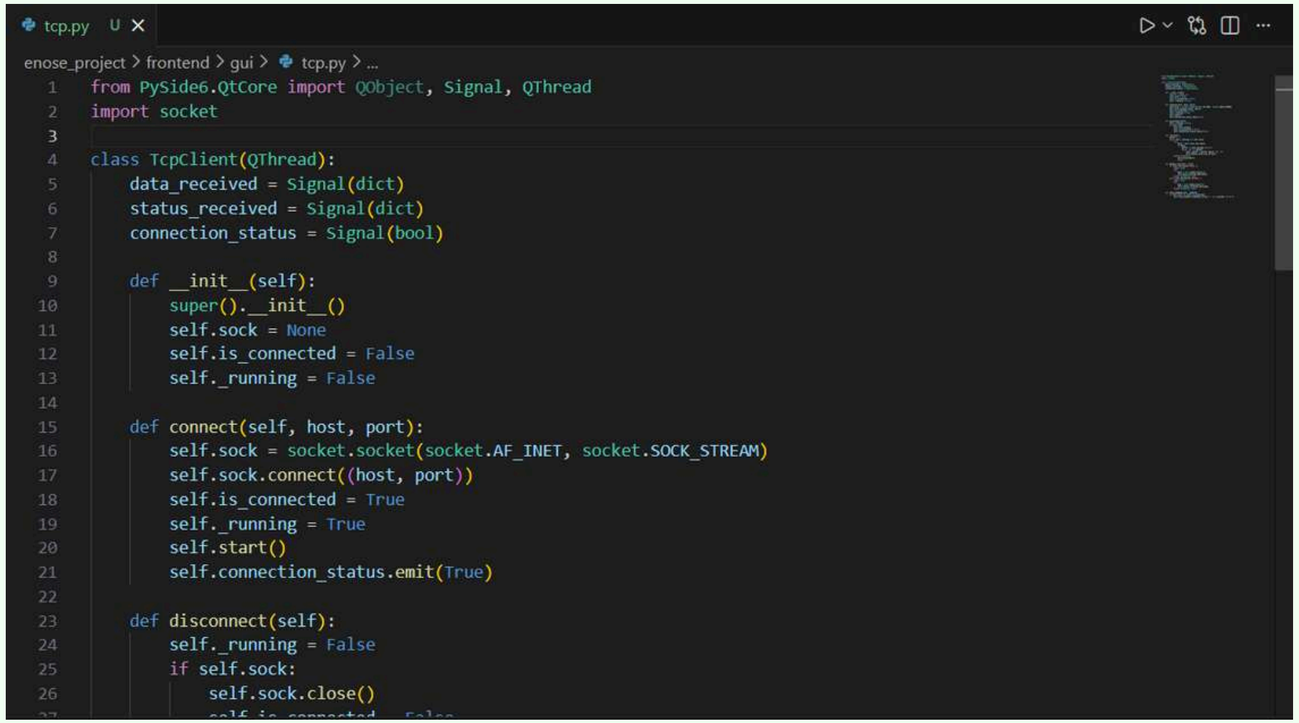
- Mendefinisikan struktur:
  - SensorData
  - StatusMessage
  - Tipe-tipe enum untuk pesan.
- Bertugas melakukan:
  - Parsing JSON
  - Normalisasi nama sampel
  - Smoothing (moving average)
  - Buffering data sensor untuk stabilitas tampilan di GUI.
- Output akhir dikemas dalam bentuk Message yang siap dibroadcast ke klien.



Model Python (frontend)

