LAPORAN AKHIR SISTEM PENGOLAHAN SINYAL

"Sistem Electronic Nose untuk Analisis Aroma Herbal"

Dosen: Ahmad Radhy, S.Si., M.Si.

Oleh:

Galen Zahid Wajendra – 2042241044

Rijal Difaul Haq – 2042241097

Sintia Ompusunggu – 2042241113

PRODI D4 TEKNOLOGI REKAYASA INSTRUMENTASI

DEPARTEMEN TEKNIK INSTRUMENTASI

FAKULTAS VOKASI

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

2025

Daftar Isi

Bab 1

PENDAHULUAN

1.1 Latar Belakang

Kebutuhan terhadap teknologi pemantauan aroma berbasis sensor meningkat pesat seiring berkembangnya standar keamanan pangan serta meningkatnya permintaan konsumen terhadap bahan pangan yang terjamin kualitasnya. Berbagai komoditas termasuk rempah-rempah seperti jahe, kencur, kunyit, dan lengkuas memiliki profil volatil yang kaya dan kompleks, sehingga teknik monitoring konvensional sering kali tidak memadai untuk menangkap dinamika senyawa aromatiknya. Metode laboratorium seperti GC–MS dan HPLC memang memiliki kemampuan analitik tinggi, namun memerlukan biaya besar, waktu lama, serta tidak dapat digunakan untuk pemantauan real-time. Kondisi ini mendorong berkembangnya sistem *electronic nose* (eNose) sebagai perangkat bioinspiratif yang meniru mekanisme penciuman manusia menggunakan sensor gas untuk mendeteksi pola *volatile organic compounds* (VOCs), seperti dijelaskan pada studi.

Kajian literatur menunjukkan bahwa eNose telah digunakan secara luas untuk penilaian mutu pangan, deteksi kontaminasi, dan klasifikasi bahan makanan, dengan dukungan algoritma analitik seperti PCA, LDA, dan SVM. Pada komoditas rempah, eNose terbukti mampu membedakan profil aroma rempah yang mirip, sebagaimana ditunjukkan dalam penelitian mengenai klasifikasi rempah menggunakan machine learning.

Selain itu, penelitian menunjukkan bagaimana proses pengolahan bahan alami mengubah profil volatilnya, dan bagaimana eNose mampu menangkap perubahan tersebut secara signifikan. Arah perkembangan terbaru seperti SMELLNET menegaskan bahwa pemrosesan sinyal VOC memerlukan pipeli-

ne digital berstruktur, mulai dari preprocessing temporal hingga pemodelan berbasis Transformer.

Di sisi implementasi, masih banyak sistem eNose akademik yang belum memiliki visualisasi sinyal yang intuitif, alur data yang rapi, dan kemampuan pemrosesan real-time yang stabil. Penelitian ini berupaya menjawab kebutuhan tersebut dengan mengembangkan aplikasi GUI berbasis Qt Python yang terhubung ke backend Rust untuk menangani komunikasi serial dan pemrosesan sinyal secara efisien. Empat jenis rempah (jahe, kencur, kunyit, lengkuas) digunakan sebagai sampel karena memiliki profil volatil kompleks dan relevan untuk studi klasifikasi aroma. Dengan demikian, penelitian ini diharapkan dapat menyediakan platform visualisasi data eNose yang stabil, informatif, dan siap dikembangkan menuju analisis machine learning di masa depan.

1.2 Rumusan Masalah

1. Bagaimana membangun sistem e-Nose lengkap dengan kontrol otomatis?
2. Bagaimana mengintegrasikan Arduino, backend Rust, dan frontend Python?
3. Bagaimana memvisualisasi data sensor secara real-time?

1.3 Tujuan Project

1. Membangun sistem e-Nose untuk analisis 4 herbal (jahe, kencur, kunyit, lengkuas)
2. Mengimplementasikan FSM otomatis dengan kontrol motor
3. Membuat GUI real-time untuk visualisasi data

1.4 Manfaat Project

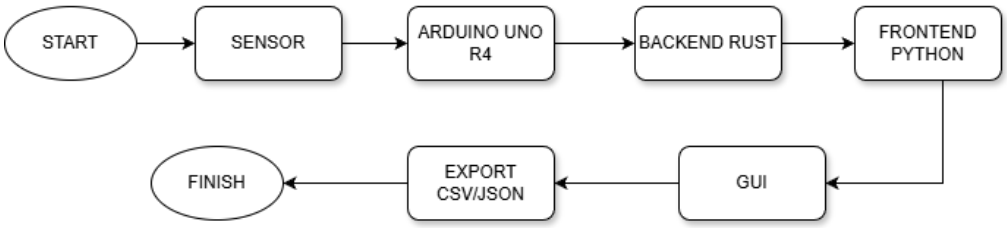
1. Sebagai sistem otomatis untuk penelitian herbal
2. Sebagai Platform untuk dataset aroma herbal
3. Sebagai Dasar untuk pengembangan machine learning

Bab 2

TINJAUAN PUSTAKA

2.1 Perancangan dan Struktur Sistem

Backend Rust, frontend Python, dan sistem Arduino bekerja sebagai satu ekosistem terpadu untuk mengoperasikan eNose secara real-time. Backend Rust dirancang dengan arsitektur server TCP dua-port, yaitu 8081 untuk menerima data sensor dari Arduino dan 8082 untuk mengirimkan data ke GUI, dengan mekanisme broadcast channel sehingga banyak klien dapat menerima data secara bersamaan. Setiap paket yang diterima dalam format string seperti SENSOR:value1,value2,... akan diparsing, diserialisasi menjadi JSON, lalu didistribusikan ke semua frontend yang terhubung sambil tetap memantau status koneksi. Frontend berbasis Python (Qt GUI) memiliki arsitektur komunikasi yang sama, menerima data JSON dari Rust untuk divisualisasikan dalam grafik real-time, indikator numerik, dan penyimpanan dataset. Adapun Arduino berperan sebagai pengendali utama perangkat fisik, menjalankan lima level stimulus kecepatan kipas dan membaca dua jenis sensor gas, serta menjalankan fitur kontrol khusus seperti kipas yang menyala lima detik lebih awal pada fase PURGE dan pompa dengan kontrol dua arah untuk pembersihan chamber. Kombinasi ketiga modul ini menghasilkan sistem eNose yang stabil, terstruktur, dan mampu menangani aliran data VOC secara konsisten dari akuisisi hingga visualisasi.



Gambar 2.1: Diagram blok sistem eNose analisis aroma herbal

Diagram blok sistem menunjukkan alur mulai dari sensor GM-XXX dan MiCS-5524 yang membaca data gas, kemudian Arduino Uno R4 WiFi mengakuisisi dan mengirimkan data tersebut melalui TCP ke Backend Rust. Backend bertugas menerima, memproses, dan menyediakan data ke Frontend Python. GUI Python (PyQt5 + PyQtGraph) menampilkan grafik dan informasi real-time. Backend juga memungkinkan ekspor data dalam format CSV/JSON. Setelah proses pengambilan dan visualisasi data selesai, sistem berakhir pada tahap finish.

