Logo

Description automatically generated

**INTERKONEKSI SISTEM INSTRUMENTASI – VI231418**

**SISTEM MONITORING SUHU DAN KELEMBAPAN BERBASIS IOT PADA AREA RAWAN KEBAKARAN DI INDUSTRI MINYAK DAN GAS SEBAGAI LANGKAH MENDUKUNG K3**

**RIZAL KHOIRUL ATOK**

NRP 2042231013

**MUHAMMAD EMIR HAKIM ZAUHARI**

NRP 2042231069

**DAFFA NAUFAL WAHYUAJI**

NRP 2042231081

Dosen Pengampu

**Ahmad Radhy, S.Si., M.Si**

**NPP. 2022198911049**

**Program Studi Rekayasa Teknologi Instrumentasi**

**Departemen Teknik Instrumentasi**

**Fakultas Vokasi**

**Institut Teknologi Sepuluh Nopember Surabaya**

**2025**

ABSTRAK

**Sistem Monitoring Suhu dan Kelembapan Berbasis IoT pada Area Rawan Kebakaran di Industri Migas sebagai Langkah Mendukung K3**

|  |  |  |
| --- | --- | --- |
| **Nama mahasiswa / NRP** | **:** | **Rizal Khirul Atok (2042231013)**  **M Emi Hakim Zauhari (20422310369)**  **Daffa Naufal Wahyuaji (2042231081)** |
| **Program studi** | **:** | **Teknologi Rekayasa Instrumentasi** |
| **Dosen Pembimbing / NIP** | **:** | **Ahmad Radhy, S.Si., M.Si (2022198911049)** |

**Abstrak**

Industri minyak dan gas merupakan sektor dengan risiko tinggi terhadap kecelakaan kerja, khususnya kebakaran akibat fluktuasi suhu dan kelembapan yang tidak terkendali. Penelitian ini merancang dan mengimplementasikan sistem monitoring suhu dan kelembapan berbasis Internet of Things (IoT) yang dilengkapi dengan teknologi blockchain untuk mendukung penerapan Keselamatan dan Kesehatan Kerja (K3) secara optimal. Sistem menggunakan sensor SHT20 yang dihubungkan melalui protokol komunikasi Modbus RTU ke TCP Server berbasis Rust. Data hasil pengukuran dikirim secara real-time ke basis data time-series InfluxDB dan divisualisasikan melalui Grafana serta antarmuka GUI menggunakan PyQt5. Untuk menjamin integritas dan transparansi data, sistem juga mencatat informasi ke dalam blockchain Ethereum melalui smart contract yang dapat diakses menggunakan platform Web3 DApp. Hasil pengujian menunjukkan bahwa sistem mampu memantau kondisi lingkungan secara akurat, memberikan notifikasi dini melalui Telegram, dan menjaga integritas data melalui pencatatan blockchain. Dengan demikian, sistem ini dapat menjadi solusi monitoring lingkungan yang handal, real-time, dan aman untuk diterapkan di area rawan kebakaran pada industri migas

**Kata kunci: *IoT,* suhu dan kelembapan*, blockchain, smart contract, K3, Web3, monitoring industri.***

DAFTAR ISI

[ABSTRAK i](#_Toc201341833)

[DAFTAR ISI ii](#_Toc201341835)

[DAFTAR GAMBAR iv](#_Toc201341836)

[DAFTAR TABEL v](#_Toc201341837)

[BAB 1 PENDAHULUAN 6](#_Toc201341838)

[1.1 Latar Belakang 6](#_Toc201341839)

[1.2 Rumusan Masalah 6](#_Toc201341840)

[1.3 Batasan Masalah 7](#_Toc201341841)

[1.4 Tujuan 7](#_Toc201341842)

[BAB 2 TINJAUAN PUSTAKA 8](#_Toc201341844)

[2.1 Hasil Penelitian Terdahulu 8](#_Toc201341845)

