LAPORAN PROJECT
INTERKONEKSI SISTEM INSTRUMENTASI

“Sistem Monitoring Suhu dan Kelembaban Pasca Panen Kopi untuk Menjaga Kualitas Fermentasi”

Kelompok 10:

Axel Fitra Ananda	2042231002
Lu’lu’ Rusyida Hamudyah	2042231058
Mushthafa Ali Akbar	2042231082

PRODI D4 TEKNOLOGI REKAYASA INSTRUMENTASI
DEPARTEMEN TEKNIK INSTRUMENTASI
FAKULTAS VOKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER

2025

Sistem Monitoring Suhu dan Kelembaban Pasca Panen Kopi untuk Menjaga Kualitas Fermentasi

I. Identitas Kelompok

- Nama Kelompok:
  Anggota Kelompok:
  1. Axel Fitra Ananda (2042231002)
  2. Lu'lu' Rusyida Hamudyah (2042231058)
  3. Mushtafa Ali Akbar (2042231082)
- Dosen Pengampu: Ahmad Radhy, S.Si., M.Si
- Mata Kuliah: Interkoneksi Sistem Instrumentasi

II. Latar Belakang

Sistem penilaian mutu biji kopi umumnya mengacu pada beberapa kriteria seperti asal, ekosistem, varietas, cara panen dan pascapanen, ukuran biji, densitas biji, nilai cacat, dan cita rasa. Salah satu tahapan penting yang menentukan nilai mutu adalah proses fermentasi. Fermentasi merupakan tahap krusial dalam proses pasca panen kopi yang berperan penting dalam pembentukan cita rasa akhir dari biji kopi (Heryanto, 2024). Selama fermentasi, lendir yang menempel pada biji kopi diuraikan oleh mikroorganisme yang menghasilkan senyawa-senyawa yang dapat memengaruhi rasa kopi. Namun, proses ini sangat sensitif terhadap kondisi lingkungan, terutama suhu dan kelembaban.

Di lapangan, petani kopi seringkali menghadapi tantangan dalam menjaga kestabilan suhu dan kelembaban selama prosesfer fermentasi. Faktor-faktor seperti fluktuasi suhu lingkungan, minimnya alat monitoring, dan keterbatasan pengetahuan teknis menyebabkan proses fermentasi sulit dikendalikan secara konsisten. Akibatnya, kualitas dan cita rasa kopi yang dihasilkan tidak seragam dan sulit diprediksi.

Seiring perkembangan teknologi digitalisasi di bidang pertanian, khususnya dalam sektor kopi, penggunaan Internet of Things (IoT) menjadi sangat relevan. IoT memungkinkan integrasi berbagai sensor seperti pengukur suhu, kelembaban, dan pH dalam satu jaringan yang mampu mengirimkan data secara real-time melalui protokol internet untuk dianalisis dan diolah. Penelitian seperti yang dilakukan oleh Nugraha et al. (2024) menunjukkan bahwa penerapan sistem monitoring berbasis IoT untuk proses fermentasi kopi secara nyata dapat meningkatkan akurasi pengukuran suhu dan pH hingga kisaran error di bawah 1,5%, sehingga mendukung proses fermentasi yang lebih optimal. Oleh karena itu, diperlukan sistem monitoring yang dapat memantau suhu dan kelembaban secara *real-time* selama proses fermentasi. Sistem ini memungkinkan petani untuk mengontrol kondisi fermentasi secara lebih akurat dan konsisten, sehingga dapat meningkatkan kualitas dan konsistensi produk kopi. Sistem monitoring suhu dan kelembaban pada proses fermentasi kopi ini dikembangkan berbasis sensor SHT20 yang mampu mendeteksi suhu dan kelembaban dengan akurasi yang tinggi. Sistem ini

dirancang menggunakan bahasa pemrograman Rust dan divisualisasikan menggunakan InfluxDB dan Grafana.

