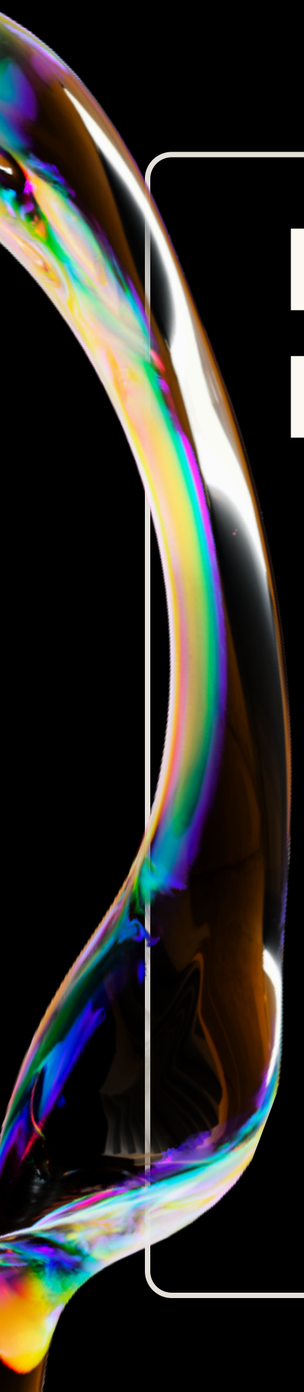
SISTEM PENGOLAHAN SINYA, TEKNIK INSTRUMENTASI ITS

SISTEM ELECTRONIC NOSE UNTUK ANALISIS AROMA HERBAL

Implementasi Backend Rust, Frontend Python,
dan Mikrokontroler Arduino

Group members :

Galen Zahid Wajendra - 20422241044
Rijal Difaul Haq - 2042241097
Sintia Ompusunggu - 2042241113



LATAR BELAKANG & PERMASALAHAN

- **Latar Belakang:**
  - Kebutuhan sistem monitoring aroma herbal secara real-time
  - Metode konvensional (GC-MS, HPLC) mahal dan tidak real-time
  - Potensi e-Nose sebagai alternatif efisien

- **Permasalahan:**
  - Bagaimana membangun sistem e-Nose lengkap dengan kontrol otomatis?
  - Bagaimana mengintegrasikan Arduino, backend Rust, dan frontend Python?
  - Bagaimana memvisualisasi data sensor secara real-time?




TUJUAN

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

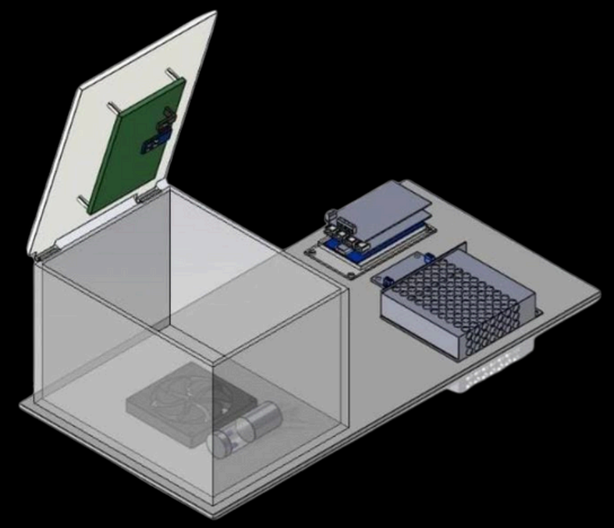
MANFAAT

- Sistem otomatis untuk penelitian herbal
- Platform untuk dataset aroma herbal
- Dasar untuk pengembangan machine learning

DESAIN SISTEM (DIAGRAM BLOK)

SENSOR → ARDUINO UNO R4 → BACKEND RUST → Frontend Python

Hardware: Arduino Uno R4 WiFi + sensor GM-XXX + MiCS-5524
Backend: Server Rust (TCP communication)
Frontend: GUI Python (PyQt5 + PyQtGraph)



IMPLEMENTASI ARDUINO (KONTROL SISTEM)