[2.2 Komponen dan Protocol Program 8](#_Toc201341846)

[2.2.1 Sensor MD02/SHT 40 8](#_Toc201341847)

[2.2.2 Antar Muka RS-48 9](#_Toc201341848)

[2.2.3 TCP (Transmission Control Protocol) 9](#_Toc201341849)

[2.2.4 InfluxDB sebagai *Database Time-Series* 10](#_Toc201341850)

[2.2.5 Visualisasi Real-Time Menggunakan Grafana](#_Toc201341851)  10

[2.2.6 Blockchain 10](#_Toc201341852)

[2.2.7 Web 3 11](#_Toc201341852)

[2.2.8 Metamask 11](#_Toc201341852)

[BAB 3 METODOLOGI 13](#_Toc201341853)

[3.1 Diagram Alur Sistem 13](#_Toc201341854)

[3.2 Pembuatan Program 14](#_Toc201341855)

[3.2.1 Kode Rush Modbus Client 14](#_Toc201341856)

[3.2.2 Kode Rust TCP Server 15](#_Toc201341857)

[3.2.3 Konfigurasi InfluxDB dan Integrasi 15](#_Toc201341858)

[3.2.4 Konfiguasi pada *Dapp* untuk blockchain 16](#_Toc201341859)

[BAB 4 Hasil dan Pembahasan 18](#_Toc201341864)

[4.1 Hasil 18](#_Toc201341865)

[4.1.1 Tampilan *Dashboard* Grafana 18](#_Toc201341866)

[4.1.2 Tampilan *Dashboard* QT 19](#_Toc201341867)

[4.1.3 Tampilan Data menggunakan *decentralized application (Dapp)* 20](#_Toc201341868)

[4.1.4 Tampilan Notifikasi pada Telegram 21](#_Toc201341869)

[4.2 Pembahasan 21](#_Toc201341875)

[BAB 5 Kesimpulan dan Saran 23](#_Toc201341876)

[5.1 Kesimpulan 23](#_Toc201341877)

[5.2 Saran 23](#_Toc201341878)

[DAFTAR PUSTAKA 24](#_Toc201341879)

[LAMPIRAN 25](#_Toc201341880)

DAFTAR GAMBAR

[**Gambar 3.1** Diagram Alur Sistem 13](#_Toc201218823)

**Gambar 3.2** Kode Rush Modbus 13

[**Gambar 3.3** Kode modbus untuk dikirim ke TCP Server 13](#_Toc201218823)

[**Gambar 3.4** Kode Rush TCP server 13](#_Toc201218823)

**Gambar 3.5** Hasil dari query di Influx DB 13

**Gambar 3.6** Dapp.js program untuk blockchain 13

[**Gambar 4.1** Data *Dashboard* Grafana 18](#_Toc201218824)

[**Gambar 4.2** Tampilan *Dashboard* QT 19](#_Toc201218825)

[**Gambar 4.3** Tampilan *Dashboard DApp* 20](#_Toc201218826)

[**Gambar 4.4** Tampilan notifikasi Telegram melalui alerts 21](#_Toc201218827)

DAFTAR TABEL

[**Tabel 2.1** State of The Art 8](#_Toc201218839)

# PENDAHULUAN

## Latar Belakang

Perkembangan ilmu pengetahuan dan teknologi dalam peradaban manusia telah mendorong kemajuan pesat industri di Indonesia, mulai dari penggunaan teknologi sederhana hingga penerapan teknologi canggih. Seiring dengan kemajuan tersebut, pembangunan juga menuntut standar keselamatan dan kesehatan kerja yang lebih tinggi untuk mengantisipasi dampak negatif dari kemajuan teknologi terhadap investasi, seperti meningkatnya risiko penyakit, kebakaran, dan kecelakaan kerja yang dapat berujung pada kematian. Kejadian-kejadian ini berpotensi menghambat laju industrialisasi serta menyebabkan kerugian atau bahkan hilangnya investasi.