III. Rumusan Masalah

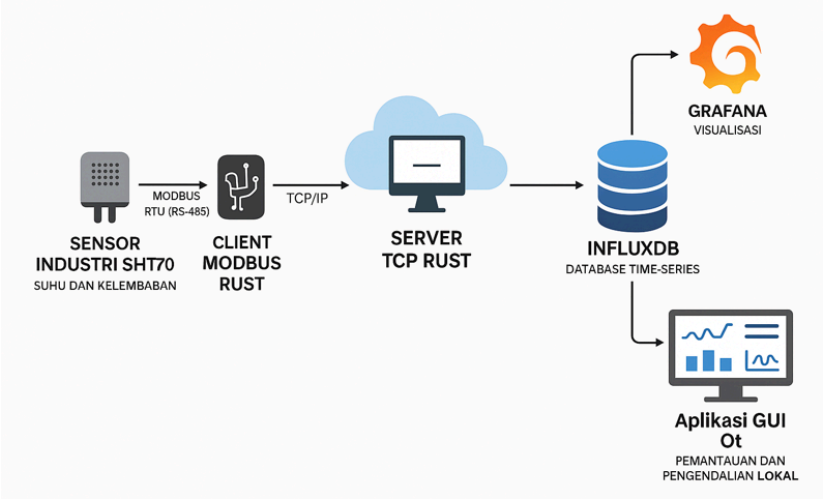
1. Bagaimana merancang sistem monitoring real-time berbasis sensor di gudang fermentasi kopi?
2. Bagaimana cara mengintegrasikan sensor agar dapat disimpan data historis suhu dan kelembabannya?
3. Bagaimana visualisasi data monitoring untuk mebantu petani dalam mengambil keputusan?

IV. Tujuan Proyek

1. Membuat sistem monitoring suhu dan kelembaban berbasis sensor SHT20.
2. Mengirimkan data secara real-time ke TCP Server dan menyimpannya di InfluxDB.
3. Menyediakan dashboard real-time menggunakan Grafana

V. Metodologi dan Arsitektur Sistem

1. Desain Arsitektur Sistem



Desain arsitektur sistem di atas adalah sistem monitoring suhu dan kelembaban untuk fermentasi kopi menggunakna sensor SHT20. Sensor SHT20 terhubung ke laptop melalui protokol Modbus RTU dan data sensor tersebut dibaca menggunakan program Rust Modbus Client. Hasil pembacaan sensor ini dikonversi ke format JSON. Data JSON dikirimkan ke TCP Server melalui port 9000, kemudian disimpan ke dalam InfluxDB sebagai *database time series*. Data yang tersimpan divisualisasikan menggunakan *dashboard* Grafana, sehingga pengguna atau petani kopi dapat memonitoring suhu dan kelembaban secara *real-time* melalui grafik.

2. **Deskripsi Komponen**

2.1 Sensor SHT 20

Sensor SHT20 merupakan sensor digital yang digunakan untuk mengukur suhu dan kelembaban relative dengan akurasi tinggi. Range operasional sensor SHT20 adalah -40°C hingga +125°C. Akurasi dari sensor SHT20 ini memiliki nilai yang tinggi, ±0.5°C untuk temperature dan 3% untuk RH (*Relative Humidity*) Sensor ini menggunakan interface I2C atau Modbus RTU untuk komunikasi datanya, dengan kecepatan komunikasi hingga 1MHz untuk I2C.

2.2 Modbus Client

Modbus Client dibangun menggunakan bahasa pemrogramman Rust yang berfungsi untuk membaca data dari sensor SHT20. Dengan mengimplementasikan protokol Modbus RTU, client ini dapat mengambil data sensor suhu dan kelembaban secara efektif.

2.3 TCP Server

TCP Server merupakan komponen dalam model komunikasi TCP/IP yang mendengarkan permintaan koneksi dari klien melalui protokol Transmission Control Protocol (TCP). Server ini bertanggung jawab untuk menerima koneksi yang masuk, memproses data yang diterima, dan mengirimkan respons yang sesuai. Pada *project* ini, TCP Server dibangun menggunakan bahasa pemrograman Rust untuk menerima data dari Modbust Client.

2.4 InfluxDB

InfluxDB adalah *database time series open source* yang dioptimalkan untuk menyimpan data historis dan memproses data dalam bentuk matriks untuk dianalisis. InfluxDB mendukung penggunaan protokol komunikasi ringan dan fleksibel, serta menyediakan bahasa *query* khusus seperti InfluxQL (mirip SQL) dan Flux untuk mendukung analisis data yang kompleks. Dengan kemampuan ini, pengguna dapat melakukan agregasi data, perhitungan statistik, maupun visualisasi data historis dan real-time dengan lebih mudah. Selain itu, InfluxDB memiliki integrasi yang baik dengan berbagai tools visualisasi dan sistem manajemen data seperti Grafana, Node-RED, dan berbagai platform cloud.

2.5 Grafana

Grafana adalah sebuah platform open-source yang digunakan untuk visualisasi data dan monitoring sistem secara real-time melalui antarmuka dashboard yang interaktif dan informatif. Grafana dirancang untuk menampilkan data dari berbagai sumber, termasuk database time-series seperti InfluxDB dan MySQL. Grafana memungkinkan pengguna untuk menggabungkan berbagai jenis data ke dalam satu tampilan yang kohesif dan mudah dipahami.