- Fitur Khusus:
  - Kipas menyala 5 detik lebih dulu saat PURGE
  - 5 Level kecepatan untuk gradasi stimulus
  - Kontrol motor bidirectional untuk purging

```
// ================== FSM & LEVEL ==================
enum State { IDLE, PRE_COND, RAMP_UP, HOLD, PURGE, RECOVERY, DONE };
State currentState = IDLE;
unsigned long stateTime = 0;
int currentLevel   0;  // 0-4 → Rust +1 → 1-5
const int speeds[5] = {51, 102, 153, 204, 255}; // Kecepatan untuk Level 1-5
bool samplingActive = false;

// Variabel flag untuk mencetak pesan di Serial Monitor hanya sekali per fase PURGE
bool printFanLead = false;
bool printBoth = false;

// ================== TIMING (SYNCHRONIZED!) ==================
const unsigned long T_FAN_LEAD = 5000;  // Kipas menyala duluan selama 5 detik
const unsigned long T_PRECOND  = 5000;   // 5 seconds
const unsigned long T_RAMP     = 3000;   // 3 seconds
const unsigned long T_HOLD     = 20000;  // 20 seconds
const unsigned long T_PURGE    = 40000;  // 40 seconds
const unsigned long T_RECOVERY = 5000;   // 5 seconds
unsigned long lastSend = 0;
unsigned long lastReconnect = 0;

// ================== MOTOR CONTROL ==================
void kipas(int speed, bool buang = false) {
  digitalWrite(DIR_KIPAS_1, buang ? LOW : HIGH);
  digitalWrite(DIR_KIPAS_2, buang ? HIGH : LOW);
  analogWrite(PWM_KIPAS, speed);
}

void pompa(int speed, bool buang = false) {
  // buang = true → pompa mengeluarkan udara (untuk PURGE)
  // buang = false → pompa normal (untuk fase lain jika diperlukan)
  // CATATAN: Kalau arah masih terbalik, tukar LOW/HIGH di bawah ini
  digitalWrite(DIR_POMPA_1, buang ? LOW : HIGH);
  digitalWrite(DIR_POMPA_2, buang ? HIGH : LOW);
  analogWrite(PWM_POMPA, speed);
}
```

IMPLEMENTASI BACKEND RUST

Architecture:

- Server TCP dual-port (8081 untuk Arduino, 8082 untuk GUI)
- Broadcast channel untuk komunikasi multi-client
- JSON serialization untuk data exchange

Fungsi Utama:

- Parsing data "SENSOR:value1,value2,..."
- Forward data ke semua frontend terhubung
- Manage connection status

```rust
// HIGHLIGHT KODE INI DI SLIDE 8
tokio::select! {
    // 1. Baca Data Sensor dari Arduino
    Ok(Some(line)) = line_reader.next_line() => {
        if line.starts_with("SENSOR:") {
            process_sensor_data(&line, &tx_sensor).await;
        }
    }
    // 2. Kirim Command ke Arduino (Jika ada dari UI)
    Ok(cmd) = rx_cmd.recv() => {
        println!("📤 Forwarding to Arduino: {}", cmd);
        if writer.write_all(cmd.as_bytes()).await.is_err() { break; }
    }
}
```

```python
# HIGHLIGHT KODE INI DI SLIDE 9
def add_data_point(self, time: float, sensor_values: list):
    """Add new data point to plot"""
    self.time_data = np.append(self.time_data, time)

    # Update data per sensor
    for i, value in enumerate(sensor_values[:self.num_sensors]):
        self.sensor_data[i] = np.append(self.sensor_data[i], value)

    # Update plot lines
    for i in range(self.num_sensors):
        self.plot_lines[i].setData(self.time_data, self.sensor_data[i])
```

