



LAPORAN INTERKONEKSI SISTEM INSTRUMENTASI - VI231418

Sistem Monitoring Temperatur dan Kelembapan pada Proses Fermentasi Kedelai untuk Optimasi Kualitas Tempe dan Oncom Hitam

NAMA KELOMPOK:

Muhammad 'Azmilfadhil S	(2042231003)
Zudan Rizky Aditya	(2042231007)
Bagus Wijaksono	(2042231039)

DOSEN PENGAMPU :

Dwi Oktavianto Wahyu
Nugroho, S.T., M.T.

Ahmad Radhy, S.Si., M.Si

**PROGRAM STUDI D4 TEKNOLOGI REKAYASA INSTRUMENTASI
DEPARTEMEN TEKNIK INSTRUMENTASI
FAKULTAS VOKASI
INSTITUT TEKNOLOGI SEPULUH
NOPEMBER 2025**

DAFTAR ISI

DAFTAR ISI	i
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Tujuan.....	4
1.4 Manfaat.....	4
BAB II TINJAUAN PUSTAKA.....	5
2.1 Sensor SHT20	5
2.2 Antar Muka RS-485	5
2.3 TCP.....	6
2.4 InfluxDB.....	6
2.5 Grafana	6
2.6 Rust.....	7
2.7 Blockchain.....	7
2.8 PyQT	7
BAB III METODOLOGI	8
BAB IV HASIL PENELITIAN	15
BAB V KESIMPULAN DAN SARAN	17
DAFTAR PUSTAKA	18

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kedelai merupakan komoditas pertanian penting di Indonesia yang memiliki peran krusial dalam ketahanan pangan dan ekonomi masyarakat. Sebagian besar kedelai diolah menjadi berbagai produk pangan hasil fermentasi, di antaranya tempe dan oncom hitam. Tempe sebagai makanan tradisional yang kaya protein, serat, serta vitamin B12, telah dikenal luas dan bahkan mendunia berkat nilai gizi dan fungsionalnya yang tinggi . Oncom hitam, meskipun kurang populer dibandingkan tempe, juga merupakan produk fermentasi yang memiliki nilai gizi dan fungsi penting, terutama dalam pemanfaatan limbah hasil pertanian seperti ampas tahu atau bungkil kacang tanah (Dwiatmaka *et al.*, 2022).

Proses fermentasi tempe dan oncom hitam sangat bergantung pada aktivitas mikroorganisme, yaitu kapang dari genus *Rhizopus sp*. Keberhasilan fermentasi ini sangat ditentukan oleh kondisi lingkungan yang optimal, khususnya temperatur dan kelembapan. Ketidaksesuaian nilai temperatur dan kelembapan dapat menghambat pertumbuhan kapang, memicu pertumbuhan mikroorganisme kontaminan yang tidak diinginkan, serta menghasilkan produk dengan kualitas (warna, aroma, tekstur, rasa) dan gizi yang kurang optimal (Safranti *et al.*, 2022).

Pada skala UMKM, proses fermentasi tempe dan oncom hitam seringkali masih dilakukan secara manual dengan mengandalkan pengalaman. Metode tradisional ini rentan terhadap ketidakpastian lingkungan, seperti perubahan suhu ruangan akibat cuaca atau aktivitas di sekitar. Akibatnya, kualitas produk seringkali bervariasi, bahkan dapat menyebabkan kegagalan produksi. Hal ini menjadi kendala serius dalam upaya standarisasi kualitas dan peningkatan kapasitas produksi (Yarlina *et al.*, 2023).

Pengembangan teknologi sistem monitoring menjadi solusi inovatif untuk mengatasi tantangan tersebut. Sistem monitoring temperatur dan kelembapan berbasis sensor memungkinkan pencatatan data secara *real-time* dan akurat selama proses fermentasi berlangsung. Data yang terkumpul dapat digunakan untuk memantau kondisi lingkungan dengan mengetahui kondisi optimal. Penerapan sistem ini tidak hanya meningkatkan konsistensi kualitas produk, tetapi juga mengoptimalkan efisiensi proses, mengurangi kerugian produksi, dan mendukung pengembangan produk fermentasi yang lebih higienis dan terstandarisasi. Oleh karena itu, penelitian mengenai sistem monitoring temperatur dan kelembapan pada proses fermentasi kedelai untuk optimasi kualitas tempe dan oncom hitam menjadi sangat relevan.

1.2 Rumusan Masalah

1. Bagaimana menciptakan kondisi suhu dan kelembapan yang optimal dan stabil selama proses fermentasi tempe dan oncom hitam agar mendukung pertumbuhan kapang *Rhizopus sp*?
2. Bagaimana merancang dan mengimplementasikan sistem inkubator fermentasi berbasis sensor SHT dan mikrokontroler yang mampu memantau serta mengatur suhu dan kelembapan secara otomatis dan real-time?

3. Sejauh mana efektivitas sistem monitoring dan kontrol suhu-kelembapan dalam meningkatkan konsistensi kualitas serta efisiensi waktu fermentasi pada produksi tempe dan oncom hitam?

1.3 Tujuan

1. Mengembangkan sistem monitoring dan kontrol otomatis terhadap parameter suhu dan kelembapan selama proses fermentasi tempe dan oncom hitam menggunakan sensor SHT serta mikrokontroler berbasis IoT.
2. Merancang inkubator fermentasi yang mampu menciptakan lingkungan stabil dan sesuai dengan kebutuhan pertumbuhan kapang *Rhizopus sp* guna menunjang keberhasilan fermentasi.
3. Meningkatkan efisiensi proses fermentasi melalui penciptaan kondisi lingkungan yang ideal, sehingga dapat mempercepat waktu fermentasi serta menghasilkan tempe dan oncom hitam dengan kualitas yang lebih konsisten dan optimal.

1.4 Manfaat

1. Bagi mahasiswa, meningkatkan pemahaman tentang prinsip-prinsip fermentasi kedelai, khususnya tempe dan oncom hitam, serta faktor-faktor lingkungan yang mempengaruhinya.
2. Bagi UMKM, membantu menstandarisasi dan meningkatkan konsistensi kualitas produk tempe dan oncom hitam, sehingga lebih kompetitif di pasar lokal maupun nasional.
3. Bagi masyarakat umum, Menjamin ketersediaan produk tempe dan oncom hitam berkualitas tinggi yang lebih konsisten, higienis, dan bernutrisi bagi konsumen.

BAB II

TINJAUAN PUSTAKA

2.1 Sensor SHT20

Sensor SHT20 adalah sensor digital generasi keempat buatan Sensirion yang dirancang untuk mengukur suhu dan kelembapan dengan akurasi tinggi serta konsumsi daya yang sangat rendah. Sensor ini menggunakan antarmuka komunikasi I2C dan memiliki rentang pengukuran suhu dari -40°C hingga $+125^{\circ}\text{C}$ serta kelembapan relatif dari 0 hingga 100% RH. Akurasi pengukuran suhu mencapai $\pm 0,2^{\circ}\text{C}$ dan kelembapan $\pm 1,8\%$ RH, menjadikannya ideal untuk aplikasi di bidang elektronik konsumen, otomasi rumah, dan perangkat IoT. Sensor SHT40 memiliki empat pin utama yang masing-masing memiliki fungsi spesifik dalam sistem pengukuran suhu dan kelembapan. Pin pertama adalah GND yang berfungsi sebagai ground atau referensi tegangan sistem. Pin kedua adalah VDD, yaitu pin catu daya yang menerima tegangan input antara 1,08 hingga 3,6 volt untuk menghidupkan sensor. Pin ketiga adalah SDA (Serial Data), yang digunakan sebagai jalur komunikasi data dua arah melalui protokol I2C. Pin keempat adalah SCL (Serial Clock), yang berfungsi sebagai jalur sinyal clock untuk sinkronisasi data I2C (Yu Yong *et al.*, 2024).



2.2 Antar Muka RS-485

Antarmuka RS-485 umumnya digunakan untuk membantu mikrokontroller dalam melakukan komunikasi data secara serial. RS-485 menggunakan dua kabel untuk mengirimkan sinyal data dan tidak memerlukan common ground. Sistem penyaluran data ini sering disebut dengan system differensial atau balanced. Salah satu IC yang dapat mengubah dari sinyal berbentuk TTL menjadi sinyal balanced RS-485 berfungsi sebagai pengirim adalah SN75176. IC SN75176 ini juga dapat berfungsi sebagai penerima atau pengubah sinyal balanced dari RS485 menjadi sinyal TTL. Karena IC SN75176 ini dapat berfungsi sebagai pengirim dan juga penerima, maka IC ini dapat disebut juga sebagai tranceicer RS-485. Pada RS-485 terdapat dua buah kaki yang berfungsi sebagai output atau input, kedua kaki ini dinamakan kaki A dan kaki B, dimana kedua kaki ini mempunyai sinyal yang saling berlawanan, artinya beda potensial antara kaki A dan kaki B dapat berubah sesuai dengan datanya. Pada penerimaan sinyal RS-485 bekerja dengan membaca perbedaan beda potensial antara sinyal A dan sinyal B. Jika sinyal A lebih besar minimal 200mV dari sinyal B, maka keluaran penerima akan berlogika tinggi (high), sebaliknya jika sinyal B lebih besar minimal 200mV dari sinyal A maka keluaran pada penerima adalah berlogika rendah. Antarmuka RS-485 merupakan salah satu standar komunikasi serial yang banyak digunakan dalam sistem industri dan otomasi karena keandalannya dalam menghubungkan banyak perangkat dengan jarak yang cukup jauh serta biaya implementasi yang relatif rendah (Tosin, 2023).