Salah satu keunggulan utama Grafana adalah kemampuannya dalam membuat dashboard kustom yang dapat memvisualisasikan metrik, log, dan peristiwa dalam bentuk grafik, tabel, gauge, heatmap, dan jenis visualisasi lainnya. Dashboard ini dapat diperbarui secara otomatis (real-time), sehingga sangat berguna untuk keperluan pemantauan performa sistem dan analisis data historis.

3. Format Payload

Contoh format JSON:


```
{
 "timestamp": "2025-04-28T15:45:00Z",
 "sensor_id": "SHT20-PascaPanen-001",
 "location": "Gudang Fermentasi 1",
 "process_stage": "Fermentasi",
 "temperature_celsius": 27.5,
 "humidity_percent": 65.2
}
```

VI. Implementasi dan Kode Program

1. Kode Rust Modbus Client

a. Penjelasan cara membaca sensor menggunakan Modbus RTU

Pembacaan sensor SHT20 menggunakan Modbus RTU dilakukan dengan cara menghubungkan sensor SHT20 dan modul RS485 to USB dengan konfigurasi pin sebagai berikut:

Sensor SHT20	Modul RS484
A	A
B	B
(-)	GND
(+)	VCC

Setelah menghubungkan pin dari kedua komponen tersebut, cek koneksi USB dari converter module melalui terminal dengan command “sudo dmesg | grep ttyUSB*”, akan terlihat converter terbaca di USB Port berapa. Ubah permission-nya “sudo chood a+rw /dev/ttyUSB1”. Lanjut ke direktori modpoll dengan prompt “cd modpoll/x86_64-linux-gnu”, direktori ini berisi file modpoll yang sesuai dengan sistem operasi Linux 64-bit. Agar file modpoll ini dapat dijalankan, perlu izin execute dengan prompt “chmod +x modpoll”, sehingga file modpoll bisa diexecute di terminal.

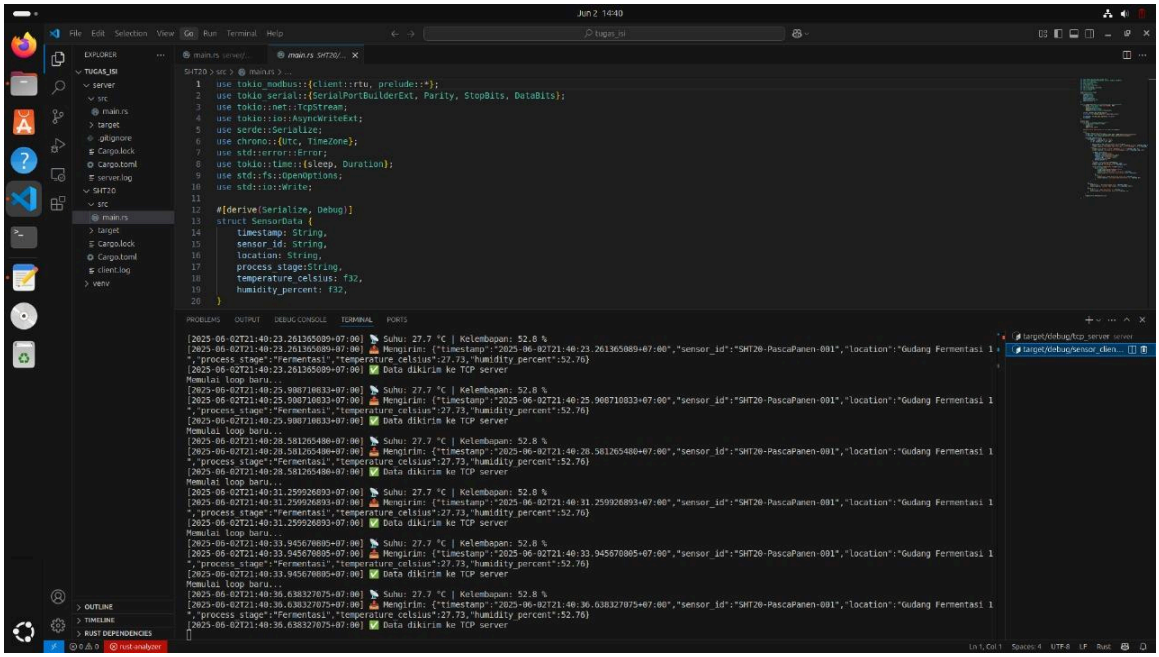
Kemudian, mencoba query data sensor SHT20 dengan command prompt “./modpoll -t3 -b 9600 -p none -m rtu -a 1 -r 0 -c 2 /dev/ttyUSB1” yang merupakan settingan dari modbusnya. Terminal akan menampilkan “Polling slave …” yang merupakan data register yang terbaca dari sensornya. Data yang dikirimkan oleh sensor masih memerlukan konversi dengan cara dibagi 10.