tcp.py — TcpClient

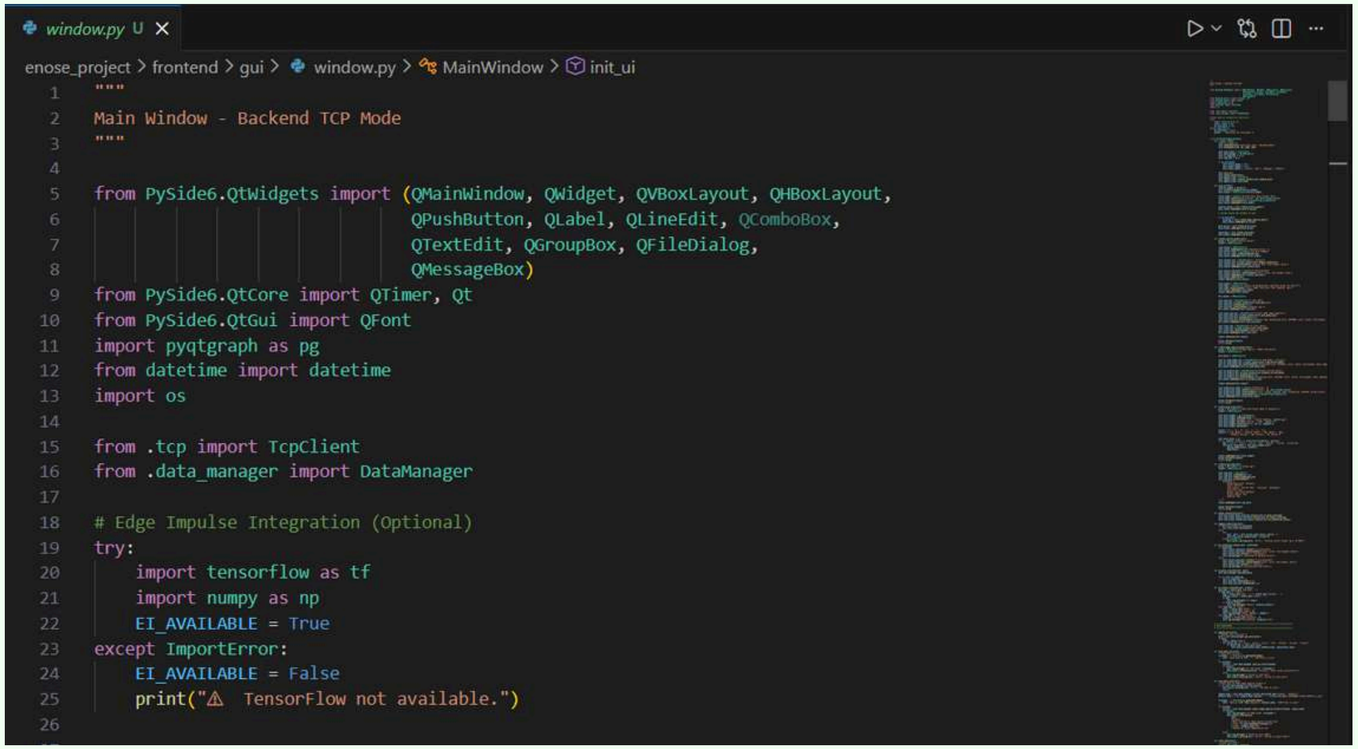
- Mengelola koneksi TCP ke backend (Rust).
- Menerima stream data mentah, memisahkan paket berdasarkan prefix:
  - DATA: → data sensor
  - STATUS: → status sistem / notifikasi
- Melakukan parsing dasar dan emit sinyal ke GUI (Qt/PySide).



Model Python (frontend)

window.py — MainWindow

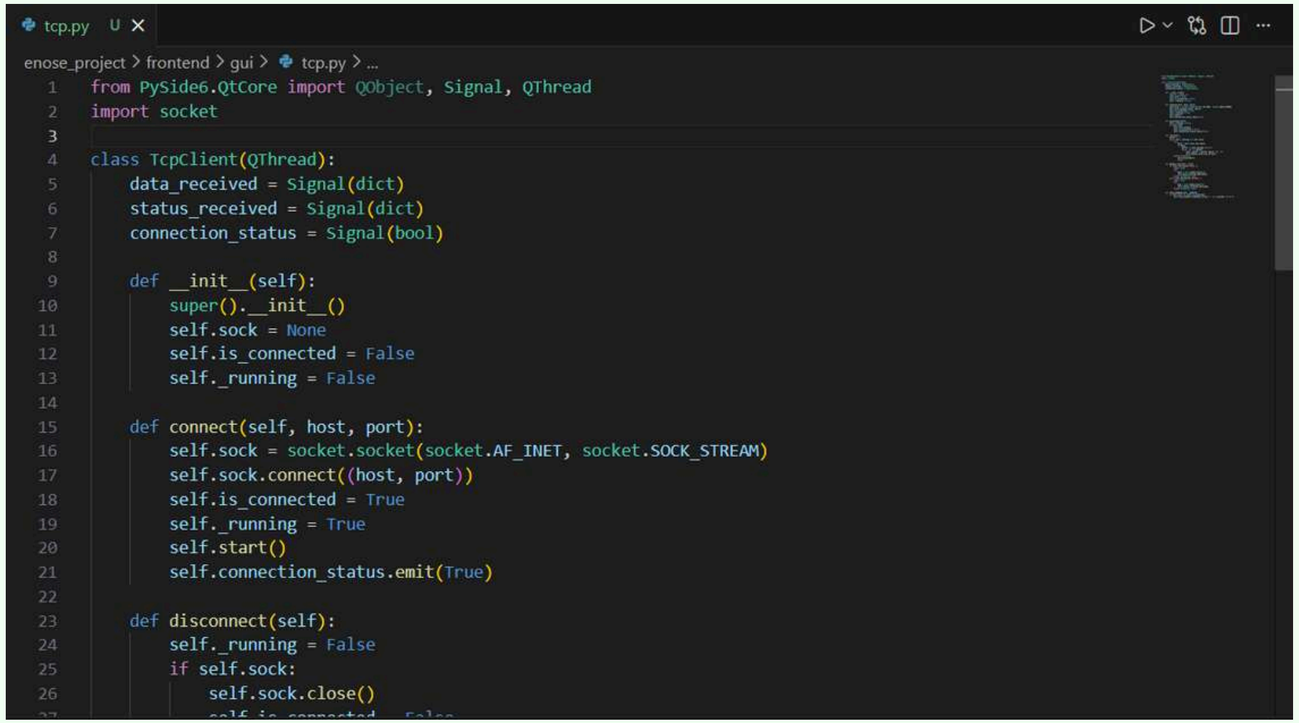
- Mengatur GUI utama:
  - kontrol koneksi
  - tombol simpan
  - panel log
  - indikator status
- Menghubungkan sinyal dari TcpClient → DataManager → plot real-time.
- Mendukung integrasi opsional panel model TFLite untuk inferensi lokal.