2.1.1 Electronic Nose (eNose)

Electronic Nose atau eNose merupakan perangkat berbasis sensor gas yang berfungsi meniru sistem penciuman manusia. Perangkat ini menggunakan array sensor semikonduktor (MOS) atau sensor elektrokimia untuk mendeteksi *volatile organic compounds* (VOCs). Setiap sensor memberikan respons resistansi berbeda terhadap perubahan konsentrasi gas, menghasilkan pola yang dapat dianalisis secara komputasional untuk mengenali aroma atau kualitas bahan pangan.

2.1.2 Pengolahan Sinyal VOC

Pengolahan sinyal pada eNose umumnya meliputi proses filtrasi noise, normalisasi, baseline correction, ekstraksi fitur, dan analisis pola. Teknik seperti PCA, LDA, SVM, dan neural network banyak digunakan dalam klasifikasi aroma rempah serta penilaian kualitas produk pangan.

2.1.3 Sistem Akuisisi Data Real-Time

Data VOC perlu diakuisisi secara real-time untuk mengamati perubahan respons sensor sepanjang sesi pengujian. Sistem akuisisi umumnya menggunakan

ADC, komunikasi serial, serta protokol TCP/UDP agar data dapat dikirimkan ke perangkat pemroses utama secara kontinu.

2.2 Penjelasan Komponen

2.2.1 Backend (Rust)

Rust dipilih sebagai backend karena memiliki *memory safety*, kecepatan mendekati C++, dan stabil untuk koneksi TCP jangka panjang. Rust mengelola parsing data, penyimpanan, serta integrasi ke GUI secara efisien tanpa risiko *race condition*.

2.2.2 Frontend (Python, PyQt5 + PyQtGraph)

Qt Python menyediakan antarmuka grafis modern untuk plotting sinyal, tampilan numerik, input form, hingga penyimpanan dataset. Framework ini mendukung real-time graphing melalui PyQtGraph dan integrasi event-based yang cepat serta interaktif.

2.2.3 Arduino Uno R4 WiFi sebagai Platform Akuisisi Data

Arduino Uno R4 WiFi berfungsi sebagai pengontrol utama sensor eNose. Dengan mikrokontroler Renesas RA4M1 (Cortex-M4), perangkat ini mampu melakukan pembacaan sensor berkecepatan tinggi. Fitur WiFi berbasis ESP32-S3 memungkinkan pengiriman data langsung ke backend Rust.

Fitur relevan untuk sistem eNose:

- **ADC presisi tinggi** digunakan untuk membaca sensor MiCS-5524.
- **I2C bus** digunakan untuk membaca sensor GM-XXX.
- **PWM dan kontrol arah motor** untuk kipas dan pompa.
- **WiFi TCP client** agar data dapat dikirimkan secara real-time.

Sistem Embedded eNose Berbasis Arduino Uno R4 WiFi

Kode Arduino yang digunakan pada sistem eNose dirancang untuk mengelola sensor, aktuator, dan komunikasi secara terintegrasi. Berikut adalah komponen-komponen utama berdasarkan analisis kode:

Manajemen Koneksi WiFi

Modul WiFiS3 digunakan untuk menjaga koneksi TCP persisten ke backend Rust. Fungsi `ensureConnected()` memastikan Arduino selalu berusaha terhubung kembali jika koneksi terputus.

Pembacaan Sensor GM-XXX dan MiCS-5524

Sensor GM-XXX dibaca melalui I2C menggunakan pustaka `Multichannel_Gas_GMXXX`. Sementara itu, MiCS-5524 dibaca melalui ADC analog, dihitung resistansinya (Rs), lalu dikonversi menjadi estimasi ppm melalui fungsi `ppmFromRatio()` berdasarkan kurva sensitivitas gas.

Pengendalian Motor Kipas dan Pompa

Motor dikontrol dengan PWM dan dua pin arah (H-bridge). Kipas dan pompa dapat diarahkan untuk:

- menghisap sampel (normal),
- membuang udara (reverse) untuk proses *purge*.

Pada fase PURGE, kipas menyala 5 detik lebih dahulu sebelum pompa, sesuai modifikasi logika pembersihan chamber.

Finite State Machine (FSM)

FSM mengatur seluruh tahapan sampling:

1. **PRE_COND** persiapan awal sensor.
2. **RAMP_UP** peningkatan kecepatan kipas bertahap.
3. **HOLD** pengambilan data VOC selama durasi tertentu.
4. **PURGE** pembersihan chamber (kipas lead + pompa buang).

5. **RECOVERY** pendinginan dan stabilisasi sistem.

Setiap tahap memiliki durasi terdefinisi, misalnya:

- HOLD 2030 detik,
- PURGE 40 detik dengan kipas lead 5 detik.

Pengiriman Data Sensor ke Backend

Data dikirim setiap 250ms menggunakan format string:

```
SENSOR:NO2,ETH,VOC,CO,CO_mics,ETH_mics,VOC_mics,state,level
```

Backend Rust membaca string ini, memprosesnya, lalu meneruskan ke GUI untuk divisualisasikan dalam bentuk grafik dan indikator.

Integrasi dengan GUI Qt Python

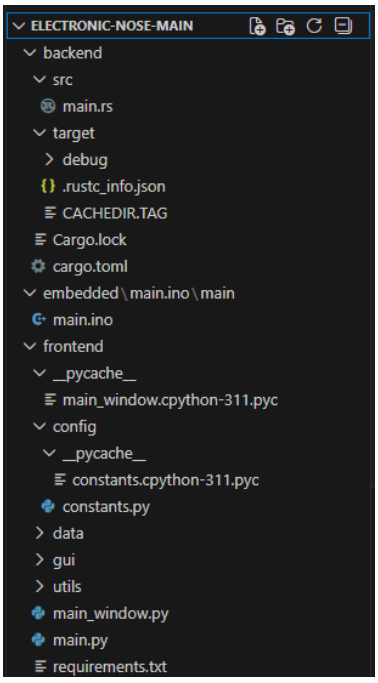
GUI menerima data TCP, memplotnya secara real-time, dan menyimpan dataset dalam format CSV/JSON berdasarkan hasil sampling kelima level rempah (jahe, kencur, kunyit, lengkuas).