Nilai yang dikirimkan oleh sensor masih dalam bentuk integer dan perlu diubah ke dalam bentuk float dengan cara dibagi 10. Sensor pada umumnya mengirimkan data dalam format nilai yang dikali 10. Setelah berhasil dikonversi, data diubah ke dalam format JSON dan dikirimkan ke server melalui koneksi TCP menggunakan port 9000.

b. Cuplikan kode utama dan cara kirim data ke TCP Server.

```
use std::error::Error;
use std::net::TcpStream;
use std::io::Write;
```

```
use std::thread;
use std::time::Duration;
use chrono::Utc;
use serde::Serialize;

#[derive(Serialize)]
struct SensorData {
    timestamp: String,
    sensor_id: String,
    location: String,
    process_stage: String,
    temperature_celsius: f32,
    humidity_percent: f32,
}

fn main() -> Result<(), Box<dyn Error>> {
    // Koneksi ke TCP Server (sesuaikan IP dan port jika berbeda)
    let mut stream = TcpStream::connect("127.0.0.1:9000")?;
    println!("✅ Terhubung ke TCP Server");

    // Kirim identifikasi sebagai klien sensor
    stream.write_all(b"SENSOR_CLIENT\n")?;

    loop {
        // Simulasi data sensor
        let data = SensorData {
            timestamp: Utc::now().to_rfc3339(),
            sensor_id: "SHT20-01".to_string(),
            location: "Gudang A".to_string(),
            process_stage: "Sortir".to_string(),
            temperature_celsius: 25.3,
            humidity_percent: 60.7,
        };

        // Serialisasi ke JSON
        let json = serde_json::to_string(&data)? + "\n";

        // Kirim ke server
        stream.write_all(json.as_bytes())?;
        println!("📤 Data terkirim: {}", json.trim());

        // Tunggu 5 detik sebelum kirim lagi
        thread::sleep(Duration::from_secs(5));
    }
}
```



2. Kode Rust TCP Server

a. Penjelasan cara menerima data JSON, parsing, dan simpan ke InfluxDB

Pada tahap kedua, menerima data dari client menggunakan program Rust TCP Server. Server ini akan menerima data dari sensor SHT20 yang terhubung melalui protokol komunikasi RS485. Data dari sensor dikirim melalui jaringan TCP dalam format JSON yang berisi informasi suhu, kelembaban, timestamp, dan sensor_id.

Saat koneksi TCP berhasil dibuat, server mulai memantau dan menerima data yang dikirimkan melalui jalur komunikasi tersebut. Data diterima dalam bentuk string JSON, misalnya mencakup informasi seperti nama variabel proses, nilai temperature, nilai kelembaban, serta waktu pencatatan. Rust kemudian akan memproses data dengan melakukan parsing menggunakan library serde_json, yang akan menterjemahkan string JSON menjadi data yang jelas dapat dipahami. Hal ini membuat program dapat mengakses nilai yang dikirim seperti temperatur dan kelembaban.

Setelah parsing, data disimpan ke dalam InfluxDB menggunakan protokol Line Protocol untuk disimpan sebagai database time series, dengan sensor_id digunakan sebagai tag. Server mengirimkan data melalui HTTP request. Rust akan mengirimkan data InfluxDB yang konfigurasinya sudah diatur sesuai kebutuhan kita. Data yang terkirim akan langsung ditulis ke dalam bucket penyimpanan tertentu di InfluxDB, sesuai dengan pengaturan organisasi dan waktu.