2.3 TCP

TCP *Transmission Control Protocol* merupakan protokol jaringan yang banyak digunakan dalam komunikasi data berbasis internet dan intranet. Dalam proyek ini, TCP Server digunakan sebagai antarmuka penerima data dari sensor yang dibaca oleh Modbus client. Rust, sebagai bahasa pemrograman sistem modern, menawarkan performa tinggi dan keamanan memori tanpa garbage collector, sehingga cocok digunakan dalam pengembangan sistem monitoring real-time. Implementasi TCP Server dalam Rust dilakukan dengan memanfaatkan pustaka jaringan seperti stdnet, TcpListener atau pustaka eksternal seperti tokio untuk keperluan asynchronous dan concurrent connection. Server ini akan menunggu koneksi dari klien (dalam hal ini program Modbus client), menerima data suhu dan kelembaban, kemudian meneruskannya ke database untuk disimpan. Dalam *ACM Transactions on Networking* mengeksplorasi tantangan dan solusi untuk komunikasi TCP di jaringan 5G dan masa depan yang sangat dinamis (Qiao *et al.*, 2024).

2.4 InfluxDB

InfluxDB adalah basis data time-series yang dirancang khusus untuk menangani data yang bersifat waktu (timestamp-based) seperti suhu, kelembaban, tekanan, dan data IoT lainnya. InfluxDB unggul dalam pencatatan dan kueri data berdasarkan waktu, serta memiliki struktur penyimpanan yang efisien dan query language yang sederhana (InfluxQL atau Flux). Dalam sistem ini, data yang diterima oleh TCP Server akan disimpan ke dalam InfluxDB dengan metadata waktu pencatatan. Penyimpanan ini penting untuk analisis tren, monitoring historis, serta integrasi dengan sistem visualisasi data real-time. Menunjukkan bahwa InfluxDB unggul dalam performa penyimpanan dan kecepatan query dibandingkan dengan TSDB lainnya seperti OpenTSDB dan Prometheus, terutama untuk workload dengan jumlah penulisan (writes) yang tinggi dan interval waktu yang rapat. InfluxDB juga mendukung arsitektur edge-cloud, sehingga cocok digunakan untuk sistem monitoring real-time berbasis mikrokontroler seperti ESP32 atau Raspberry Pi (Lehmann *et al.*, 2023).

2.5 Grafana

Grafana adalah platform open-source yang digunakan untuk visualisasi data dalam bentuk grafik interaktif, tabel, dan alert. Grafana mendukung berbagai sumber data, termasuk InfluxDB, dan dapat diakses melalui antarmuka web. Dalam proyek ini, Grafana digunakan untuk menampilkan data suhu dan kelembaban dari gudang fermentasi secara real-time. Dalam aplikasi praktis, Grafana banyak digunakan sebagai frontend visualisasi data sensor dan sistem monitoring. Misalnya, dalam sistem monitoring suhu dan kelembapan di ruang produksi, Grafana dikombinasikan dengan backend Node-Red dan database MySQL untuk menampilkan data secara real-time dalam bentuk grafik yang mudah dipahami, meningkatkan efektivitas pengawasan lingkungan produksi (Anam *et al.*, 2023).

2.6 Rust

Rust merupakan bahasa pemrograman sistem yang dirancang untuk memberikan keamanan memori tanpa harus mengorbankan performa. Bahasa ini dikembangkan oleh Mozilla dan menjadi populer karena kemampuannya dalam mencegah berbagai kesalahan umum seperti race condition, null pointer dereferencing, dan buffer overflow yang sering terjadi dalam bahasa seperti C dan C++. Rust menggunakan sistem kepemilikan memori (ownership system) yang unik untuk mengelola alokasi memori secara statis pada waktu kompilasi, tanpa memerlukan garbage collector. Hal ini menjadikannya ideal untuk aplikasi dengan kebutuhan efisiensi tinggi seperti sistem embedded, pemrograman perangkat keras, dan pengembangan web server performa tinggi. Menunjukkan bahwa Rust memberikan kinerja eksekusi yang setara bahkan melebihi C++ dalam beberapa benchmark, dengan keunggulan signifikan dalam hal deteksi error saat kompilasi. Implementasi dekompile ASN.1 antara Rust dan C++ menunjukkan bahwa Rust cenderung lebih efisien dalam penggunaan sumber daya jangka panjang dan lebih mudah dipelihara, meskipun performa awalnya sedikit lebih rendah (Emberg & Wadsten, 2023).

2.7 Blockchain

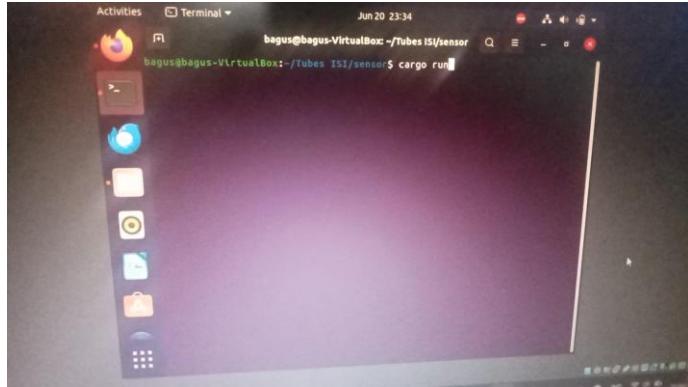
Blockchain merupakan teknologi terdesentralisasi yang menyimpan data dalam bentuk blok yang saling terhubung dan diamankan secara kriptografi. Teknologi ini pertama kali dikenal melalui implementasinya pada mata uang kripto seperti Bitcoin, namun kini telah berkembang luas di berbagai sektor seperti logistik, kesehatan, sistem voting, hingga Internet of Things (IoT). Salah satu keunggulan utama blockchain adalah kemampuannya untuk menciptakan sistem yang transparan, tidak dapat dimodifikasi, serta memiliki jejak audit yang jelas. Blockchain digunakan sebagai infrastruktur penyimpanan data terdistribusi untuk sistem IoT, dengan hasil yang menunjukkan peningkatan keamanan dan integritas data sensor. Mekanisme konsensus seperti Proof-of-Work (PoW) dan Proof-of-Stake (PoS) menjadi komponen penting dalam menjaga validitas data yang masuk ke jaringan blockchain. Selain itu, smart contract fitur kontrak otomatis dalam blockchain seperti Ethereum juga membuka peluang otomatisasi proses bisnis dan transaksi antar perangkat. Dalam beberapa tahun terakhir, penelitian tentang blockchain telah meledak, mencakup optimasi teknis, model konsensus baru, dan aplikasi transformatif di berbagai industri. Pengembangan model konsensus Proof-of-Stake (PoS) yang lebih efisien energi untuk blockchain dan Mengatasi beberapa keterbatasan utama blockchain generasi pertama (Liu & Sun, 2024).

2.8 Web3

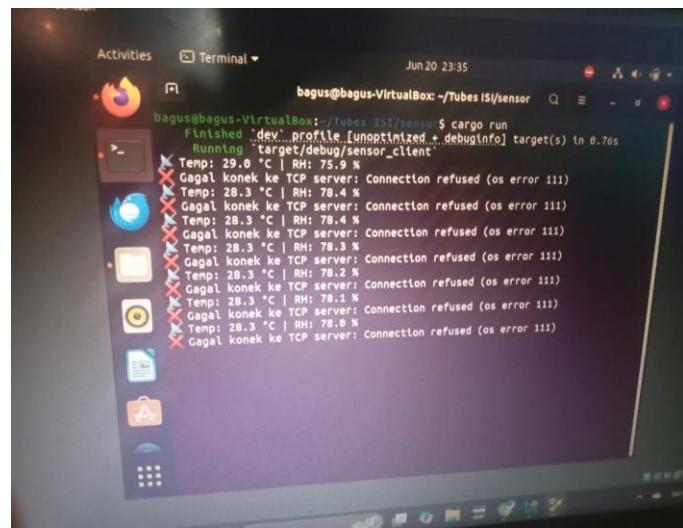
Web3 adalah evolusi yang diusulkan dari World Wide Web, didasarkan pada konsep desentralisasi, teknologi blockchain, dan ekonomi berbasis token. Berbeda dengan Web 1.0 (web statis) dan Web 2.0 (web interaktif yang didominasi oleh platform terpusat), Web3 bertujuan untuk memberikan kembali kontrol data dan kepemilikan kepada pengguna, bukan kepada perusahaan besar. Dalam sektor e-commerce, integrasi Web3 dan blockchain menghadirkan manfaat signifikan dalam keamanan transaksi, audit smart contract berkala, serta pelacakan pengaduan konsumen. Pengembangan ke depan diarahkan pada integrasi AI dalam smart contract untuk meningkatkan fleksibilitas transaksi, interoperabilitas lintas blockchain (cross-chain), dan model tata kelola berbasis Decentralized Autonomous Organization (DAO). Selain itu, ekspansi aplikasi Web3 juga mencakup sektor layanan digital dan produk digital seperti NFT dan e-learning (Sama et al., 2025).