Bab 3

METODE PENELITIAN

3.1 Desain Sistem eNose

Sistem eNose terdiri dari tiga komponen utama: perangkat akuisisi sensor berbasis Arduino Uno R4 WiFi, backend Rust untuk pemrosesan data dan komunikasi TCP, serta frontend Qt Python untuk visualisasi grafis. Data VOC dibaca oleh sensor GM-XXX dan MiCS-5524, kemudian dikemas dan dikirimkan melalui koneksi TCP ke backend Rust. Sistem bekerja secara real-time dan mendukung lima level pengujian aroma rempah.



Gambar 3.1: Susunan struktur project eNose

Struktur folder pada proyek *Electronic Nose* terdiri dari tiga komponen

utama, yaitu **embedded**, **backend**, dan **frontend**. Bagian *embedded* berisi file `main.ino` yang digunakan pada Arduino untuk melakukan pembacaan sensor. Folder *backend* memuat kode Rust pada file `main.rs` beserta berkas Cargo yang berfungsi sebagai server TCP untuk menerima dan memproses data dari Arduino. Selanjutnya, folder *frontend* berisi kode Python seperti `main_window.py` serta modul *gui*, *data*, dan *utils* yang digunakan untuk menampilkan antarmuka pengguna dan grafik real-time. Secara keseluruhan, struktur ini menunjukkan pemisahan fungsi yang jelas antara sisi perangkat keras, pengolahan data di server, dan tampilan visual pada GUI.

3.2 Implementasi Backend Rust

```rust
async fn process_sensor_data(line: &str, tx: &broadcast::Sender<String>) {
    let content = line.trim_start_matches("SENSOR:");
    let parts: Vec<&str> = content.split(',').collect();

    if parts.len() >= 9 {
        let data = SensorData {
            timestamp: Utc::now(),
            no2: parts[0].parse().unwrap_or(-1.0),
            eth: parts[1].parse().unwrap_or(-1.0),
            // ... parsing 7 sensor values
            state: parts[7].parse().unwrap_or(0),
            level: parts[8].parse().unwrap_or(0),
        };

        // Kirim data sensor ke Frontend
        if let Ok(json_str) = serde_json::to_string(&data) {
            let _ = tx.send(json_str);
        }
    }
}
```

Gambar 3.2: Potongan kode backend

Fungsi `process_sensor_data()` bertugas melakukan parsing dan transformasi data dari format string yang dikirim Arduino menuju struktur data Rust yang lebih terorganisasi. Data mentah dalam bentuk `"SENSOR:value1,value2,..."` dipisahkan menggunakan delimiter koma dan dikonversi ke tipe `f64` dengan penanganan error yang aman menggunakan `unwrap_or()`. Setiap paket data juga dilengkapi dengan *timestamp* menggunakan `Utc::now()` untuk menandai waktu penerimaan di sisi server.

Konversi ke JSON menggunakan library `serde` memberikan kemampuan interoperabilitas dengan berbagai frontend. Struktur `SensorData` yang didefinisikan dengan atribut `#[derive(Serialize)]` memungkinkan proses serialisasi otomatis dengan field yang konsisten. Dengan demikian, backend tidak hanya berfungsi sebagai penerus data, tetapi juga memperkaya paket dengan metadata, memvalidasi format, dan memastikan integritas sebelum data dikirimkan ke frontend Python.

```python
class NetworkWorker(QThread):
    data_received = Signal(dict)
    connection_status = Signal(bool)
    arduino_status = Signal(bool)

    def run(self):
        while self.running:
            try:
                self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                self.socket.connect((self.host, self.port))
                self.connection_status.emit(True)
                self._listen_for_data()
            except Exception as e:
                self.error_occurred.emit(str(e))

    def _listen_for_data(self):
        while self.running:
            data = self.socket.recv(4096).decode('utf-8')
            if data:
                self._process_received_data(data)

    def send_command(self, command: str):
        if self.socket:
            self.socket.sendall(f"{command}\n".encode('utf-8'))
```

Gambar 3.3: Potongan kode backend

`NetworkWorker` berperan sebagai *QThread* yang menangani komunikasi jaringan dengan backend Rust secara asinkron. Kelas ini memanfaatkan tiga sinyal utama, yaitu `data_received` untuk meneruskan data sensor ke GUI, `connection_status` untuk memperbarui status koneksi backend, serta `arduino_status` untuk memantau keterhubungan perangkat Arduino. Dengan pendekatan berbasis thread ini, proses jaringan berjalan di *background* sehingga antarmuka aplikasi tetap responsif selama streaming data berlangsung.

Mekanisme *reconnection* otomatis dengan pola *exponential backoff* melalui variabel `reconnect_attempts` membantu sistem memulihkan koneksi ketika terjadi gangguan jaringan. Pengelolaan buffer menggunakan `buffer.split('\n')`

dirancang untuk mengatasi fragmentasi paket yang umum terjadi pada komunikasi TCP, sehingga setiap baris data dapat diproses secara konsisten. Selain itu, fungsi `send_command()` memungkinkan frontend mengirimkan instruksi kontrol ke Arduino melalui backend Rust menggunakan format `command + "\n"`, sesuai dengan protokol yang diimplementasikan pada firmware Arduino.

3.3 Implementasi GUI Qt Python

```python
def on_data_received(self, data: dict):
    """Handle data received from Backend"""
    try:
        # Extract sensor values from JSON data
        sensor_values = [
            float(data.get('no2', 0.0)),
            float(data.get('eth', 0.0)),
            # ... 7 sensor values
        ]

        state_idx = int(data.get('state', 0))
        state_name = STATE_NAMES.get(state_idx, "UNKNOWN")
        level = data.get('level', 0)

        # Update system status
        progress = int((state_idx / 6) * 100)
        self.update_system_status(state_name, progress)

        if self.is_sampling:
            self.process_new_data(sensor_values)

    except Exception as e:
        print(f"Error parsing data: {e}")

def process_new_data(self, sensor_values: list):
    """Process new sensor data"""
    self.plot_widget.add_data_point(self.start_time, sensor_values)
```

Gambar 3.4: Potongan kode frontend

Callback `on_data_received()` berfungsi sebagai pusat pemrosesan data pada sisi frontend dan dieksekusi setiap kali data baru diterima dari backend. Fungsi ini mengekstrak tujuh nilai sensor dari paket JSON, menerjemahkan indeks state menjadi nama state yang lebih mudah dibaca, serta memperbarui *progress bar* berdasarkan kondisi *state machine*. Seluruh proses berlangsung secara real-time tanpa menghambat *GUI thread*, sehingga visualisasi tetap responsif selama proses sampling berlangsung.