Model Python (frontend)

dtcp.py — TcpClient

- Mengelola koneksi TCP ke backend (Rust).
- Menerima stream data mentah, memisahkan paket berdasarkan prefix:
  - DATA: → data sensor
  - STATUS: → status sistem / notifikasi
- Melakukan parsing dasar dan emit sinyal ke GUI (Qt/PySide).



Sampel

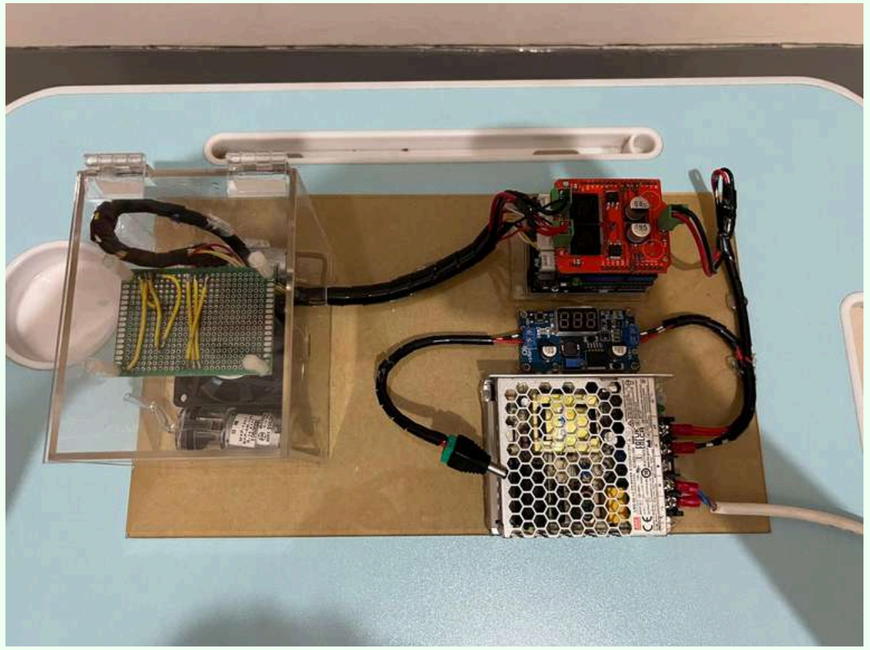
Daun Pandan

Daun Jeruk

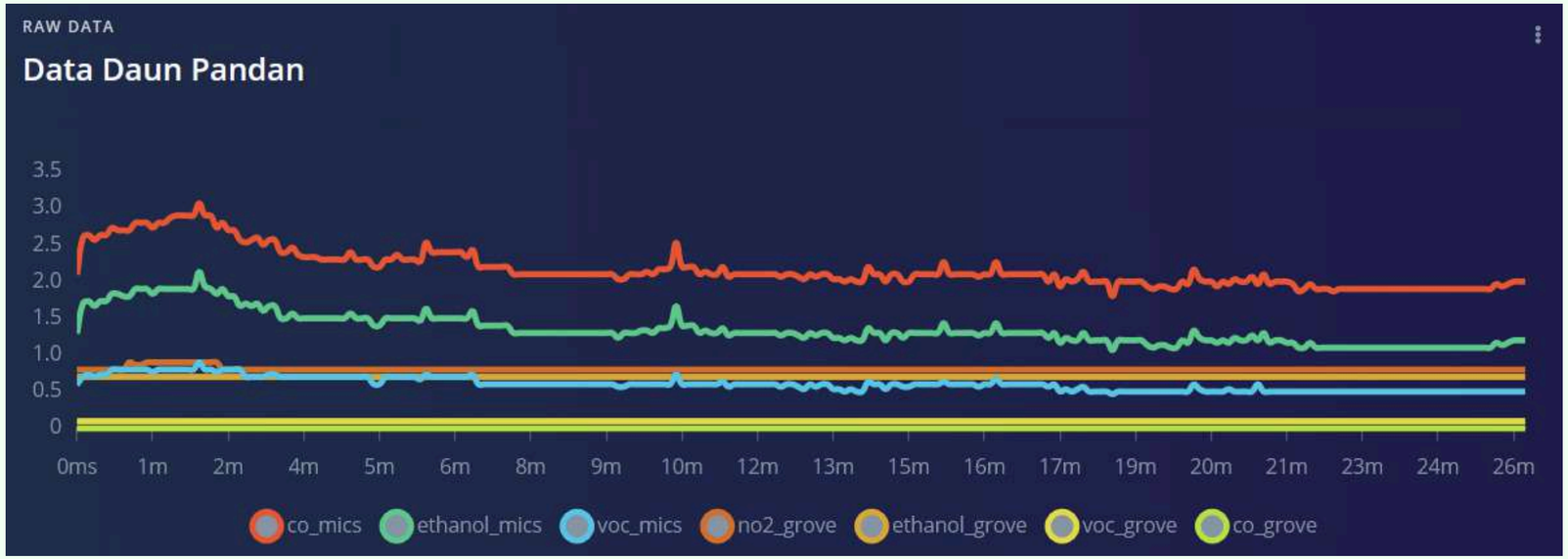
Daun Kari

Daun Kemangi

Dokumentasi Alat

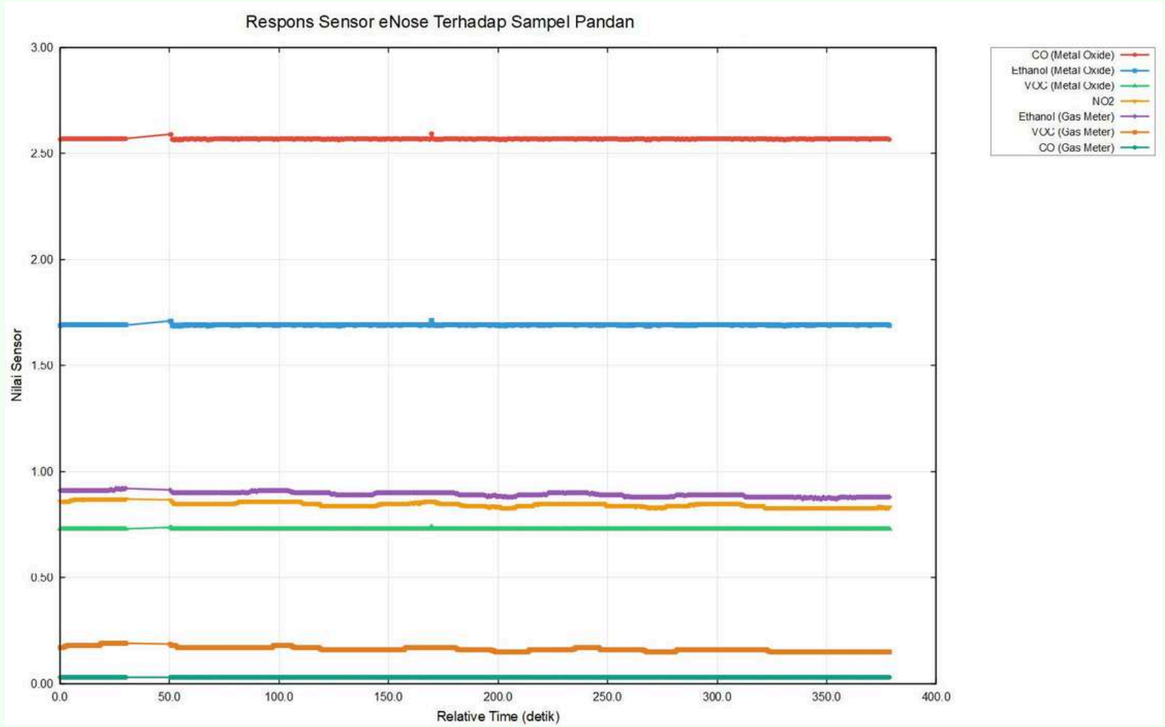


Hasil Data