b. Cuplikan kode fungsi utama.

```
#[tokio::main]
async fn main() -> Result<()> {
    dotenv().ok();

    let influx_token = env::var("INFLUX_TOKEN")
        .expect("INFLUX_TOKEN tidak ditemukan di environment.");

    let evm_rpc_url = env::var("EVM_RPC_URL")
        .unwrap_or_else(|_| "http://127.0.0.1:8545".to_string());
    let evm_private_key_hex = env::var("EVM_PRIVATE_KEY")
        .expect("EVM_PRIVATE_KEY tidak ditemukan di environment (format
hex tanpa 0x).");
    let evm_contract_address_str = env::var("EVM_CONTRACT_ADDRESS")
        .expect("EVM_CONTRACT_ADDRESS tidak ditemukan di
environment.");

    let provider = Provider::<Http>::try_from(evm_rpc_url)?;
    let provider = Arc::new(provider);

    let wallet: LocalWallet =
evm_private_key_hex.parse::<LocalWallet>()?.with_chain_id(1337u64);
    let client_evm = Arc::new(SignerMiddleware::new(provider.clone(),
wallet));
    let contract_address: Address = evm_contract_address_str.parse()?;
    let contract = SensorDataRecorder::new(contract_address,
client_evm.clone());

    let influx_url =
"http://localhost:8086/api/v2/write?org=ITS&bucket=ISI&precision=ns";
    let listener = TcpListener::bind("0.0.0.0:9000").await?;
    let http_client = Client::new();

    println!("🚪 TCP Server listening on port 9000...");

    let log_file = OpenOptions::new()
        .create(true)
        .append(true)
        .open("server.log")?;

    // Broadcast channel untuk kirim data ke GUI client
    let (tx_sensor_data, _) = broadcast::channel::<String>(16);

    // Loop utama menerima koneksi
    loop {
        let (socket, addr) = listener.accept().await?;
        let timestamp_main = chrono::FixedOffset::east_opt(7 * 3600)
```

```
            .unwrap()
            .from_utc_datetime(&chrono::Utc::now().naive_utc())
            .to_rfc3339();

        println!("[{}] 🔌 Koneksi masuk dari {}", timestamp_main, addr);

        // Kloning variabel untuk digunakan dalam task asynchronous
        let http_client_clone = http_client.clone();
        let influx_url_clone = influx_url.to_string();
        let influx_token_clone = influx_token.clone();
        let contract_clone = contract.clone();
        let log_file_clone = log_file.try_clone()?;
        let tx_sensor_data_clone = tx_sensor_data.clone();

        // Jalankan task async untuk setiap klien yang terhubung
        tokio::spawn(async move {
            if let Err(e) = handle_connection(
                socket,
                addr,
                http_client_clone,
                influx_url_clone,
                influx_token_clone,
                contract_clone,
                log_file_clone,
                tx_sensor_data_clone,
            ).await {
                eprintln!("[{}] Error handling client {}: {}",
                    chrono::FixedOffset::east_opt(7 * 3600)
                        .unwrap()
                        .from_utc_datetime(&chrono::Utc::now().naive_utc())
                        .to_rfc3339(),
                    addr, e);
            }
        });
    }
}
```

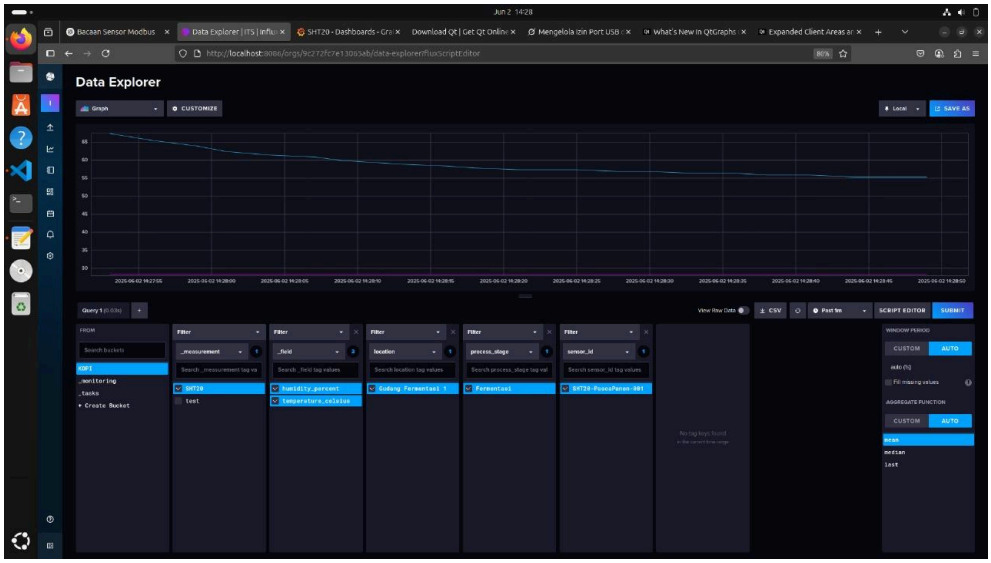
3. Konfigurasi InfluxDB dan Integrasi

1) Struktur measurement, tags, fields.