Fungsi `process_new_data()` memiliki tiga tugas utama: (1) menambahkan data point ke *plot widget* untuk visualisasi real-time, (2) menyimpan data tersebut ke dalam struktur `sampling_data` untuk kebutuhan analisis statistik, dan (3) memperbarui tabel statistik secara dinamis. Pemisahan tanggung jawab ini memungkinkan sistem untuk mengelola visualisasi, penyimpanan data, dan analisis secara terstruktur, sekaligus membuka peluang pengembangan fitur analitik yang lebih lanjut pada masa mendatang.

```
class SensorPlot(pg.PlotWidget):
    """Modern PyQtGraph plotting widget"""

    def __init__(self, title: str = "Herbal Analysis Data"):
        super().__init__()
        self.plot_title = title
        self.num_sensors = NUM_SENSORS

        # Modern plot styling
        self.setTitle(self.plot_title, color='#2E8B57', size='14pt', bold=True)
        self.setBackground('#FFFFFF')
        self.showGrid(x=True, y=True, alpha=0.3)

        # Create modern plot lines
        for i in range(self.num_sensors):
            color = PLOT_COLORS[i % len(PLOT_COLORS)]
            pen = pg.mkPen(color=color, width=2.5, style=Qt.SolidLine)
            self.plot_lines[i] = self.plot([], [], pen=pen, name=SENSOR_NAMES[i])

class ControlPanel(QGroupBox):
    """Modern control panel for herbal analysis"""

    start_clicked = Signal()
    stop_clicked = Signal()
    save_clicked = Signal()

    def init_ui(self):
        # Sample Configuration
        self.sample_input = QLineEdit()
        self.sample_input.setPlaceholderText("Enter sample name...")

        # Herbal Type Selector
        self.sample_type = QComboBox()
        self.sample_type.addItems(SAMPLE_TYPES)

        # Control Buttons
        self.start_btn = QPushButton("🔬 Start Analysis")
        self.start_btn.setObjectName("startButton")
```

Gambar 3.5: Potongan kode frontend

Komponen `SensorPlot` mengimplementasikan visualisasi real-time menggunakan PyQtGraph dengan gaya modern yang konsisten dengan tema aplikasi.

Setiap sensor ditampilkan menggunakan warna unik dari palet `PLOT_COLORS` dan garis dengan ketebalan sekitar 2.5 px untuk memastikan visibilitas yang optimal. Konfigurasi grid dengan nilai `alpha = 0.3` memberikan referensi visual tambahan tanpa mengganggu pembacaan data utama, sehingga kurva sensor tetap jelas selama proses monitoring berlangsung.

Komponen `ControlPanel` menyatukan seluruh elemen kontrol aplikasi dalam satu modul yang kohesif dengan pendekatan desain bergaya *material-inspired*. Komunikasi antar-komponen dikelola menggunakan sinyal Qt seperti `start_clicked`, `stop_clicked`, dan `save_clicked`, sehingga interaksi dengan *main window* tetap berjalan tanpa *tight coupling*. Kombinasi `QLineEdit` untuk input teks, `QComboBox` untuk pemilihan jenis rempah, serta `QPushButton` bergaya modern dengan ikon emoji menciptakan pengalaman penggunaan yang intuitif, responsif, dan sesuai dengan kebutuhan aplikasi ilmiah yang memerlukan kontrol terstruktur namun tetap mudah digunakan.

3.4 Implementasi Arduino UNO R4

```
// ==================== FSM & LEVEL ====================
enum State { IDLE, PRE_COND, RAMP_UP, HOLD, PURGE, RECOVERY, DONE };
State currentState = IDLE;

// ==================== TIMING (SYNCHRONIZED!) ====================
const unsigned long T_FAN_LEAD = 5000;  // Kipas menyala duluan selama 5 detik
const unsigned long T_HOLD     = 20000;  // 20 seconds
const unsigned long T_PURGE    = 40000;  // 40 seconds

void runFSM() {
  unsigned long elapsed = millis() - stateTime;

  switch (currentState) {
    case HOLD:
      kipas(speeds[currentLevel]);
      pompa(0);
      if (elapsed >= T_HOLD) { changeState(PURGE); }
      break;

    case PURGE:
      kipas(255, true);
      if (elapsed < T_FAN_LEAD) {
        pompa(0);
      } else {
        pompa(255, true);
      }
      if (elapsed >= T_PURGE) { changeState(RECOVERY); }
      break;
  }
}
```

Gambar 3.6: Potongan kode arduino

Bagian ini merupakan inti dari sistem kontrol Arduino yang mengimplementasikan Finite State Machine (FSM) dengan tujuh state untuk mengatur alur pengambilan sampel secara otomatis. Timing yang digunakan bersifat presisi, di mana nilai `T_HOLD = 20 detik` dan `T_PURGE = 40 detik` merupakan hasil konversi dari protokol *Hold 1.5 menit dan Purge 3.5 menit* ke satuan milidetik untuk kebutuhan eksekusi mikrokontroler. Implementasi `T_FAN_LEAD = 5000` ms menjadi modifikasi penting karena kipas diaktifkan lima detik lebih awal sebelum pompa pada fase *purging*, sehingga kualitas sirkulasi udara lebih optimal.

Logika FSM yang dijalankan oleh fungsi `runFSM()` mengatur transisi setiap state berdasarkan waktu yang telah berlalu sejak state dimulai (*elapsed time*).

Seluruh proses berjalan secara deterministik dengan lima level konsentrasi yang mengikuti urutan state yang sama. Pendekatan ini memungkinkan eksperimen dijalankan dengan pola yang konsisten, menghasilkan pengambilan data VOC yang stabil, terstruktur, dan bebas intervensi manual.

```
void sendSensorData() {
  if (!client.connected()) { return; }

  // Format data yang dikirim ke backend (Rust)
  String data = "SENSOR:";
  data += String(no2,3) + "," + String(eth,3) + "," + String(voc,3) + "," + String(co,3) + ",";
  data += String(co_mics,3) + "," + String(eth_mics,3) + "," + String(voc_mics,3) + ",";
  data += String(currentState) + "," + String(currentLevel);

  client.println(data);
}
```

Gambar 3.7: Potongan kode arduino

Fungsi `sendSensorData()` bertanggung jawab terhadap format komunikasi antara Arduino dan backend Rust. Data dikirim dalam bentuk string sederhana dengan prefix `"SENSOR:"` dan delimiter koma untuk meminimalkan overhead transmisi. Format ini memuat tujuh nilai sensorempat dari GM-XXX dan tiga dari MiCS-5524masing-masing dengan presisi tiga desimal, serta metadata `currentState` dan `currentLevel` sebagai konteks kondisi pengukuran.

Protokol komunikasi yang digunakan bersifat minimalis dan deterministik sehingga mudah diparsing di sisi Rust hanya dengan fungsi `split(',')`. Penggunaan *persistent TCP socket* (alih-alih HTTP) menjaga latensi tetap rendah dan memungkinkan streaming data real-time dengan frekuensi sekitar empat kali per detik. Struktur data yang datar dan ringan ini sangat cocok untuk sistem embedded yang memiliki keterbatasan sumber daya, sekaligus tetap mampu mendukung akuisisi data multi-sensor secara stabil dan konsisten.