BAB III METODOLOGI

1. Run folder sensor pada TUBES ISI

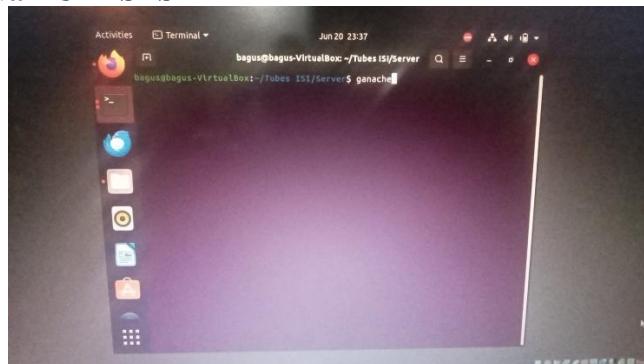


Dalam tampilan desktop ubuntu lalu pasang sensor SHT20 dan hubungkan ke USB yang sudah tertera pada settingan ubuntu, jika sudah pergi ke bagian file penyimpanan, cari folder TUGAS ISI lalu run dengan terminal lalu jalankan "cargo run"



Lalu akan menghasilkan informasi bahwa sensor sudah terdeteksi pada sistem yang sudah dijalankan dan menampilkan data sensor kondisi lingkungan saat itu.

2. Run folder server pada TUBES ISI



Langkah selanjutnya run pada folder server dengan “ganache” untuk menghasilkan informasi server akan dijalankan.

Lalu akan muncul informasi sistem blockchain dengan menginformasikan beberapa privat keys untuk diolah menjadi ip atau akun untuk digunakan menjalankan sistem.

```
Open main.rs Save Run Tab Width: 8 Ln 30 Col 98 IPS
19 j
20
21 #[tokio::main]
22 async fn main() -> anyhow::Result<()> {
23     // ... InfluxDB setup ...
24     let influx_url = "http://localhost:8086/api/v2/write?org=bagusfluxbuckets=sensor&precisions";
25     let influx_token =
26         "7cakTB5B8D1WjTM77qAqPlBwAtDx24R7GLTqA63J10JF4jjaqe2yw4lg0qCtg9U_3psva5gT8m";
27     let http_client = Client::new();
28
29     // ... Ethereum setup ...
30     let provider = Provider::try_from("http://localhost:8545")?;
31     let wallet = LocalWallet::parse(<LocalWallet>{});
32     .with_chain_id(1000);
33     let client = Arc::new(ArcMiddleware::new(provider, wallet));
34
35     // Baca dan parse ABI dan bytecode dengan benar
36     let abi_str = fs::read_to_string("build/sensorstorage.abi")?;
37     let bytecode_hex = fs::read_to_string("build/sensorstorage.bytecode")?;
38
39     let abi: Abi = serde_json::from_str(&(abi_str));
40     let bytecode = bytecode_hex.trim().parse(<bytes>())?;
41
42     let factory = ContractFactory::new(abi, bytecode, client.clone());
43
44 }
```

Pilih dan salin privat keys tadi untuk dimasukkan pada progam atau sistem pada folder server “main.rs” dengan ganti pada “let wallet: local wallet”

A screenshot of a Linux desktop environment, likely Ubuntu, featuring a purple gradient background. On the left is a vertical dock with various icons: a yellow bird (GNOME Activities), a terminal window icon, a file manager icon, a browser icon, a file icon, a folder icon, a target icon, a document icon, a briefcase icon, and a terminal window icon. The main window is a terminal titled "Terminal" with the status bar showing "Jun 20 23:41". The terminal window displays the command "bagus@bagus-VirtualBox: ~/Tubes ISI/Server \$ cargo run" in green text. The desktop has a standard top panel with icons for volume, brightness, and system status.

```
bagus@bagus-VirtualBox: ~/Tubes ISI/Server Jun 20 23:42
Compiling sensor_server_blockchain v0.1.0 (/home/bagus/Tubes ISI/Server)
Finished dev...profile [unoptimized + debuginfo] target(s) in 39.46s
Running "target/debug/sensor_server_blockchain"
Smart contract deployed at: 0x6b975d1692e19ba8f844fc7875a91bda5cd6d365
TCP Server listening on port 9000...
New connection from 127.0.0.1:54810
Received sensor data: SensorData { timestamp: "2025-06-20T16:42:31.566401917+08:00", sensor_id: "SHT20", location: "Ruang Fermentasi", process_stage: "Fermentasi Tempa dan Oncom Hitam", temperature_celsius: 28.5, humidity_percent: 77.1 }
```

Run folder server dengan “cargo run” lalu akan menghasilkan informasi mengenai *smart contract* pada sistem blockchain.

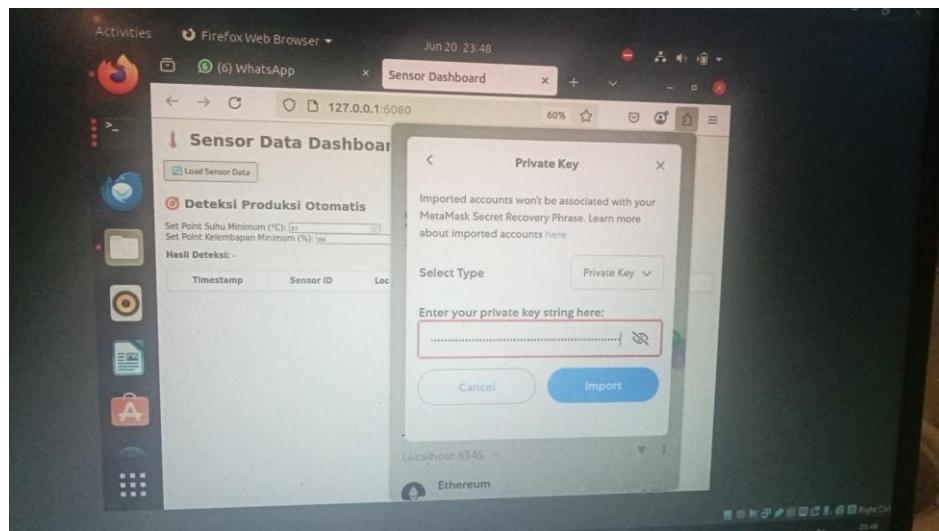
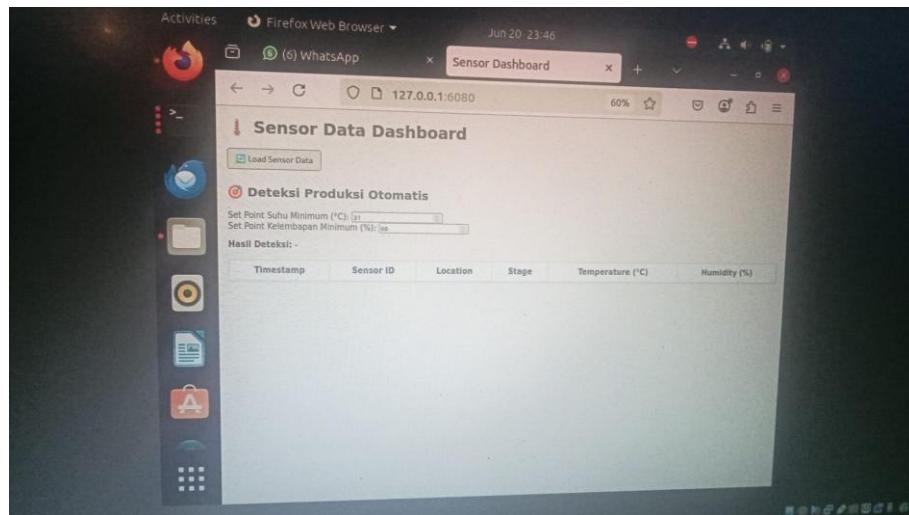
3. Run folder WEB3 pada TUBES ISI

```
bagus@bagus-VirtualBox: ~/Tubes ISI/Web3 Fitur Jun 20 23:43
python3 -m http.server 6080
Serving HTTP on 0.0.0.0 port 6080 (http://0.0.0.0:6080/)...
```

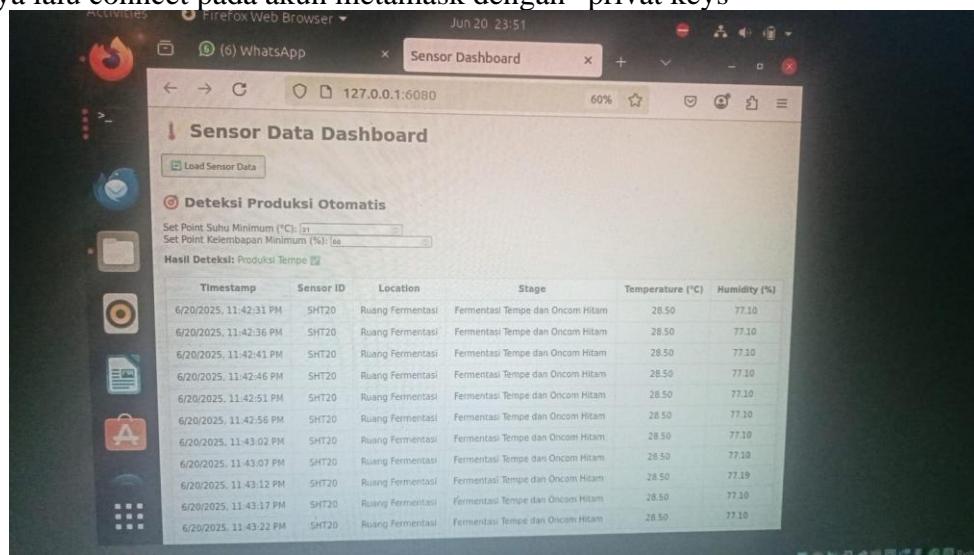
Salin *smart contract* yang sudah didapat pada langkah sebelumnya lalu masukkan pada program “script.js” pada folder web3 pada “const contractaddress”

```
bagus@bagus-VirtualBox: ~/Tubes ISI/Web3 Fitur Jun 20 23:45
python3 -m http.server 6080
Serving HTTP on 0.0.0.0 port 6080 (http://0.0.0.0:6080/)...
```

Run folder web3 dengan terminal lalu jalankan dengan perintah “python3 -m http.server 6080” untuk menuju ke localhost pada browser.