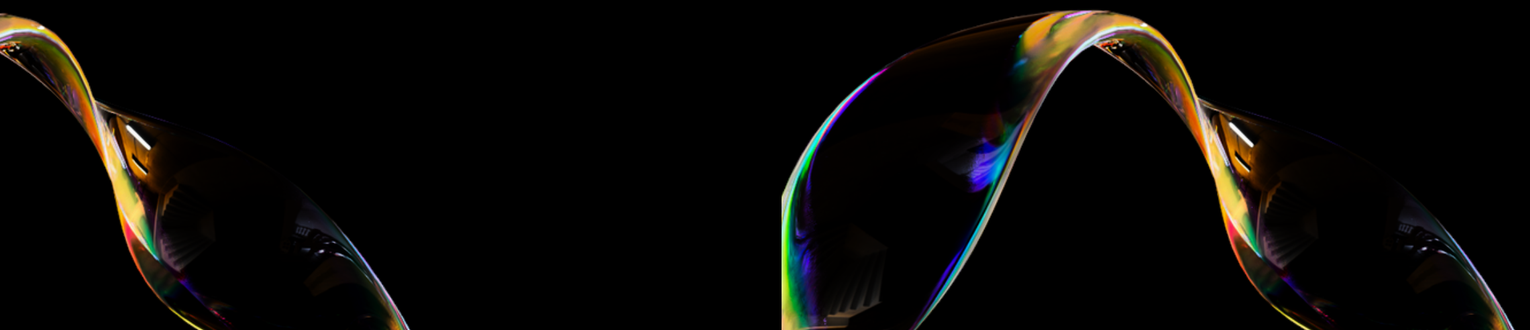
IMPLEMENTASI FRONTEND PYTHON

Architecture:

- Server TCP dual-port (8081 untuk Arduino, 8082 untuk GUI)
- Broadcast channel untuk komunikasi multi-client
- JSON serialization untuk data exchange

Fungsi Utama:

- Parsing data "SENSOR:value1,value2,..."
- Forward data ke semua frontend terhubung
- Manage connection status



```python
# HIGHLIGHT KODE INI DI SLIDE 9
def add_data_point(self, time: float, sensor_values: list):
    """Add new data point to plot"""
    self.time_data = np.append(self.time_data, time)

    # Update data per sensor
    for i, value in enumerate(sensor_values[:self.num_sensors]):
        self.sensor_data[i] = np.append(self.sensor_data[i], value)

    # Update plot lines
    for i in range(self.num_sensors):
        self.plot_lines[i].setData(self.time_data, self.sensor_data[i])
```

```rust
// HIGHLIGHT KODE INI DI SLIDE 8
tokio::select! {
    // 1. Baca Data Sensor dari Arduino
    Ok(Some(line)) = line_reader.next_line() => {
        if line.starts_with("SENSOR:") {
            process_sensor_data(&line, &tx_sensor).await;
        }
    }
    // 2. Kirim Command ke Arduino (Jika ada dari UI)
    Ok(cmd) = rx_cmd.recv() => {
        println!("📤 Forwarding to Arduino: {}", cmd);
        if writer.write_all(cmd.as_bytes()).await.is_err() { break; }
    }
}
```

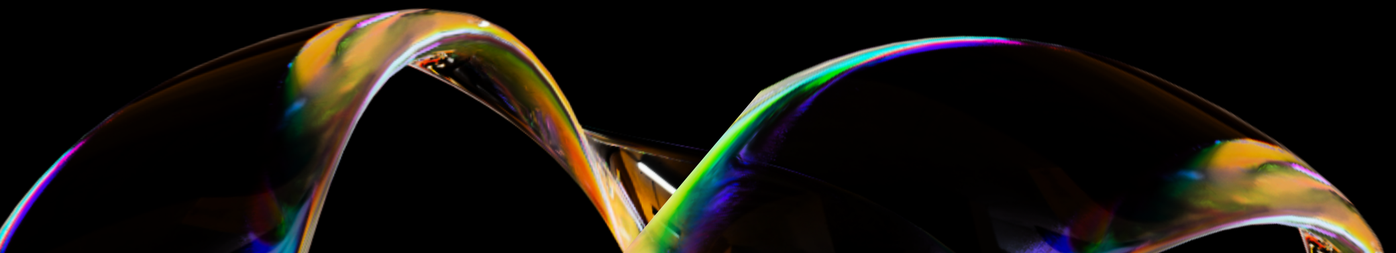

```python
# HIGHLIGHT KODE INI DI SLIDE 9
def add_data_point(self, time: float, sensor_values: list):
    """Add new data point to plot"""
    self.time_data = np.append(self.time_data, time)

    # Update data per sensor
    for i, value in enumerate(sensor_values[:self.num_sensors]):
        self.sensor_data[i] = np.append(self.sensor_data[i], value)

    # Update plot lines
    for i in range(self.num_sensors):
        self.plot_lines[i].setData(self.time_data, self.sensor_data[i])
```