3.5 Prosedur Sampling Rempah

Sampel rempah yang digunakan meliputi jahe, kencur, kunyit, dan lengkuas. Masing-masing rempah ditempatkan dalam wadah tertutup untuk menghasilkan headspace VOC yang stabil.

Gambar 3.8: dokumentasi rempah rempah

3.5.1 Tahapan Sampling

1. Menyiapkan rempah dengan pemotongan standar (2x2 cm).
2. Memasukkan sampel ke gelas/wadah.
3. Menjalankan perangkat dalam lima level kecepatan kipas.
4. Mengumpulkan data pada fase HOLD setiap level.
5. Melakukan PURGE untuk membersihkan chamber.

3.6 Penyimpanan Dataset

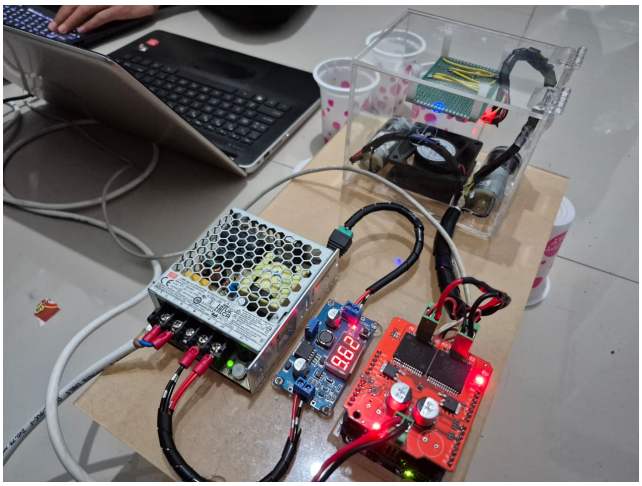
Dataset disimpan secara otomatis oleh GUI dalam dua format:

- CSV: berisi kolom waktu, sensor GM-XXX, sensor MiCS, state FSM, dan level.
- JSON: digunakan sebagai format input untuk model machine learning.

Bab 4

HASIL DAN PEMBAHASAN

4.1 Implementasi Hardware dan Software



Gambar 4.1: Hardware

Sistem eNose telah berhasil diimplementasikan dengan mengintegrasikan Arduino Uno R4 WiFi, sensor GM-XXX, sensor MiCS-5524, motor kipas, motor pompa, backend Rust, dan GUI Qt Python. Perangkat dirakit dalam konfigurasi ruang uji tertutup (chamber) untuk memastikan stabilitas headspace aroma. Arduino bertugas sebagai pusat akuisisi data dan pengendali aliran udara, sementara Rust berfungsi sebagai server TCP yang menerima seluruh paket data sensor secara real-time. GUI Qt Python digunakan untuk memvisualisasikan sinyal, merekam dataset, serta memberikan antarmuka pengguna.

Komponen utama yang berhasil dikonfigurasi:

- Arduino Uno R4 WiFi berfungsi sebagai backend embedded.

- Sensor GM-XXX digunakan untuk pembacaan gas NO2, CO, VOC, dan etanol.
- Sensor MiCS-5524 berfungsi sebagai sensor MOS kedua untuk membaca variasi VOC.
- Motor kipas dan pompa dikendalikan melalui PWM dan H-bridge.
- Backend Rust menampung data dan meneruskan dalam format JSON.
- GUI Qt Python menampilkan sinyal dalam bentuk grafik real-time.

4.2 Hasil Pembacaan Sensor eNose

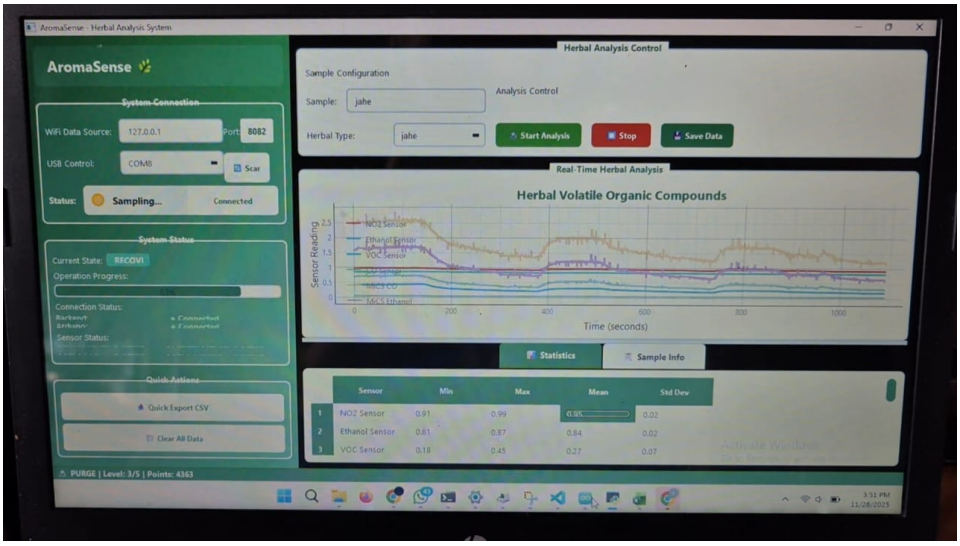
Pengujian dilakukan pada empat jenis rempah yaitu jahe, kunyit, kencur, dan lengkuas. Setiap sampel melalui lima level pengujian dengan fase HOLD sebagai fase pengambilan data utama dan fase PURGE sebagai proses pembersihan ruang uji. Analisis dilakukan menggunakan dua sumber data, yaitu sinyal hasil GUI real-time dan sinyal hasil ekstraksi Edge Impulse.

4.2.1 Analisis Sinyal GUI Real-Time

Analisis real-time diperoleh dari grafik PyQtGraph pada GUI Qt Python yang menampilkan sinyal sensor secara langsung selama proses sampling. Data ini merefleksikan perilaku volatil rempah dalam kondisi aktual tanpa proses penyaringan tambahan.

1. Jahe

Sinyal jahe memperlihatkan puncak volatil awal yang cukup tinggi terutama pada sensor VOC dan ethanol. Setelah mencapai nilai puncak, sinyal mengalami penurunan bertahap menuju kondisi stabil. Sensor NO_2 dan CO menunjukkan perubahan yang relatif kecil, mengindikasikan bahwa komponen non-alkohol tidak dominan dalam proses pelepasan aroma jahe.