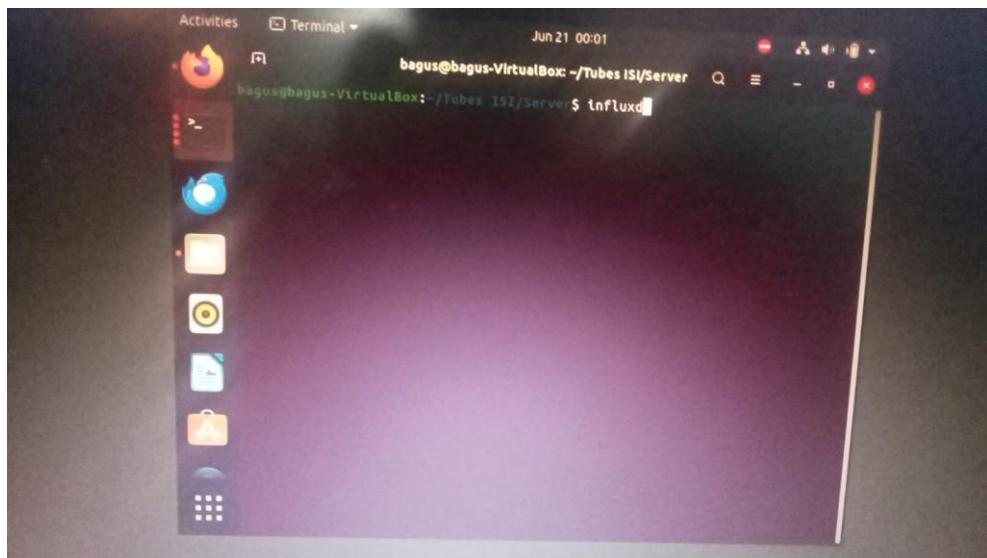


Lalu akan menghasilkan tampilan tersebut dengan juga menambahkan peraturan pada metamask yang akan menampilkan data dari sensor dengan menambahkan “privat keys” yang sama pada step sebelumnya lalu connect pada akun metamask dengan “privat keys”

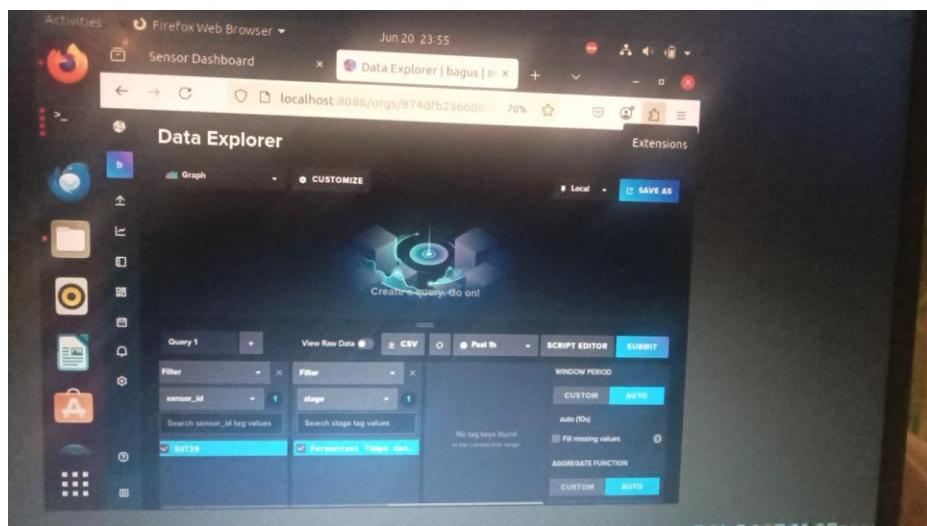


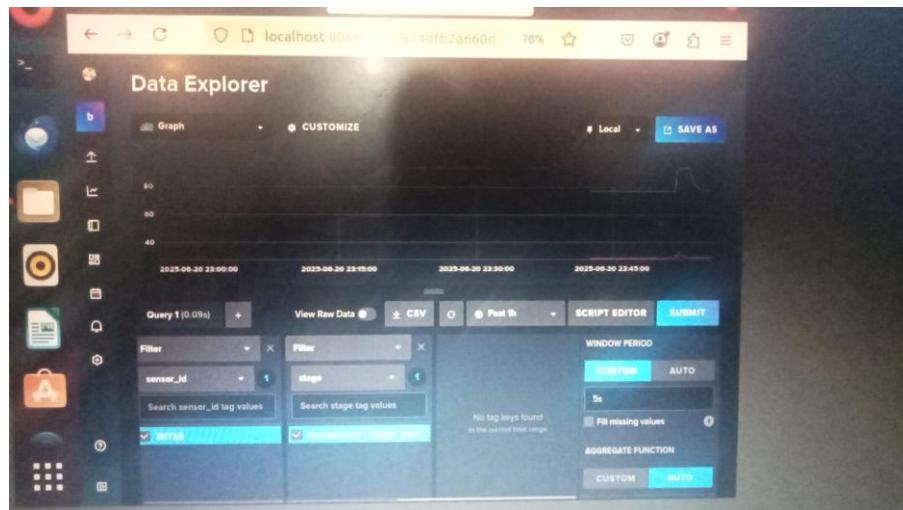
Akan muncul data secara real time dari sensor pada kondisi lingkungan tersebut

4. Influxdb



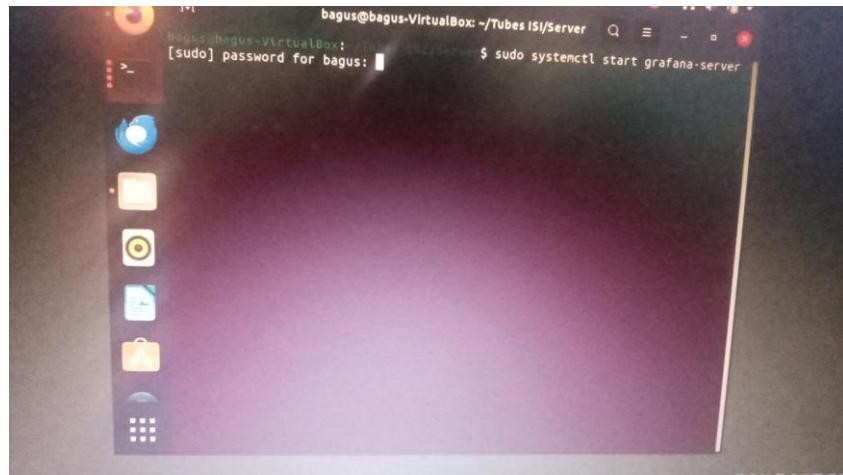
Run menggunakan terminal saat ingin membuka influxdb dengan “influxd” pada folder TUBES ISI



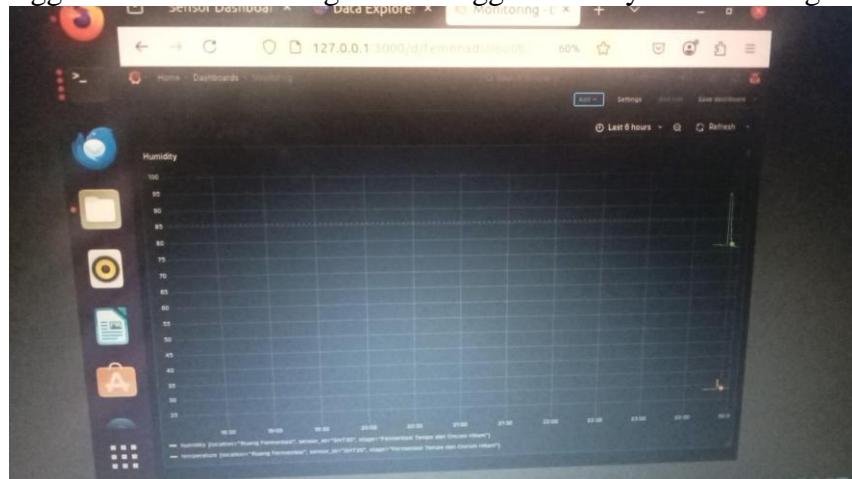


Lalu akan memunculkan tampilan influxdb dan akan menampilkan data dari sensor sesuai display influxdb