Dalam InfluxDB, struktur data disusun secara hierarki dan terdiri dari beberapa bagian penting seperti measurement, tags, dan fields. *Measurement* berfungsi sebagai pengelompokkan data, hal ini mewakili jenis pengukuran yang disimpan. *Measurement* pada proyek ini diberi label “SHT20”. *Field* adalah bagian yang menyimpan nilai utama dari hasil pengukuran. *Field* pada proyek ini

adalah humidity_percent dan temperature_celsius. *Tags* yaitu informasi pendukung yang disimpan dalam bentuk pasangan nama dan nilai, yang berfungsi untuk memberi konteks atau mengelompokkan data. *Tags* pada proyek ini adalah location “Gedung Fermentasi 1”, process_stage “Fermentasi”, dan sensor_id “SHT20-PascaPanen-001”.

2) Contoh hasil query di InfluxDB CLI

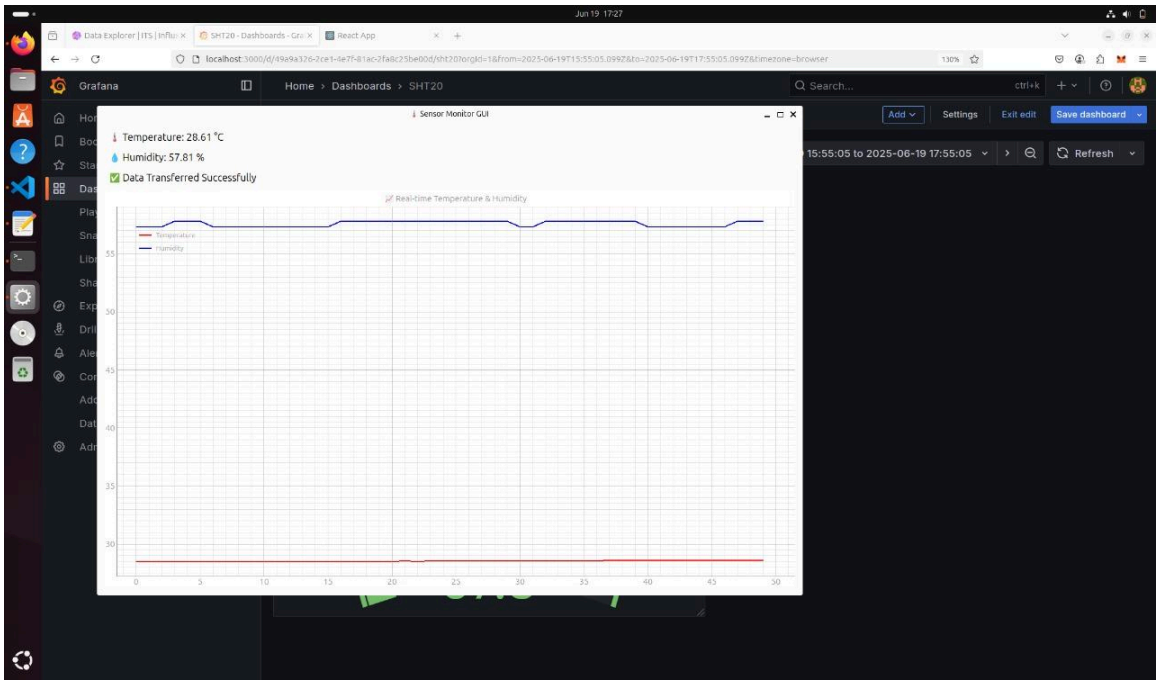


4. Dashboard Grafana

1) Desain visual (grafik suhu dan kelembaban)