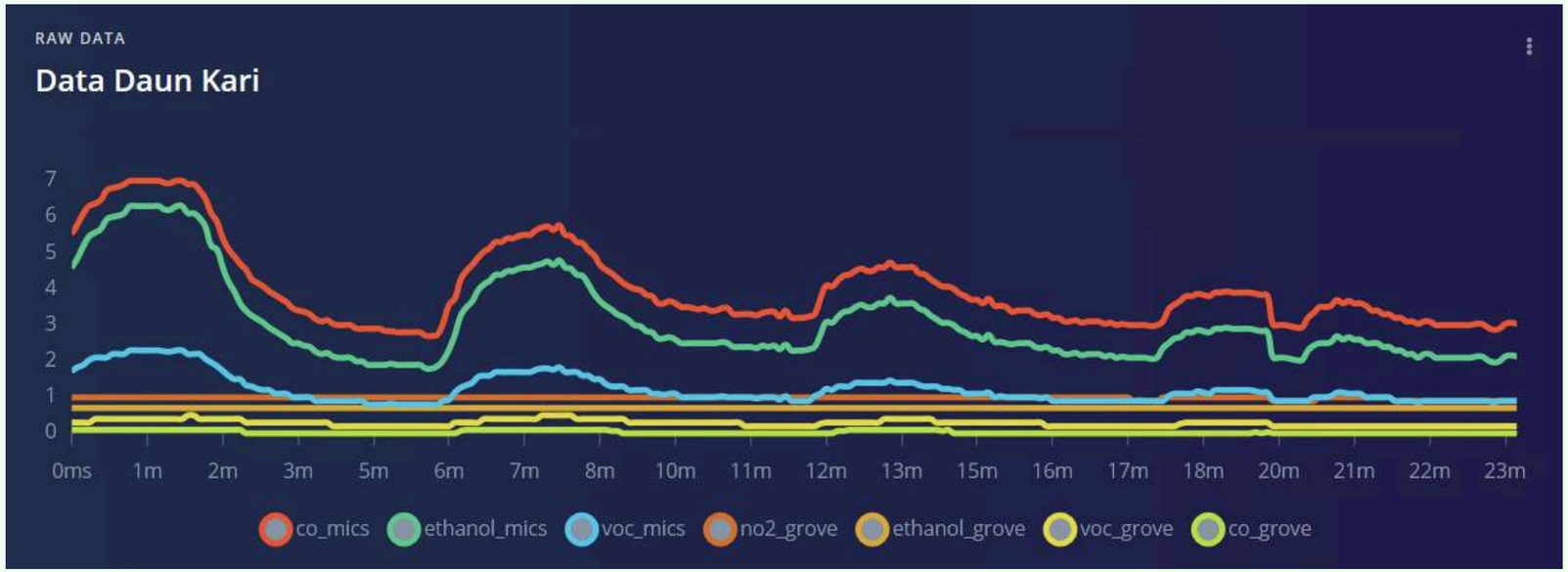
Daun Pandan Pompa 1

Hasil Data



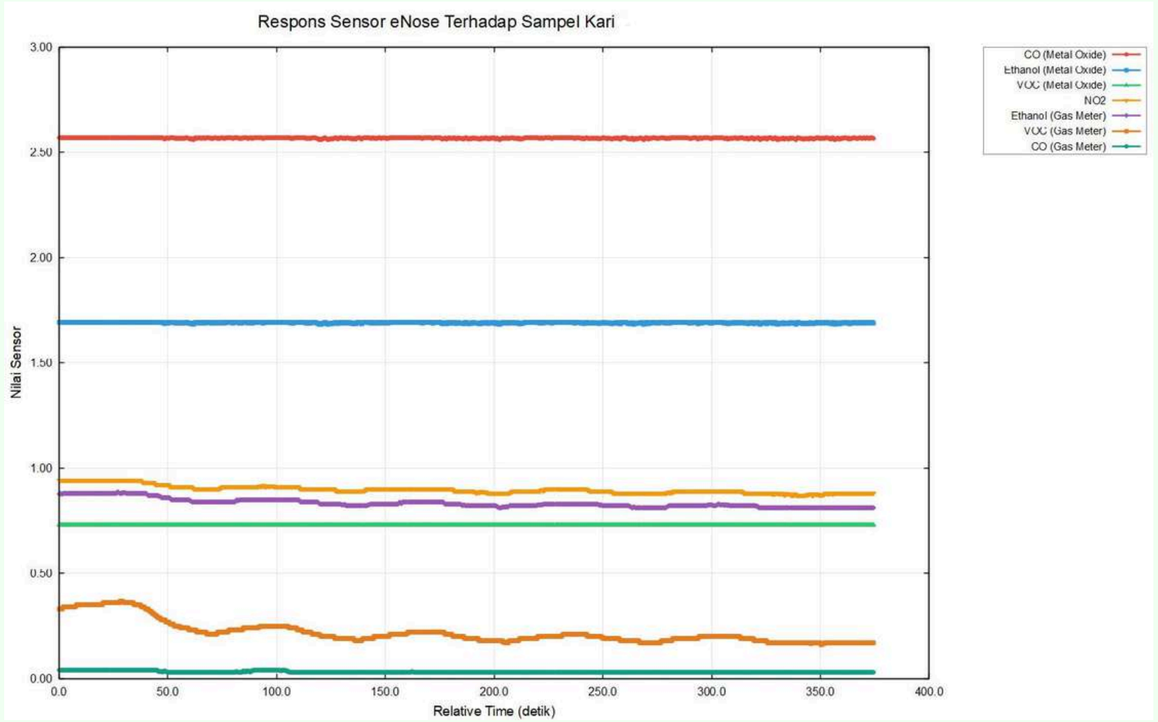
Daun Pandan Pompa 2

Hasil Data



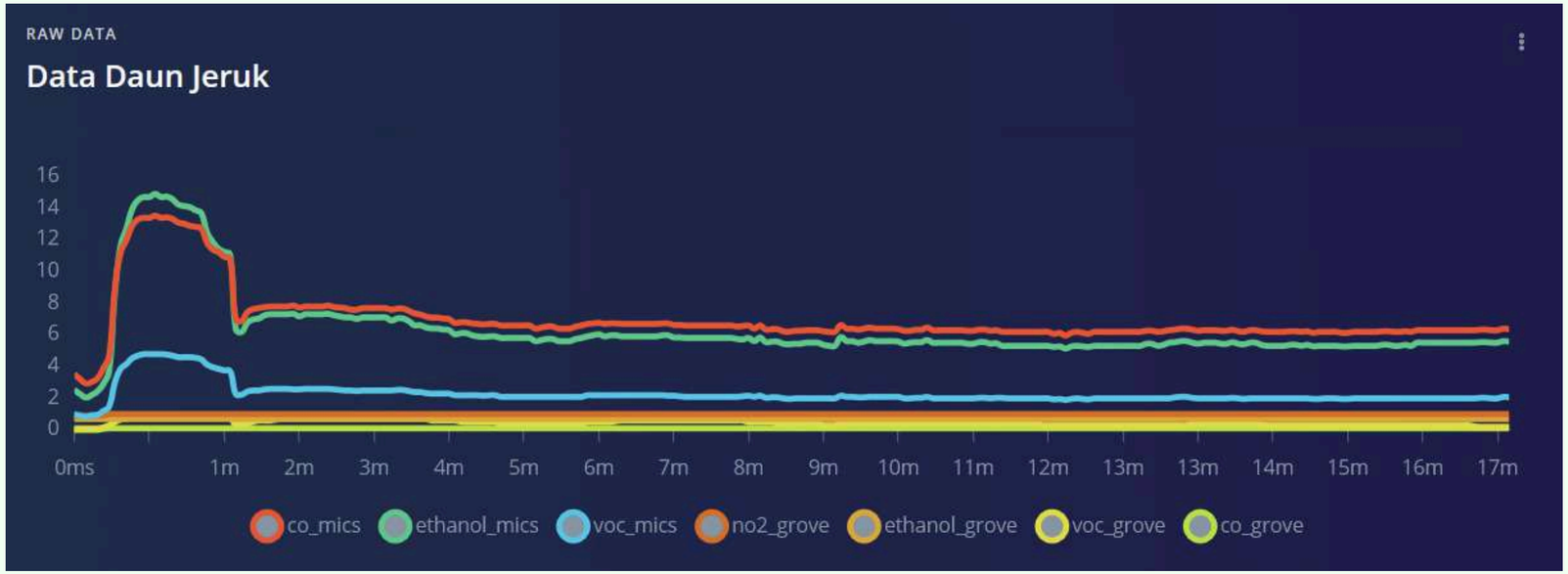
Daun Kari Pompa 1

Hasil Data



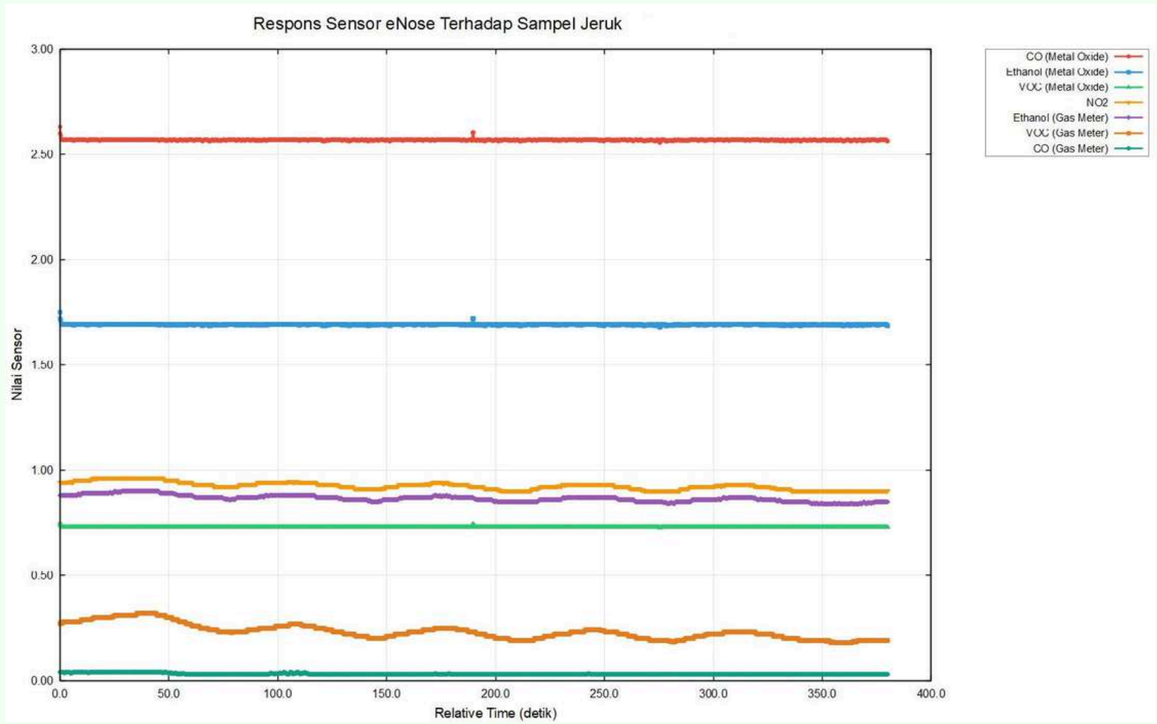
Daun Kari Pompa 2