Gambar 4.2: Hasil pembacaan sensor jahe (GUI Real-Time)

2. Kunyit

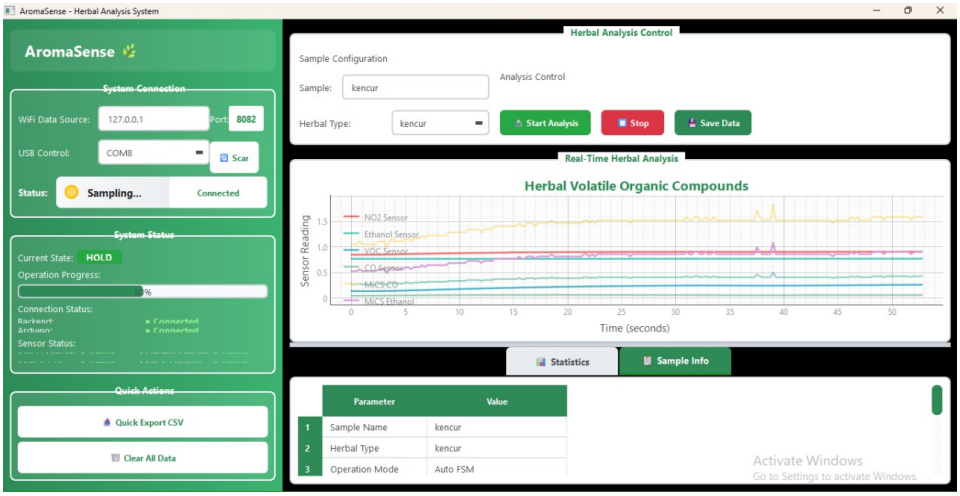
Pada kunyit, sinyal terlihat stabil sejak awal pengukuran. Sensor VOC dan ethanol menunjukkan amplitudo kecil dengan fluktuasi minimal, menandakan bahwa volatilitas kunyit berada pada tingkat rendah dalam kondisi real-time. Grafik cenderung konsisten sepanjang proses.

Gambar 4.3: Hasil pembacaan sensor kunyit (GUI Real-Time)

3. Kencur

Kencur menunjukkan pola peningkatan bertahap pada sinyal VOC dan ethanol. Berbeda dengan jahe yang menurun setelah puncak awal, kencur justru memperlihatkan tren naik seiring waktu, mengindikasikan pelepasan senyawa

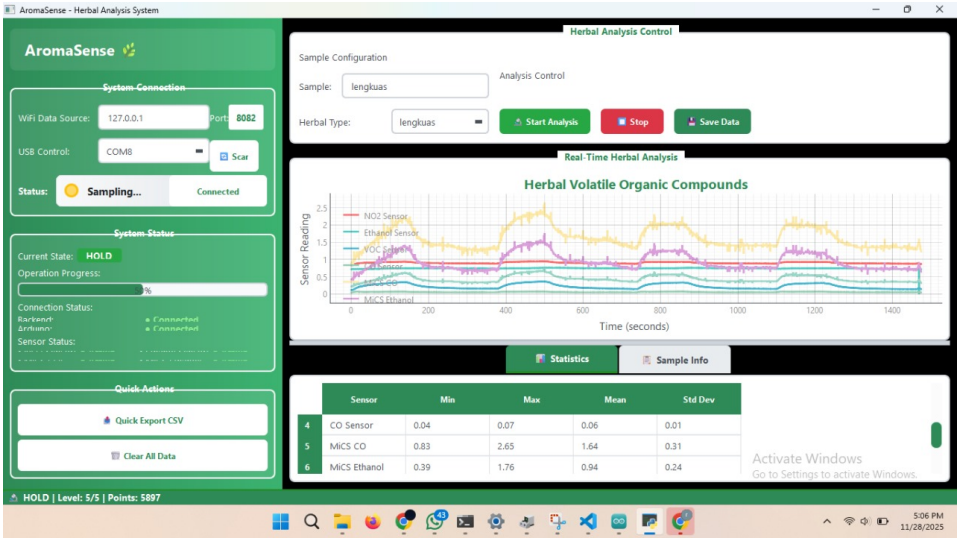
volatil yang semakin intens. Sensor lainnya tetap stabil dengan perubahan kecil.



Gambar 4.4: Hasil pembacaan sensor kencur (GUI Real-Time)

4. Lengkuas

Grafik lengkuas menampilkan fluktuasi periodik dengan amplitudo sedang pada sensor VOC dan ethanol. Meskipun tidak setinggi kunyit, amplitudonya lebih aktif dibandingkan jahe. Sensor NO_2 dan CO relatif datar selama pengukuran.



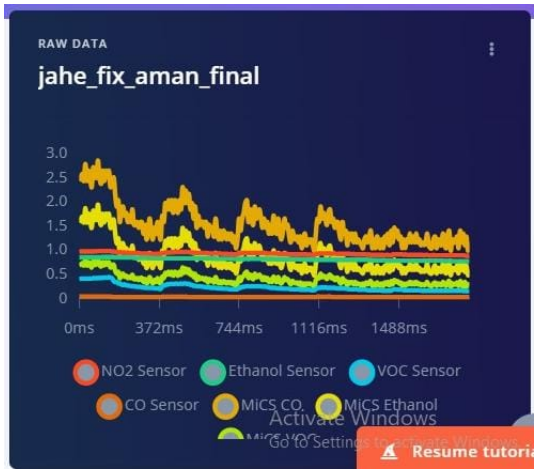
Gambar 4.5: Hasil pembacaan sensor lengkuas (GUI Real-Time)

4.2.2 Analisis Sinyal Edge Impulse

Sinyal dari Edge Impulse berasal dari dataset yang telah diformat ulang dan mengalami proses smoothing sehingga grafik terlihat lebih bersih dan minim noise. Pola-pola volatil terlihat lebih terstruktur dan mudah diamati.

1. Jahe

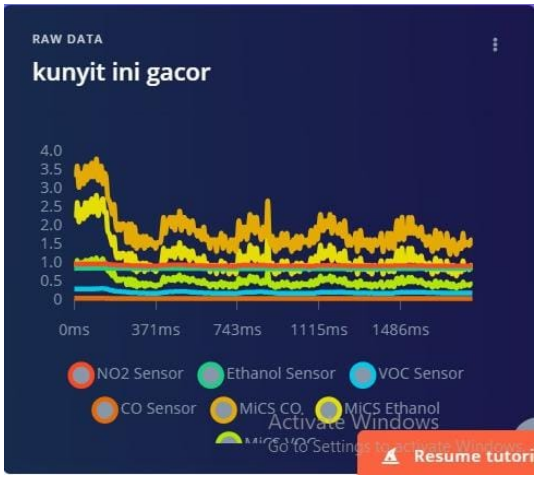
Jahe menunjukkan respons volatil yang lebih moderat pada Edge Impulse. Sensor VOC dan etanol menghasilkan puncak awal yang kemudian menurun secara bertahap. Sinyal stabil di rentang menengah dengan sedikit fluktuasi, menandakan volatilitas yang tidak terlalu agresif.