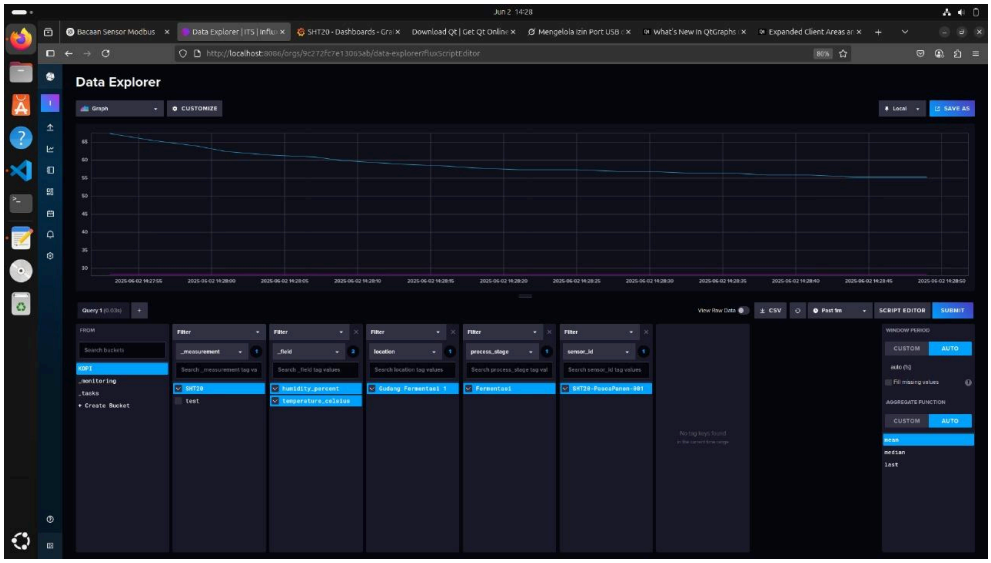
Langkah selanjutnya adalah menampilkan data visual melalui Grafana. Untuk menampilkan data suhu dan kelembaban secara *real-time*, Grafana diintegrasikan dengan InfluxDB. Dashboard menampilkan dua grafik utama yakni grafik suhu dan temperature terhadap waktu. Pada dashboard di bawah, grafik tervisualisasikan hingga detik 50. Grafana dapat dikonfigurasikan dengan fitur filter waktu seperti 1 jam, 24 jam, atau 7 hari terakhir.

2) Screenshot Dashboard

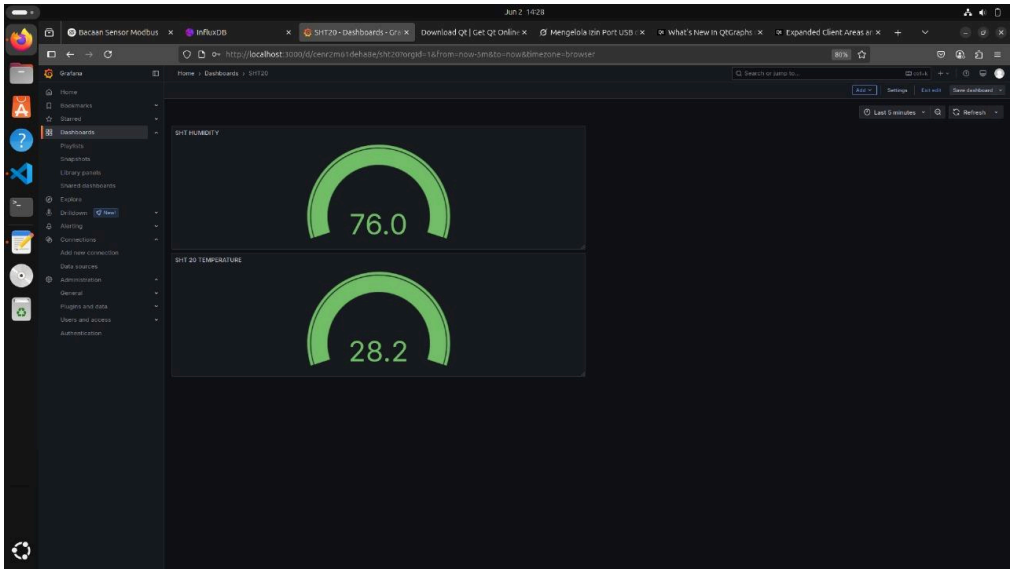


VII. Pengujian dan Hasil

1. Tabel hasil pembacaan suhu dan kelembaban selama proses fermentasi (minimal 24 jam pengambilan data).
2. Screenshot hasil penyimpanan di InfluxDB



3. Screenshot real-time dashboard di Grafana



4. Analisis apakah suhu dan kelembaban berada dalam rentang optimal (24-30°C dan 50-70%)

Suhu ideal untuk fermentasi kopi Arabika umumnya berkisar antara 25–30 °C, di mana aktivitas mikroba seperti bakteri asam laktat dan ragi berlangsung optimal. Suhu di bawah 20 °C dapat memperlambat fermentasi, sementara di atas 30 °C berisiko memicu over-fermentasi dan rasa off-flavor. Sedangkan kelembaban relatif dalam proses fermentasi biasanya dipertahankan di kisaran 50–70%, untuk meminimalkan resiko pertumbuhan jamur dan memastikan fermentasi berjalan konsisten.