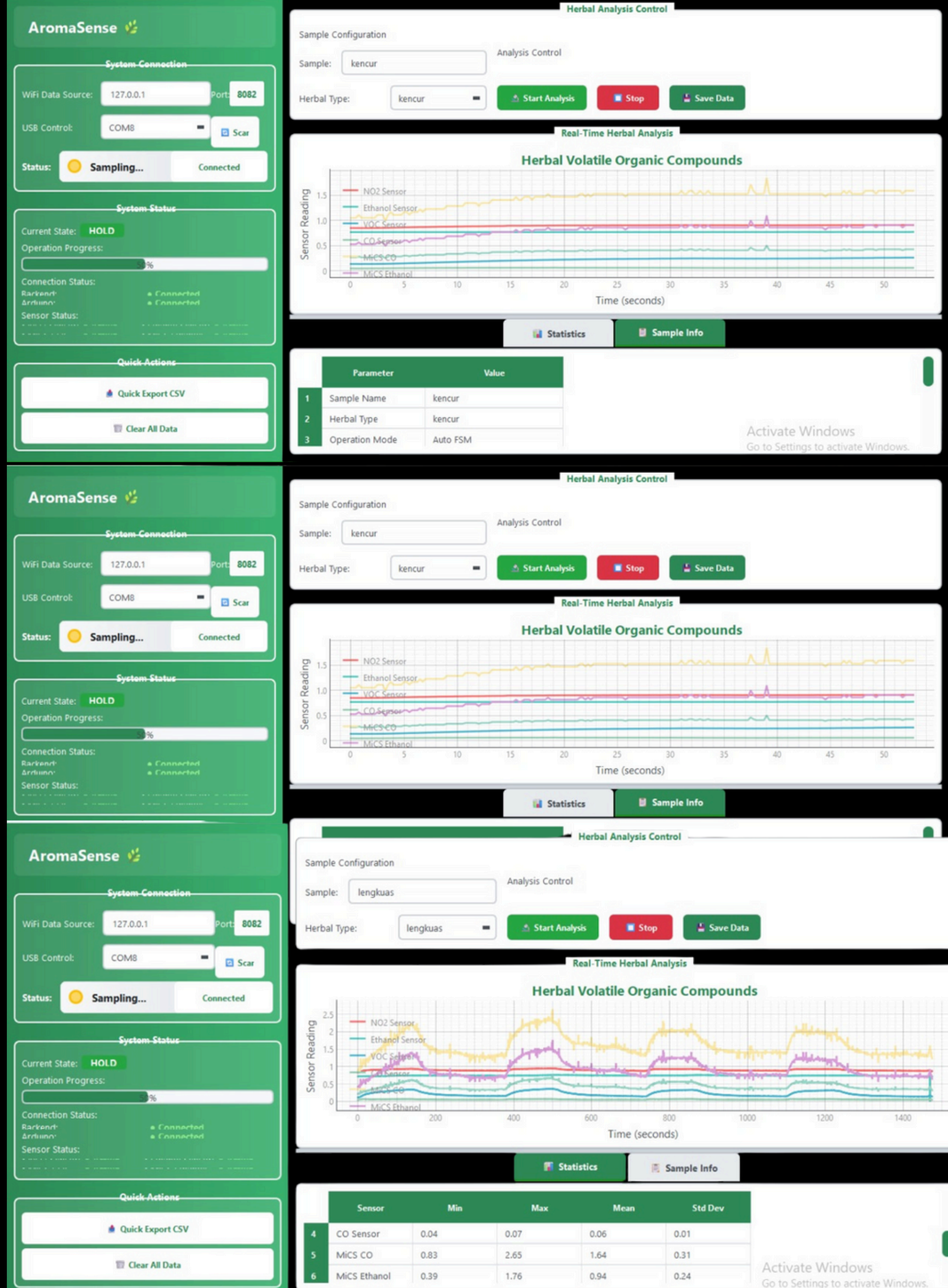
HASIL EKSPERIMEN (DATA SENSOR)

4 Jenis Herbal Tested:

- Jahe: Sinyal VOC stabil, moderat
- Kencur: Respon cepat, volatilitas tinggi
- Kunyit: Perubahan sinyal terbesar
- Lengkuas: Pola fluktuasi karakteristik

5 Level Testing: Gradasi kecepatan kipas





KESIMPULAN

1. Sistem e-Nose berhasil diimplementasikan dengan 3-layer architecture
2. FSM Arduino mengontrol proses sampling secara otomatis dan konsisten
3. Backend Rust efektif sebagai message broker dengan performa tinggi
4. GUI Python memberikan visualisasi real-time yang informatif
5. Sistem mampu membedakan pola aroma dari 4 jenis herbal berbeda

SISTEM PENGOLAHAN SINYA, TEKNIK INSTRUMENTASI ITS

THANK YOU

for your time and attention

Present by GROUP 7