5. Grafana

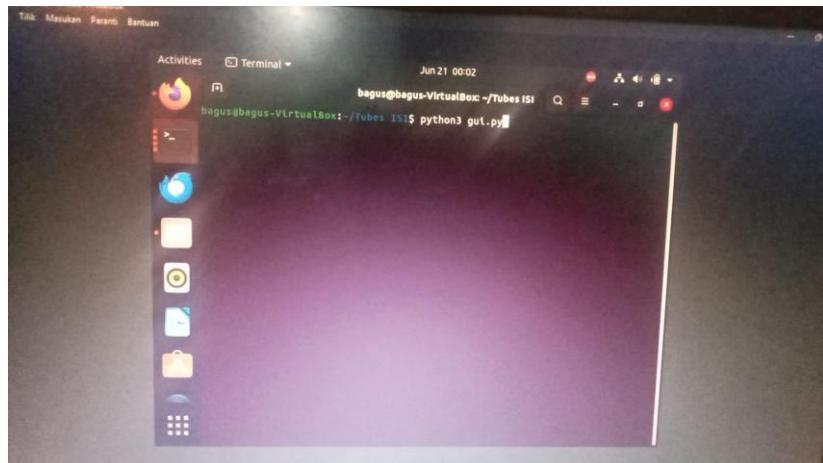


Run grafana menggunakan terminal dengan memnaggil “sudo systemctl start grafana-server”

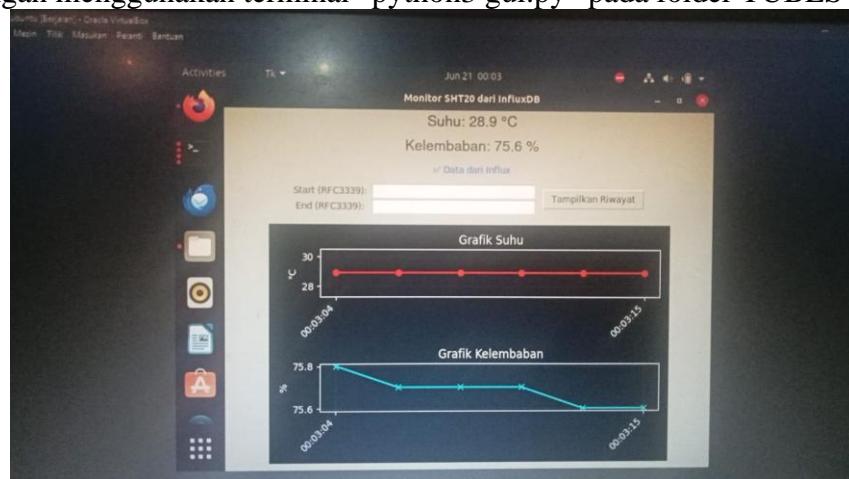


Lalu akan menampilkan data sesuai display grafana yang sesuai dengan sesnor pada kondisi lingkungan.

6. PyQt



Run PyQt dengan menggunakan terminal “python3 gui.py” pada folder TUBES ISI

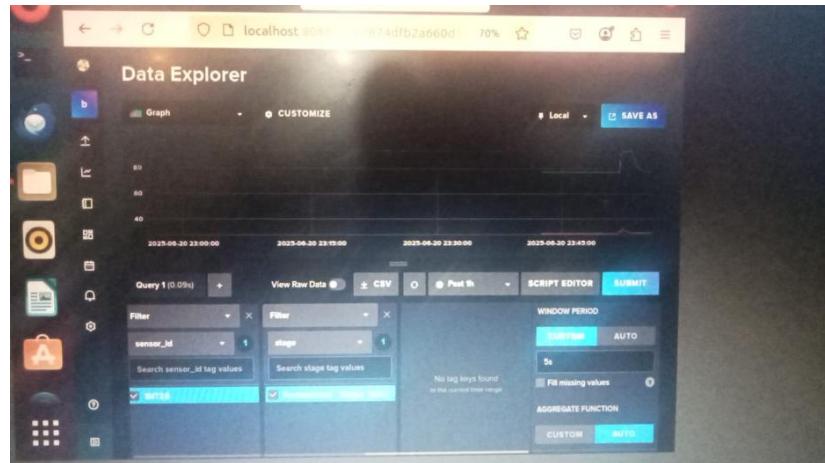


Lalu akan menghasilkan tampilan data sesuai display PyQt dari sensor pada kondisi lingkungan tersebut.

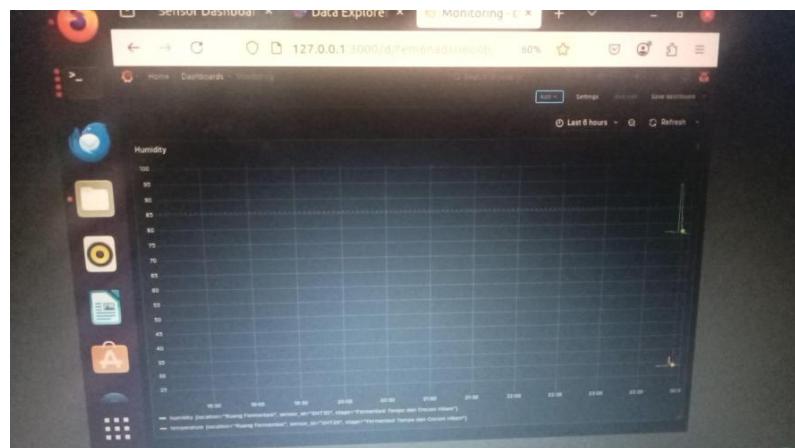
BAB IV

HASIL PENELITIAN

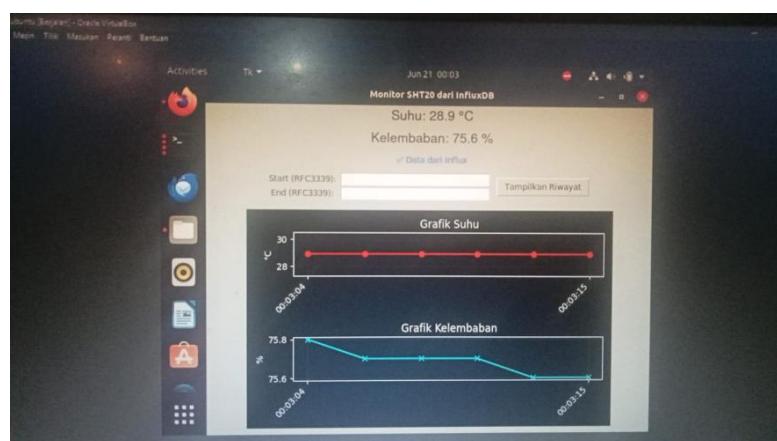
1. Berikut data yang sudah dihasilkan dan didapat pada sistem ini



Data dari influxdb



Data dari grafama



Data dari PyQt

Sensor Dashboard X Data Explore x Monitoring - +

127.0.0.1:6080 60%

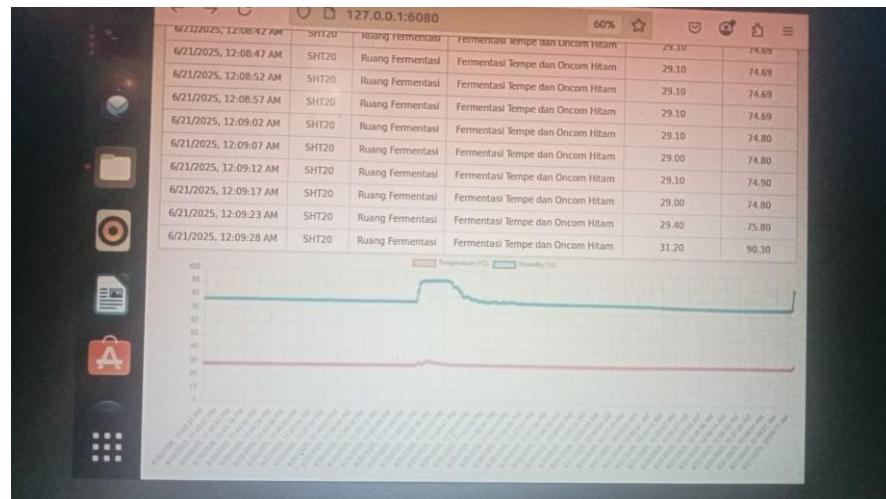
Sensor Data Dashboard

Deteksi Produksi Otomatis

Set Point Suhu Minimum (°C): Set Point Kelembapan Minimum (%):