Pada dashboard Grafana, nilai suhu yang terukur sudah berada dalam *range* optimal fermentasi yakni 28.2 °C. Namun, nilai kelembaban masih terlalu tinggi di angka 76% yang berpotensi untuk memperlambat fermentasi. Hal ini terjadi karena nilai tersebut tidak merepresentasikan kondisi ambient fermentasi kopi, melainkan berasal dari sensor di ruangan biasa, bukan di plant fermentasi. Akibatnya, nilai kelembaban bias karena pengaruh kelembaban ruang yang lebih tinggi dibanding fermentasi sebenarnya. Oleh karena itu, nilai kelembaban tampak tidak sesuai dengan range ideal fermentasi.

VIII. Kesimpulan dan Rekomendasi

Sistem monitoring suhu dan kelembaban yang dikembangkan dengan integrasi Senso SHT20, TCP Server, InfluxDB, dan Grafana menunjukkan kinerja yang baik dalam melakukan akuisisi dan visualisasi data secara real-time. Implementasi sistem ini ditujukan untuk mendukung proses fermentasi kopi, yang sangat bergantung pada kestabilan parameter lingkungan, khususnya suhu dan

kelembaban. Berdasarkan hasil uji coba, nilai suhu yang terpantau (28,2 °C) telah berada dalam rentang optimal fermentasi kopi Arabika, yaitu antara 24–30 °C. Namun demikian, nilai kelembaban relatif yang terukur (76%) berada di atas batas atas rentang optimal, yaitu 50–70%. Ketidaksesuaian ini dapat dijelaskan oleh fakta bahwa pengukuran dilakukan di ruangan biasa, bukan di lingkungan fermentasi kopi.

Oleh karena itu, direkomendasikan agar sensor diletakkan secara langsung pada lokasi fermentasi guna memperoleh data yang lebih representatif dan relevan terhadap proses yang dimaksud. Selain itu, pengembangan lebih lanjut disarankan mencakup penambahan fitur notifikasi otomatis ketika parameter berada di luar rentang yang ditetapkan, serta integrasi sensor tambahan seperti pH dan CO_2 sebagai indikator fermentasi mikrobiologis. Secara keseluruhan, sistem ini memiliki potensi untuk diterapkan dalam praktik industri, khususnya dalam meningkatkan efisiensi dan konsistensi mutu produk melalui pemantauan kondisi lingkungan berbasis data.

IX. Daftar Pustaka

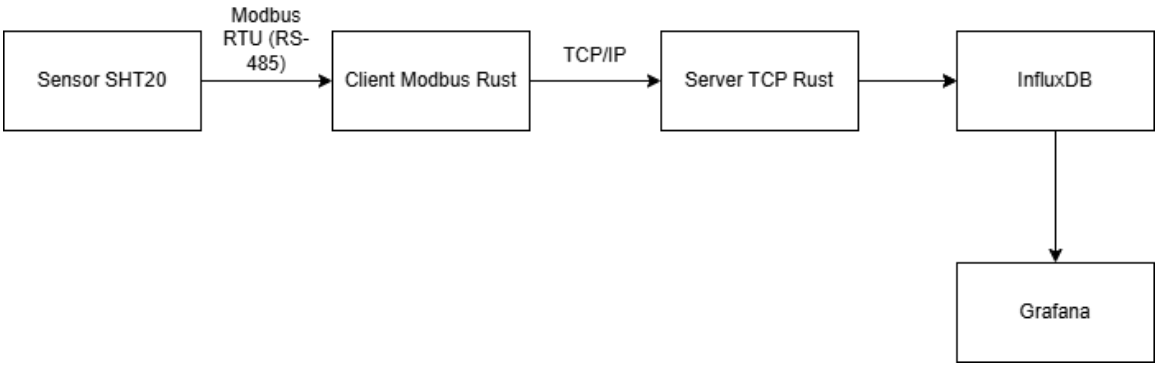
Nugraha, B. I. A., Susanto, R., & Erlinawati, M. (2024). Sistem Monitoring pH dan Suhu Pada Fermentasi Kopi Berbasis Internet of Things. *G-Tech*, *8*(3), 1816–1826. https://doi.org/10.33379/gtech.v8i3.4571

Sirappa, M. P., Heryanto, R., & Silitonga, Y. R. (2024). Standardisasi pengolahan biji kopi berkualitas. *Warta BSIP Perkebunan*, *2*(1), 18-25.

Noprianto, N., Wijayaningrum, V. N., & Wakhidah, R. (2023). Monitoring development board based on InfluxDB and Grafana. *Telematika: Jurnal Informatika dan Teknologi Informasi*, *20*(1), 81-90.

X. Lampiran

Lampiran 1. Diagram sistem lengkap



Lampiran 2. Link repository GitHub

https://github.com/mushthafali/RustProject-Blockchain.git