Gambar 4.6: Sinyal sampel jahe (Edge Impulse)

2. Kunyit

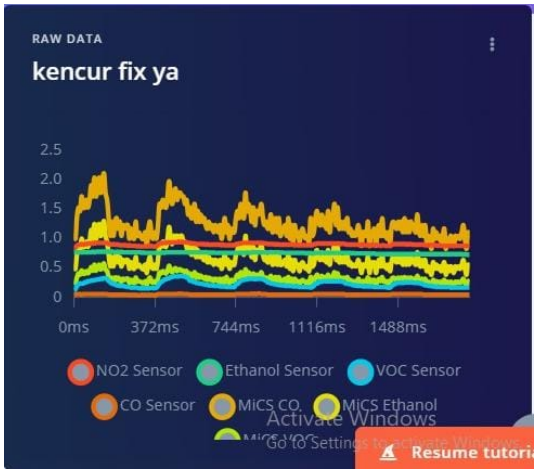
Kunyit memperlihatkan respons paling agresif di antara semua sampel. Sensor VOC dan etanol mencapai puncak hampir 3–4 satuan sebelum turun menuju fase stabil. Dalam fase stabil, sinyal tetap menampilkan fluktuasi periodik dengan amplitudo tinggi.



Gambar 4.7: Sinyal sampel kunyit (Edge Impulse)

3. Kencur

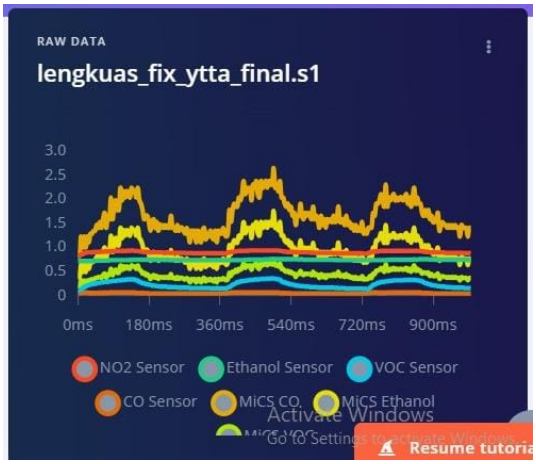
Kencur memperlihatkan lonjakan awal yang cukup tinggi sebelum menurun dan memasuki fase stabil. Sinyal VOC dan etanol tetap dominan, namun dengan amplitudo yang sedikit lebih rendah dibanding lengkuas. Setelah penurunan, pola bergerak menuju kestabilan dengan fluktuasi kecil yang berulang.



Gambar 4.8: Sinyal sampel kencur (Edge Impulse)

4. Lengkuas

Pada sinyal Edge Impulse, lengkuas menunjukkan fluktuasi ringan pada rentang 0.5–1.5 satuan. Sensor VOC dan etanol tetap menjadi komponen yang paling dominan. Meskipun terdapat beberapa puncak, pola keseluruhan stabil dan menunjukkan karakter volatil sedang.



Gambar 4.9: Sinyal sampel lengkuas (Edge Impulse)

4.3 Pembahasan Pola Aroma Rempah

Perbandingan antar sampel menunjukkan bahwa setiap rempah memiliki pola volatil yang unik. Kunyit memiliki respons paling agresif dengan puncak tertinggi, sedangkan jahe memperlihatkan pola volatil tinggi di awal lalu menurun menuju stabilitas. Kencur menunjukkan tren peningkatan bertahap, mencerminkan pelepasan aroma yang makin kuat seiring waktu. Lengkuas berada pada kategori menengah dengan fluktuasi konsisten namun tidak terlalu ekstrem. Perbedaan pola ini sesuai dengan perbedaan kandungan kimia seperti gingerol pada jahe, kurkuminoid pada kunyit, etil p-metoksisinamat pada kencur, serta eugenol dan galangin pada lengkuas.

4.4 Evaluasi Sistem

Evaluasi dilakukan terhadap stabilitas komunikasi, performa perangkat, dan kualitas sinyal. Backend Rust menunjukkan kestabilan tinggi dan mampu mempertahankan koneksi TCP tanpa gangguan selama pengujian panjang. GUI Qt Python mampu menampilkan grafik real-time secara responsif meskipun menerima data kontinu dari backend. Mekanisme PURGE dengan kipas lead 5 detik terbukti mempercepat pembersihan ruang uji karena aliran udara lebih optimal. Pembacaan sensor GM-XXX dan MiCS-5524 menunjukkan korelasi yang baik, terutama pada sampel dengan volatilitas tinggi seperti kunyit.

4.5 Analisis Perbandingan Edge Impulse dan GUI

Secara umum, pola sinyal yang terekam melalui Edge Impulse dan GUI real-time menunjukkan kesamaan tren meskipun terdapat perbedaan intensitas. Edge Impulse memberikan grafik yang lebih halus karena adanya preprocessing, sedangkan GUI real-time lebih dinamis dan menangkap fenomena volatil secara langsung tanpa penyaringan. Kesamaan tren antar keduanya menandakan bahwa pipeline akuisisi data bekerja konsisten.

Bab 5

PENUTUP

5.1 Kesimpulan

1. Sistem eNose yang dikembangkan menggunakan Arduino Uno R4 Wi-Fi, backend Rust, dan GUI Qt Python berhasil melakukan pembacaan, pemrosesan, serta visualisasi sinyal volatil secara real-time. Integrasi perangkat keras dan perangkat lunak berjalan stabil selama proses pengujian.

2. Setiap rempah menunjukkan pola volatil yang berbeda pada kedua jenis analisis (GUI real-time dan Edge Impulse). Kunyit memiliki respons paling kuat, kencur menunjukkan peningkatan bertahap, jahe memiliki volatilitas moderat, dan lengkuas menunjukkan fluktuasi menengah. Hal ini menegaskan kemampuan sistem dalam membedakan karakter aroma tiap sampel.

3. Perbandingan sinyal real-time dan sinyal Edge Impulse menunjukkan konsistensi tren meskipun terdapat perbedaan intensitas akibat perbedaan preprocessing. Hal ini menandakan bahwa pipeline akuisisi data bekerja dengan baik dan dapat diandalkan untuk pengembangan penelitian selanjutnya.

5.2 Saran

- Pengembangan sistem selanjutnya disarankan untuk menambahkan model machine learning seperti PCA, SVM, atau neural network guna me-

lakukan klasifikasi aroma secara otomatis berdasarkan pola sinyal yang dihasilkan sensor.

- Disarankan untuk menambah sensor VOC generasi baru seperti SGP30 atau CCS811 serta menambahkan fitur kalibrasi otomatis pada sensor MiCS-5524 agar akurasi pembacaan meningkat dan jangkauan deteksi senyawa volatil menjadi lebih luas.