Hasil Data



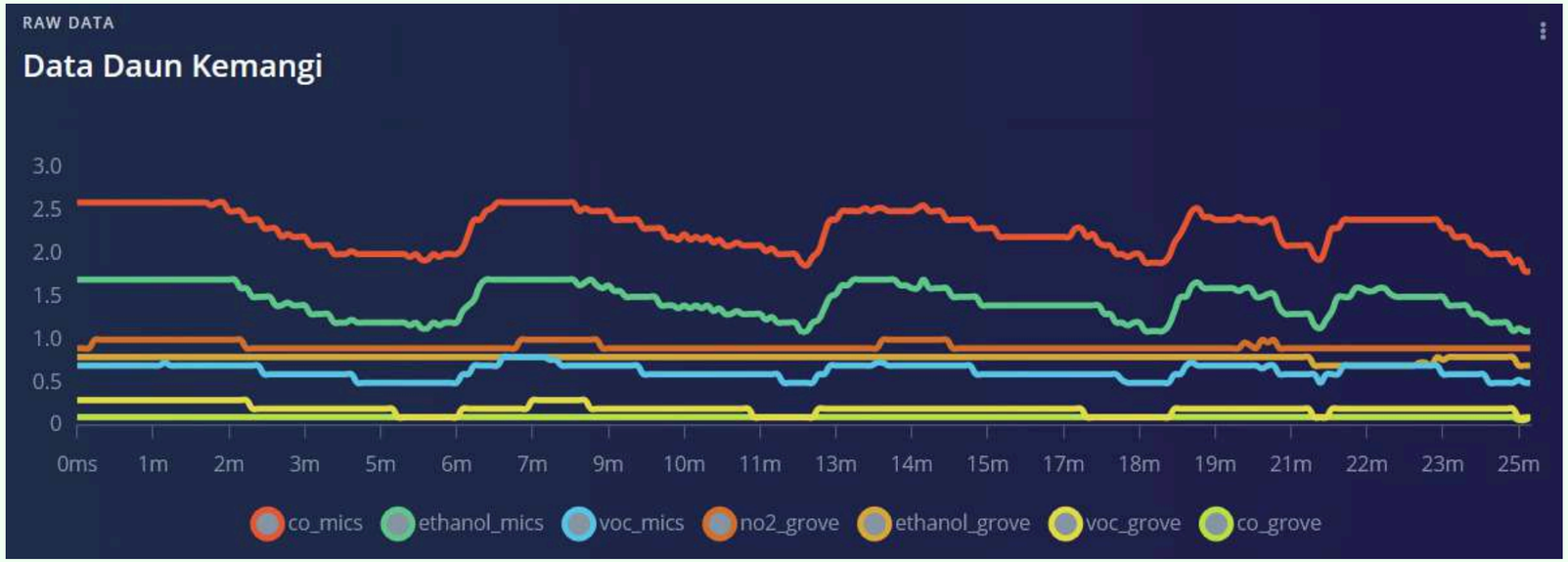
Daun Jeruk Pompa 1

Hasil Data



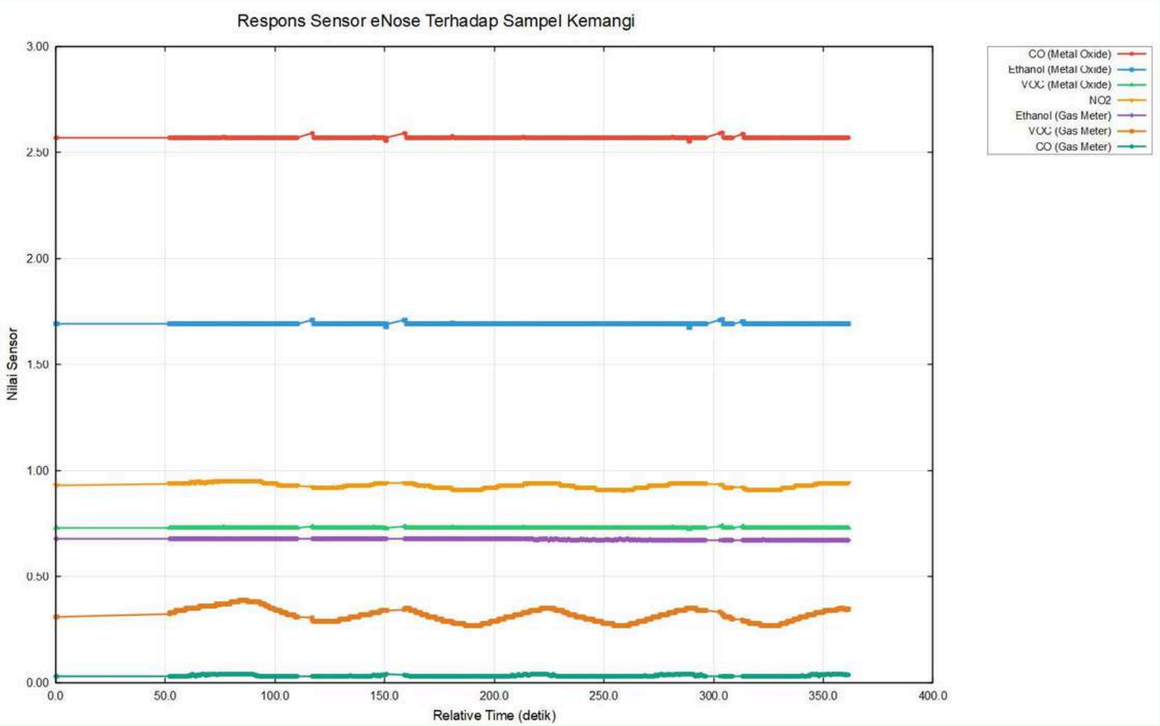
Daun Jeruk Pompa 2

Hasil Data



Daun Kemangi Pompa 1

Hasil Data



Daun Kemangi Pompa 2

Kesimpulan

Berdasarkan hasil percobaan, sistem eNose mampu mendeteksi gas yang dikeluarkan oleh keempat sampel daun. Namun, grafik keluaran menunjukkan bahwa penggunaan satu pompa dengan durasi pengukuran 30 menit menghasilkan pola sinusoidal yang lebih jelas dibandingkan konfigurasi dua pompa dengan durasi hanya 5 menit. Perbedaan kualitas grafik ini dapat disebabkan oleh beberapa faktor, antara lain perbedaan jumlah pompa, variasi temperatur lingkungan, tingkat kekedapan chamber uji, maupun karakteristik gas dari masing-masing sampel daun.