Hasil Deteksi: Produksi Tempe

Timestamp	Sensor ID	Location	Stage	Temperature (°C)	Humidity (%)
6/20/2025, 11:42:31 PM	SHT20	Ruang Fermentasi	Fermentasi Tempe dan Oncom Hitam	28.50	77.10
6/20/2025, 11:42:36 PM	SHT20	Ruang Fermentasi	Fermentasi Tempe dan Oncom Hitam	28.50	77.10
6/20/2025, 11:42:41 PM	SHT20	Ruang Fermentasi	Fermentasi Tempe dan Oncom Hitam	28.50	77.10
6/20/2025, 11:42:46 PM	SHT20	Ruang Fermentasi	Fermentasi Tempe dan Oncom Hitam	28.50	77.10
6/20/2025, 11:42:51 PM	SHT20	Ruang Fermentasi	Fermentasi Tempe dan Oncom Hitam	28.50	77.10
6/20/2025, 11:42:56 PM	SHT20	Ruang Fermentasi	Fermentasi Tempe dan Oncom Hitam	28.50	77.10
6/20/2025, 11:43:02 PM	SHT20	Ruang Fermentasi	Fermentasi Tempe dan Oncom Hitam	28.50	77.10
6/20/2025, 11:43:07 PM	SHT20	Ruang Fermentasi	Fermentasi Tempe dan Oncom Hitam	28.50	77.10
6/20/2025, 11:43:12 PM	SHT20	Ruang Fermentasi	Fermentasi Tempe dan Oncom Hitam	28.50	77.10
6/20/2025, 11:43:17 PM	SHT20	Ruang Fermentasi	Fermentasi Tempe dan Oncom Hitam	28.50	77.10
6/20/2025, 11:43:22 PM	SHT20	Ruang Fermentasi	Fermentasi Tempe dan Oncom Hitam	28.50	77.10



Open script.js

```

main.rs
script.js

main.rs
script.js

83     responsive: true,
84     scales: {
85       y: { beginAtZero: true }
86     }
87   });
88 }
89
90 function deteksiProduksi(temp, hum) {
91   const setTemp = parseFloat(document.getElementById("setTemp").value);
92   const setHum = parseFloat(document.getElementById("setHum").value);
93   const hasil = document.getElementById("hasilDeteksi");
94
95   if (temp >= 31 && hum >= 80) {
96     hasil.textContent = "Produksi Tempe ✅";
97     hasil.style.color = "green";
98   } else if (temp >= 29 && hum >= 75) {
99     hasil.textContent = "Produksi Oncom Hitam ✅";
100    hasil.style.color = "blue";
101  } else {
102    hasil.textContent = "Data tidak cocok dengan produksi tempe atau oncom hitam";
103    hasil.style.color = "red";
104  }
105}
106
107

```

JavaScript Tab Width: 8 Ln 98, Col 33 INS

Data dari WEB3

BAB V

Kesimpulan dan saran

5.1 Kesimpulan

Berdasarkan hasil perancangan dan implementasi sistem monitoring suhu dan kelembapan pada proses fermentasi kedelai, dapat disimpulkan bahwa sistem ini mampu bekerja secara efektif dalam memantau kondisi lingkungan secara real-time dan akurat. Integrasi sensor SHT20 dengan mikrokontroler berbasis Rust memungkinkan akuisisi data yang cepat dan stabil. Data kemudian ditransmisikan melalui protokol TCP menuju server, disimpan dalam basis data time-series InfluxDB, dan divisualisasikan secara informatif melalui Grafana dan antarmuka PyQt. Selain itu, penggunaan teknologi Web3 dan smart contract berbasis blockchain menambah tingkat keamanan dan transparansi dalam penyimpanan data sensor. Sistem ini terbukti membantu menciptakan kondisi fermentasi yang lebih terkontrol, meningkatkan konsistensi kualitas produk tempe dan oncom hitam, serta meminimalkan risiko kegagalan akibat fluktuasi lingkungan. Oleh karena itu, pendekatan ini dapat menjadi solusi teknologi yang aplikatif untuk mendukung proses produksi pangan fermentasi di kalangan UMKM secara berkelanjutan.

5.2 Saran

Agar sistem ini dapat dioptimalkan lebih lanjut, disarankan untuk menambahkan fitur kontrol otomatis suhu dan kelembapan menggunakan aktuator seperti pemanas atau humidifier, sehingga tidak hanya melakukan monitoring, tetapi juga mampu melakukan penyesuaian kondisi secara mandiri. Selain itu, pengembangan sistem berbasis edge computing dapat dipertimbangkan untuk mengurangi latensi dan meningkatkan efisiensi pemrosesan data di lokasi fermentasi. Dari sisi antarmuka pengguna, perlu dilakukan evaluasi usability terhadap tampilan PyQt agar lebih ramah bagi operator non-teknis. Pengujian lebih lanjut dalam skala waktu yang lebih panjang juga penting dilakukan untuk menilai stabilitas sistem dalam kondisi lingkungan yang bervariasi. Terakhir, integrasi dengan sistem peringatan dini berbasis notifikasi (melalui SMS, Telegram, atau email) dapat menambah nilai fungsional dan membantu dalam pengambilan keputusan cepat saat terjadi anomali selama proses fermentasi.

DAFTAR PUSTAKA

- Anam, K., Rofi, D. N., & Meiyanti, R. (2023). Monitoring System for Temperature and Humidity Sensors in the Production Room Using Node-Red as the Backend and Grafana as the Frontend. *Journal of Systems Engineering and Information Technology (JOSEIT)*, 2(2), 59-76.
- Dwiatmaka, Y., Lukitaningsih, E., Yuniarti, N., & Wahyuono, S. (2022). Fermentation of soybean seeds using Rhizopus oligosporus for tempeh production and standardization based on isoflavones content. *Int. J. Appl. Pharm*, 14(6), 131-136.
- Emberg, C., & Wadsten, A. (2023). Attainable Safety and Long-Term Resource-Efficiency Using Rust: Evaluating the Viability of Rust in Terms of Development Effort and Performance for the Decompilation of ASN. 1 Messages.
- Lehmann, J., Schorz, S., Rache, A., Häußermann, T., Rädle, M., & Reichwald, J. (2023). Establishing reliable research data management by integrating measurement devices utilizing intelligent digital twins. *Sensors*, 23(1), 468.
- Liu, C., & Sun, Z. (2024). A Multi-Agent Reinforcement Learning-Based Task-Offloading Strategy in a Blockchain-Enabled Edge Computing Network. *Mathematics* (2227-7390), 12(14).
- Qiao, W., Zhang, Y., Dong, P., Du, X., Zhang, H., & Guizani, M. (2024). An AI-Enhanced Multipath TCP Scheduler for Open Radio Access Networks. *IEEE Transactions on Green Communications and Networking*.
- Safrianti, E., Sari, L. O., & Wulandari, F. (2022). IoT Applications in Fermented Tempe Production. *International Journal of Electrical, Energy and Power System Engineering*, 5(1), 1-5.
- Sama, H., Liang, S., & Khomali, C. J. (2025). Analisis Penerapan Teknologi Web3. 0 pada Pengembangan Game: Systematic Literature Review. *Jurnal Teknologi Dan Sistem Informasi Bisnis*, 7(1), 47-56.
- Tosin, T. (2021). Perancangan dan Implementasi Komunikasi RS-485 Menggunakan Protokol Modbus RTU dan Modbus TCP Pada Sistem Pick-By-Light. *Komputika: Jurnal Sistem Komputer*, 10(1), 85-91.
- Yarlina, S. T. P., Putri, V., Sakanti, K. K., Harlina, P. W., Kurniati, D., Wulandari, E., & Lani, M. N. Flour Substitution with Jack Bean Tempeh Flour (*Canavalia Ensiformis*) and Modified Cassava Flour (Mocaf): Physicochemical and Organoleptic Characteristics of High Protein Cookies. Available at SSRN 5074810.
- Yu, Y., Gola, M., Settimo, G., Buffoli, M., & Capolongo, S. (2024). Feasibility and Affordability of Low-Cost Air Sensors with Internet of Things for Indoor Air Quality Monitoring in Residential Buildings: Systematic Review on Sensor Information and Residential Applications, with Experience-Based Discussions. *Atmosphere*, 15(10), 1170.