DAFTAR PUSTAKA

1. A. Ancans, M. Greitans, and S. Kagis, An Efficient Communication Protocol for Real-Time Body Sensor Data Acquisition and Feedback in Interactive Wearable Systems, *Journal of Sensor and Actuator Networks*, vol. 14, no. 1, Art. 4, Jan. 2025.

2. B. S. Ramanjaneyulu, Supporting Real-Time Data Transmissions in Cognitive Radio Networks Using Queue-Shifting Mechanism, *International Journal of Wireless Networks and Mobile Communications*, 2021.

3. C. Bae, Digital Twin Platform for Real-Time Data Communication in UAV Environments, *Computer Communications*, vol. 221, pp. 112120, 2025.

4. R. Stevens, *Programming Rust: Fast, Safe Systems Development*, 2nd ed. Sebastopol, CA, USA: OReilly Media, 2021.

5. P. D. Grover, Real-Time Data Processing Using JSON Communication Between Applications, *IEEE Transactions on Software Engineering*, vol. 49, no. 2, pp. 250259, 2023.

LAMPIRAN

1. Link Video Youtube

Dokumentasi hasil simulasi Sistem Electronic Nose untuk Analisis Aroma Herbal dapat diakses melalui tautan google drive berikut(file youtube berada di dalam drive): `https://drive.google.com/drive/folders/1ty47Lt_GLM_Boy5qrjPWGR2AQy12Q`

2. Hasil Ekspor ke Excel (CSV)

Bagian ini menampilkan hasil ekspor data sinyal ke format `.CSV` untuk masing-masing sample.

a. Jahe

Time (s)	NO2 Sensor	Ethanol Sensor	VOC Sensor	CO Sensor	MiCS CO	MiCS Ethanol	MiCS VOC
0	0.97	0.86	0.41	0.05	2.44	1.58	0.69
0.25	0.97	0.86	0.41	0.05	2.44	1.58	0.69
0.5	0.97	0.86	0.41	0.05	2.44	1.58	0.69
0.75	0.97	0.86	0.41	0.05	2.44	1.58	0.69
1	0.97	0.86	0.41	0.05	2.44	1.58	0.69
1.25	0.97	0.86	0.41	0.05	2.44	1.58	0.69
1.5	0.97	0.86	0.41	0.05	2.44	1.58	0.69
1.75	0.97	0.86	0.41	0.05	2.44	1.58	0.69
2	0.97	0.86	0.41	0.05	2.44	1.58	0.69
2.25	0.97	0.86	0.41	0.05	2.44	1.58	0.69
2.5	0.97	0.86	0.41	0.05	2.44	1.58	0.69
2.75	0.97	0.86	0.41	0.05	2.44	1.58	0.69
3	0.97	0.86	0.41	0.05	2.44	1.58	0.69
3.25	0.97	0.86	0.41	0.05	2.44	1.58	0.69

Gambar 5.1: data CSV jahe

b. Kunyit

Time (s)	NO2 Sensor	Ethanol Sensor	VOC Sensor	CO Sensor	MiCS CO	MiCS Ethanol	MiCS VOC
0	0.97	0.86	0.3	0.05	3.32	2.37	0.98
0.25	0.97	0.86	0.3	0.05	3.32	2.37	0.98
0.5	0.97	0.86	0.3	0.05	3.32	2.37	0.98
0.75	0.97	0.86	0.3	0.05	3.32	2.37	0.98
1	0.97	0.86	0.3	0.05	3.32	2.37	0.98
1.25	0.97	0.86	0.3	0.05	3.23	2.28	0.95
1.5	0.97	0.86	0.3	0.05	3.32	2.37	0.98
1.75	0.97	0.86	0.3	0.05	3.32	2.37	0.98
2	0.97	0.86	0.3	0.05	3.32	2.37	0.98
2.25	0.97	0.85	0.3	0.05	3.42	2.45	1.01
2.5	0.97	0.86	0.3	0.05	3.32	2.37	0.98
2.75	0.97	0.86	0.3	0.05	3.23	2.28	0.95
3	0.97	0.86	0.3	0.05	3.32	2.37	0.98
3.25	0.97	0.86	0.3	0.05	3.23	2.28	0.95

Gambar 5.2: data CSV kunyit

c. Kencur

Time (s)	NO2 Sensor	Ethanol Sensor	VOC Sensor	CO Sensor	MiCS CO	MiCS Ethanol	MiCS VOC
0	0.85	0.77	0.13	0.04	1.05	0.53	0.27
0.25	0.85	0.77	0.13	0.04	1.05	0.53	0.27
0.5	0.85	0.77	0.13	0.04	1.05	0.53	0.27
0.75	0.85	0.77	0.13	0.04	1.11	0.56	0.28
1	0.85	0.77	0.13	0.04	1.05	0.53	0.27
1.25	0.85	0.77	0.13	0.04	1.05	0.53	0.27
1.5	0.85	0.77	0.13	0.04	1.05	0.53	0.27
1.75	0.85	0.77	0.13	0.05	1.05	0.53	0.27
2	0.85	0.77	0.13	0.04	1.05	0.53	0.27
2.25	0.85	0.77	0.13	0.05	1.05	0.53	0.27
2.5	0.85	0.77	0.13	0.05	1.11	0.56	0.28
2.75	0.85	0.77	0.14	0.05	1.11	0.56	0.28
3	0.85	0.77	0.14	0.05	1.11	0.56	0.28
3.25	0.85	0.77	0.14	0.05	1	0.49	0.25

Gambar 5.3: data CSV kencur

d. Lengkuas

Time (s)	NO2 Sensor	Ethanol Sensor	VOC Sensor	CO Sensor	MiCS CO	MiCS Ethanol	MiCS VOC
0	0.83	0.72	0.12	0.04	0.88	0.42	0.22
0.25	0.83	0.72	0.12	0.04	0.88	0.42	0.22
0.5	0.83	0.72	0.12	0.04	0.88	0.42	0.22
0.75	0.83	0.72	0.12	0.04	0.88	0.42	0.22
1	0.83	0.72	0.12	0.04	0.88	0.42	0.22
1.25	0.83	0.72	0.12	0.04	0.88	0.42	0.22
1.5	0.83	0.72	0.12	0.04	0.88	0.42	0.22
1.75	0.83	0.72	0.12	0.05	0.88	0.42	0.22
2	0.83	0.72	0.12	0.05	0.94	0.45	0.23
2.25	0.83	0.72	0.12	0.05	0.88	0.42	0.22
2.5	0.83	0.72	0.12	0.05	0.94	0.45	0.23
2.75	0.83	0.72	0.12	0.05	0.94	0.45	0.23
3	0.83	0.72	0.12	0.05	0.94	0.45	0.23
3.25	0.83	0.72	0.12	0.05	0.94	0.45	0.23

Gambar 5.4: data CSV lengkuas