Lampiran

Code Program

```
1.      use tokio_modbus::{client::rtu, prelude::*};
2. use tokio_serial::{SerialPortBuilderExt, Parity, StopBits, DataBits};
3. use tokio::net::TcpStream;
4. use tokio::io::AsyncWriteExt;
5. use serde::Serialize;
6. use chrono::Utc;
7. use std::error::Error;
8. use tokio::time::{sleep, Duration};
9.
10. #[derive(Serialize)]
11. struct SensorData {
12.     timestamp: String,
13.     sensor_id: String,
14.     location: String,
15.     process_stage: String,
16.     temperature_celsius: f32,
17.     humidity_percent: f32,
18. }
19.
20. async fn read_sensor(slave: u8) -> Result<vec<u16>, Box<dyn error=""/>> {
21.     let builder = tokio_serial::new("/dev/ttyUSB0", 9600)
22.         .parity(Parity::None)
23.         .stop_bits(StopBits::One)
24.         .data_bits(DataBits::Eight)
25.         .timeout(std::time::Duration::from_secs(1));
26.
27.     let port = builder.open_native_async()?;
28.     let mut ctx = rtu::connect_slave(port, Slave(slave)).await?;
29.     let response = ctx.read_input_registers(1, 2).await?;
30.
31.     Ok(response)
32. }
33.
34. #[tokio::main]
35. async fn main() -> Result<(), Box<dyn error=""/>> {
36.     loop {
37.         match read_sensor(1).await {
38.             Ok(response) if response.len() == 2 => {
39.                 let temp = response[0] as f32 / 10.0;
40.                 let rh = response[1] as f32 / 10.0;
41.
42.                 println!("` Temp: {:.1} °C | RH: {:.1} %", temp, rh);
43.
44.                 let data = SensorData {
45.                     timestamp: Utc::now().to_rfc3339(),
46.                     sensor_id: "SHT20".into(),
```

```

47.             location: "Ruang Fermentasi".into(),
48.             process_stage: "Fermentasi Tempe dan Oncom Hitam".into(),
49.             temperature_celsius: temp,
50.             humidity_percent: rh,
51.         );
52.
53.         let json = serde_json::to_string(&data)?;
54.
55.         match TcpStream::connect("127.0.0.1:9000").await {
56.             Ok(mut stream) => {
57.                 stream.write_all(json.as_bytes()).await?;
58.                 stream.write_all(b"\n").await?;
59.                 println!("✅ Data dikirim ke TCP server");
60.             },
61.             Err(e) => {
62.                 println!("❌ Gagal koneksi ke TCP server: {}", e);
63.             }
64.         }
65.     },
66.     Ok(other) => {
67.         println!("⚠ Data tidak lengkap: {:?}", other);
68.     },
69.     Err(e) => {
70.         println!("❌ Gagal baca sensor: {}", e);
71.     }
72. }
73.
74.         sleep(Duration::from_secs(5)).await;
75.     }
76. }</dyn></vec<u16>
```

```

1. use tokio::net::TcpListener;
2. use tokio::io::{AsyncBufReadExt, BufReader};
3. use serde::Deserialize;
4. use reqwest::Client;
5.
6. use ethers::prelude::*;
7. use ethers::abi::Abi;
8. use std::fs, sync::Arc;
9. use chrono::DateTime;
10.
11. #[derive(Deserialize, Debug)]
12. struct SensorData {
13.     timestamp: String,
14.     sensor_id: String,
15.     location: String,
16.     process_stage: String,
17.     temperature_celsius: f32,
18.     humidity_percent: f32,
19. }
20.
```

```
21. #[tokio::main]
22.     async fn main() -> anyhow::Result<()> {
23.         // --- InfluxDB setup ---
24.         let influx_url =
25.             "http://localhost:8086/api/v2/write?org=bagus&bucket=sensor&precision=s";
26.         let influx_token =
27.             "7cxktB5BI01WjTM77vAqPL8WaTdXz4X7GLTqA63JI6JF4jjaqe2yw4LgDqCTg9U_JpsvaSgT8mjeSVLDLMv59Q=="
28. ;
29.         let http_client = Client::new();
30.         // --- Ethereum setup ---
31.         let provider = Provider::try_from("http://localhost:8545")?;
32.         let wallet: LocalWallet =
33.             "0xf90efc0585aca78cae0610d22d8c001b8184505886b342bce47ae17b3ac221d8"
34.                 .parse::<localwallet>()?;
35.                 .with_chain_id(1337u64);
36.         let client = Arc::new(SignerMiddleware::new(provider, wallet));
37.         // Baca dan parse ABI dan bytecode dengan benar
38.         let abi_str = fs::read_to_string("build/SensorStorage.abi")?;
39.         let bytecode_str = fs::read_to_string("build/SensorStorage.bin")?;
40.         let abi: Abi = serde_json::from_str(&abi_str)?;
41.         let bytecode = bytecode_str.trim().parse::<bytes>()?;
42.         let factory = ContractFactory::new(abi, bytecode, client.clone());
43.         let contract = factory.deploy()?.send().await?;
44.         println!("☑ Smart contract deployed at: {:?}", contract.address());
45.         // --- TCP Server ---
46.         let listener = TcpListener::bind("0.0.0.0:9000").await?;
47.         println!("🕒 TCP Server listening on port 9000...");
```

50.

```
51.         loop {
52.             let (socket, addr) = listener.accept().await?;
53.             println!("🔌 New connection from {}", addr);
54.             let influx_url = influx_url.to_string();
55.             let influx_token = influx_token.to_string();
56.             let http_client = http_client.clone();
57.             let contract = contract.clone();
58.             tokio::spawn(async move {
59.                 let reader = BufReader::new(socket);
60.                 let mut lines = reader.lines();
61.                 while let Ok(Some(line)) = lines.next_line().await {
62.                     match serde_json::from_str::<sensordata>(&line) {
63.                         Ok(data) => {
64.                             println!("⌚ Received sensor data: {:?}", data);
```



```

116.
117.             match tx {
118.                 Ok(pending_tx) => {
119.                     println!("📝 Ethereum: tx sent: {:?}", pending_tx);
120.                 }
121.                 Err(e) => {
122.                     println!("✖ Ethereum tx error: {:?}", e);
123.                 }
124.             }
125.         }
126.     Err(e) => println!("✖ Invalid JSON received: {}", e),
127.     }
128.   }
129. );
130. }
131. }</sensordata></bytes></localwallet></http>

1. const contractAddress = "0x25b23ee3f98f8ee75150a6b76085c825e9c58909"; // Ganti dengan
   alamat kontrakmu
2. const abiPath = "abi/SensorStorage.abi"; // Pastikan file ABI ini tersedia
3.
4. let chart;
5.
6. async function loadSensorData() {
7.   const abiRes = await fetch(abiPath);
8.   const abi = await abiRes.json();
9.
10.   const provider = new ethers.BrowserProvider(window.ethereum ||
    "http://localhost:8545");
11.   await provider.send("eth_requestAccounts", []);
12.   const signer = await provider.getSigner();
13.   const contract = new ethers.Contract(contractAddress, abi, signer);
14.
15.   const filter = contract.filters.DataStored();
16.   const events = await contract.queryFilter(filter, 0, "latest");
17.
18.   const tableBody = document.querySelector("#sensorTable tbody");
19.   tableBody.innerHTML = "";
20.
21.   const labels = [];
22.   const temps = [];
23.   const hums = [];
24.
25.   events.forEach((e) => {
26.     const data = e.args;
27.     const timeStr = new Date(Number(data.timestamp) * 1000).toLocaleString();
28.     const temp = Number(data.temperature) / 100;
29.     const hum = Number(data.humidity) / 100;
30.
31.     tableBody.innerHTML += `
32.

```

```
33.         ${timeStr}
34.         ${data.sensorId}
35.         ${data.location}
36.         ${data.stage}
37.         ${temp.toFixed(2)}
38.         ${hum.toFixed(2)}
39.
40.     `;
41.
42.     labels.push(timeStr);
43.     temps.push(temp);
44.     hums.push(hum);
45. });
46.
47. renderChart(labels, temps, hums);
48.
49. // Deteksi berdasarkan data terakhir
50. if (events.length > 0) {
51.     const latest = events[events.length - 1];
52.     const latestTemp = Number(latest.args.temperature) / 100;
53.     const latestHum = Number(latest.args.humidity) / 100;
54.     deteksiProduksi(latestTemp, latestHum);
55. }
56. }
57.
58. function renderChart(labels, temps, hums) {
59.     const ctx = document.getElementById('chart').getContext('2d');
60.
61.     if (chart) chart.destroy();
62.
63.     chart = new Chart(ctx, {
64.         type: 'line',
65.         data: {
66.             labels,
67.             datasets: [
68.                 {
69.                     label: "Temperature (°C)",
70.                     data: temps,
71.                     borderColor: 'rgba(255, 99, 132, 1)',
72.                     fill: false
73.                 },
74.                 {
75.                     label: "Humidity (%)",
76.                     data: hums,
77.                     borderColor: 'rgba(54, 162, 235, 1)',
78.                     fill: false
79.                 }
80.             ]
81.         },
82.         options: {
83.             responsive: true,
```

```

84.         scales: {
85.             y: { beginAtZero: true }
86.         }
87.     );
88. );
89. }
90.
91. function deteksiProduksi(temp, hum) {
92.     const setTemp = parseFloat(document.getElementById("setTemp").value);
93.     const setHum = parseFloat(document.getElementById("setHum").value);
94.     const hasil = document.getElementById("hasilDeteksi");
95.
96.     if (temp >= 25 && hum >= 60) {
97.         hasil.textContent = "Produksi Tempe ✅";
98.         hasil.style.color = "green";
99.     } else if (temp >= 24 && hum >= 48) {
100.        hasil.textContent = "Produksi Oncom Hitam ✅";
101.        hasil.style.color = "blue";
102.    } else {
103.        hasil.textContent = "Data tidak cocok dengan produksi tempe atau oncom hitam.";
104.        hasil.style.color = "red";
105.    }
106. }

```

```

1. <meta charset="UTF-8">
2. <title>Sensor Dashboard</title>
3. <script src="https://cdn.jsdelivr.net/npm/ethers/dist/ethers.umd.min.js"></script>
4. <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
5. <link rel="stylesheet" href="style.css">
6.
7.
8. <h1> Sensor Data Dashboard</h1>
9.
10. <button onclick="loadSensorData()"> Load Sensor Data</button>
11.
12. <h2> Deteksi Produksi Otomatis</h2>
13. <div>
14.     <label for="setTemp">Set Point Suhu Minimum (°C):</label>
15.     <input type="number" id="setTemp" value="31" step="0.1"><br>
16.
17.     <label for="setHum">Set Point Kelembapan Minimum (%):</label>
18.     <input type="number" id="setHum" value="66" step="0.1"><br>
19.
20.     <p><strong>Hasil Deteksi:</strong> <span id="hasilDeteksi">-</span></p>
21. </div>
22.
23. <table id="sensorTable">
24.     <thead>
25.         <tr>
26.             <th>Timestamp</th>
27.             <th>Sensor ID</th>

```

```
28.          <th>Location</th>
29.          <th>Stage</th>
30.          <th>Temperature (°C)</th>
31.          <th>Humidity (%)</th>
32.      </tr>
33.  </thead>
34.  <tbody></tbody>
35. </table>
36.
37.  <canvas id="chart" height="100"></canvas>
38.
39.  <script src="script.js"></script>

1. import tkinter as tk
2. from tkinter import ttk
3. import requests
4. import threading
5. import time
6. import csv
7. from io import StringIO
8. from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
9. from matplotlib.figure import Figure
10.    from collections import deque
11.
12.    # Konfigurasi InfluxDB
13.    INFLUX_QUERY_URL = "http://localhost:8086/api/v2/query"
14.    ORG = "bagus"
15.    BUCKET = "sensor"
16.    TOKEN =
17.        "7cxktB5BI0lWjTM77vAqPL8WaTdXz4X7GLTqA63JI6JF4jjqe2yw4LgDqCTg9U_JpsvaSgT8mjeSVLDLMv59Q=="
18.
19.    # Riwayat data
20.    history_length = 50
21.    temp_history = deque(maxlen=history_length)
22.    rh_history = deque(maxlen=history_length)
23.    time_history = deque(maxlen=history_length)
24.
25.    def get_latest_data():
26.        flux_query = f'''
27.            from(bucket: "{BUCKET}")
28.                |> range(start: -1m)
29.                |> filter(fn: (r) => r._measurement == "monitoring")
30.                |> filter(fn: (r) => r._field == "temperature" or r._field == "humidity")
31.                |> last()
32.        '''
33.
34.        headers = {
35.            "Authorization": f"Token {TOKEN}",
36.            "Content-Type": "application/vnd.flux",
37.            "Accept": "application/csv"
38.        }
```

```

38.
39.     try:
40.         response = requests.post(
41.             INFLUX_QUERY_URL,
42.             params={"org": ORG},
43.             headers=headers,
44.             data=flux_query
45.     )
46.
47.     reader = csv.DictReader(StringIO(response.text))
48.     data = {}
49.     for row in reader:
50.         try:
51.             field = row["_field"]
52.             value = float(row["_value"])
53.             data[field] = value
54.         except:
55.             continue
56.
57.         if "temperature" in data and "humidity" in data:
58.             return data["temperature"], data["humidity"]
59.     return None
60. except Exception as e:
61.     print("X Exception query Influx:", e)
62.     return None
63.
64. def get_data_range(start_time, end_time):
65.     flux_query = f'''
66.         from(bucket: "{BUCKET}")
67.             |> range(start: {start_time}, stop: {end_time})
68.             |> filter(fn: (r) => r._measurement == "monitoring")
69.             |> filter(fn: (r) => r._field == "temperature" or r._field == "humidity")
70.             ...
71.
72.         headers = {
73.             "Authorization": f"Token {TOKEN}",
74.             "Content-Type": "application/vnd.flux",
75.             "Accept": "application/csv"
76.         }
77.
78.     try:
79.         response = requests.post(
80.             INFLUX_QUERY_URL,
81.             params={"org": ORG},
82.             headers=headers,
83.             data=flux_query
84.     )
85.
86.         reader = csv.DictReader(StringIO(response.text))
87.         temp_map = {}
88.         rh_map = {}

```

```

89.
90.         for row in reader:
91.             try:
92.                 t = row["_time"]
93.                 field = row["_field"]
94.                 value = float(row["_value"])
95.                 if field == "temperature":
96.                     temp_map[t] = value
97.                 elif field == "humidity":
98.                     rh_map[t] = value
99.             except:
100.                 continue
101.
102.             sorted_keys = sorted(set(temp_map.keys()) & set(rh_map.keys()))
103.             temps = [temp_map[t] for t in sorted_keys]
104.             rhs = [rh_map[t] for t in sorted_keys]
105.             times = [t[11:19] for t in sorted_keys] # jam:menit:detik
106.
107.             return temps, rhs, times
108.     except Exception as e:
109.         print("X Exception query Influx:", e)
110.         return [], [], []
111.
112.     def update_data():
113.         while True:
114.             result = get_latest_data()
115.             current_time = time.strftime('%H:%M:%S')
116.
117.             if result:
118.                 temp, rh = result
119.                 label_temp.config(text=f"Suhu: {temp:.1f} °C")
120.                 label_rh.config(text=f"Kelembaban: {rh:.1f} %")
121.                 status_label.config(text="✓ Data dari Influx")
122.
123.                 temp_history.append(temp)
124.                 rh_history.append(rh)
125.                 time_history.append(current_time)
126.
127.                 plot_graph()
128.             else:
129.                 label_temp.config(text="Suhu: ---")
130.                 label_rh.config(text="Kelembaban: ---")
131.                 status_label.config(text="X Gagal ambil data")
132.
133.                 time.sleep(2)
134.
135.     def plot_graph():
136.         ax1.clear()
137.         ax2.clear()
138.

```

```
139.     # Set background ke hitam
140.     fig.patch.set_facecolor('black')
141.     ax1.set_facecolor('black')
142.     ax2.set_facecolor('black')
143.
144.     x = list(range(len(time_history)))
145.     times = list(time_history)
146.
147.     # Plot data dengan garis dan warna yang kontras
148.     ax1.plot(x, list(temp_history), label='Suhu (°C)', color='red', marker='o',
149.     linestyle='--')
150.     ax2.plot(x, list(rh_history), label='Kelembaban (%)', color='cyan', marker='x',
151.     linestyle='--')
152.     ax1.set_title("Grafik Suhu", color='white')
153.     ax2.set_title("Grafik Kelembaban", color='white')
154.     ax1.set_ylabel("°C", color='white')
155.     ax2.set_ylabel("%", color='white')
156.     # Tampilkan hanya label waktu setiap 5 data
157.     interval = 5
158.     tick_positions = x[::interval]
159.     tick_labels = times[::interval]
160.
161.     ax1.set_xticks(tick_positions)
162.     ax2.set_xticks(tick_positions)
163.     ax1.set_xticklabels(tick_labels, rotation=45, ha="right", color='white')
164.     ax2.set_xticklabels(tick_labels, rotation=45, ha="right", color='white')
165.
166.     for ax in [ax1, ax2]:
167.         ax.tick_params(axis='y', colors='white')
168.         ax.tick_params(axis='x', colors='white')
169.         ax.grid(True, linestyle='--', alpha=0.3, color='gray')
170.         for spine in ax.spines.values():
171.             spine.set_color('white')
172.
173.     fig.tight_layout()
174.     canvas.draw()
175.
176. def show_history():
177.     start = entry_start.get()
178.     end = entry_end.get()
179.     temps, rhs, times = get_data_range(start, end)
180.
181.     if temps and rhs:
182.         temp_history.clear()
183.         rh_history.clear()
184.         time_history.clear()
185.
186.         temp_history.extend(temps)
187.         rh_history.extend(rhs)
```

```
188.         time_history.extend(times)
189.
190.         label_temp.config(text="(Hist) Suhu: -- °C")
191.         label_rh.config(text="(Hist) RH: -- %")
192.         status_label.config(text="☒ Menampilkan data historis")
193.         plot_graph()
194.     else:
195.         status_label.config(text="☒ Tidak ada data historis")
196.
197.     # GUI Setup
198.     root = tk.Tk()
199.     root.title("Monitor SHT20 dari InfluxDB")
200.     root.geometry("800x650")
201.
202.     label_temp = tk.Label(root, text="Suhu: -- °C", font=("Helvetica", 16))
203.     label_temp.pack(pady=5)
204.
205.     label_rh = tk.Label(root, text="Kelembaban: -- %", font=("Helvetica", 16))
206.     label_rh.pack(pady=5)
207.
208.     status_label = tk.Label(root, text="Status: ---", fg="blue")
209.     status_label.pack(pady=5)
210.
211.     frame_input = tk.Frame(root)
212.     frame_input.pack(pady=5)
213.
214.     tk.Label(frame_input, text="Start (RFC3339):").grid(row=0, column=0, padx=5)
215.     entry_start = tk.Entry(frame_input, width=30)
216.     entry_start.grid(row=0, column=1)
217.
218.     tk.Label(frame_input, text="End (RFC3339):").grid(row=1, column=0, padx=5)
219.     entry_end = tk.Entry(frame_input, width=30)
220.     entry_end.grid(row=1, column=1)
221.
222.     btn_show = tk.Button(frame_input, text="Tampilkan Riwayat", command=show_history)
223.     btn_show.grid(row=0, column=2, rowspan=2, padx=10)
224.
225.     fig = Figure(figsize=(6, 4), dpi=100)
226.     ax1 = fig.add_subplot(211)
227.     ax2 = fig.add_subplot(212)
228.
229.     canvas = FigureCanvasTkAgg(fig, master=root)
230.     canvas.get_tk_widget().pack(pady=10)
231.
232.     # Mulai update realtime
233.     threading.Thread(target=update_data, daemon=True).start()
234.
235.     root.mainloop()
```