Pada hakikatnya, penerapan keselamatan dan kesehatan kerja oleh suatu perusahaan mencerminkan bentuk penghormatan dan pengakuan terhadap nilai-nilai kemanusiaan. Penghormatan ini diwujudkan melalui langkah-langkah preventif guna mencegah terjadinya kecelakaan kerja baik bagi pekerja maupun individu lain yang berada di lingkungan kerja tersebut.

Pemerintah Indonesia telah menetapkan berbagai regulasi untuk melindungi tenaga kerja dan lingkungan kerja. Salah satunya adalah melalui Peraturan Pemerintah Nomor 28 Tahun 2024, yang merupakan pelaksanaan dari Undang-Undang Nomor 17 Tahun 2023 tentang Kesehatan. Peraturan ini menegaskan pentingnya penyelenggaraan upaya kesehatan, termasuk di lingkungan kerja, guna memastikan keselamatan dan kesehatan pekerja. Selain itu, Peraturan Menteri Ketenagakerjaan Nomor 17 Tahun 2024 juga mengatur tentang layanan pengawasan ketenagakerjaan serta keselamatan dan kesehatan kerja (K3), sebagai bentuk penguatan komitmen pemerintah agar setiap perusahaan memenuhi hak-hak tenaga kerja.

Industri minyak dan gas (migas) merupakan salah satu sektor dengan risiko kerja tinggi, terutama di area rawan kebakaran akibat sifat mudah terbakar dari material yang ditangani. Oleh karena itu, penerapan sistem monitoring lingkungan kerja berbasis teknologi sangat diperlukan. Salah satu pendekatan yang relevan dan efektif adalah penggunaan Internet of Things (IoT) untuk memantau parameter lingkungan seperti suhu dan kelembapan secara real-time.

Sistem monitoring suhu dan kelembapan berbasis IoT dapat memberikan informasi cepat dan akurat kepada pihak terkait dalam mendeteksi potensi bahaya, khususnya di area rawan kebakaran. Penerapan sistem ini tidak hanya mendukung langkah pencegahan kecelakaan kerja, tetapi juga menjadi bagian penting dari pelaksanaan budaya K3 di lingkungan industri migas. Dengan demikian, sistem ini diharapkan mampu mendukung keselamatan kerja sekaligus meningkatkan efisiensi dan keamanan operasional di sektor migas

## Rumusan Masalah

Berdasarkan latar belakang diatas, maka rumusan masalah yang dapat kami ambil adalah sebagai berikut

1. Bagaimana cara melakukan monitoring secara realtime pada sekitar area industri migas?
2. Apasaja dampak dari terkendalinya monitoring suhu dan kelembapan terhadap keamanan di sekitar area industri migas guna menunjang K3?
3. Bagaimana sistem monitoring ini dapat membantu keamanan para pekerja di sekitar area industri migas?

## Batasan Masalah

Agar penelitian ini lebih fokus dan sesuai dengan ruang lingkup yang ditentukan, maka batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Sistem monitoring lingkungan hanya memantau dua parameter utama, yaitu suhu dan kelembapan, dengan menggunakan sensor SHT 20
2. Komunikasi data dari sensor dibatasi hanya pada dua jenis protokol, yaitu TCP/IP dan Modbus RTU, yang merepresentasikan hasil melalui tampilan grafik dan nilai suhu dan kelembapan
3. Data hasil pembacaan sensor akan disimpan dalam basis data time-series InfluxDB dan divisualisasikan secara real-time melalui platform Grafana.
4. Studi kasus yang kami angkat difokuskan pada monitoring suhu dna kelembapan secara real-time pada area sekitar industri migas.

## Tujuan

Dari rumusan masalah yang kami dapat, maka tujuan dari tugas ini adalah

1. Untuk mengetahui cara melakukan monitoring secara real-time pada area sekitar industri migas.
2. Untuk menganalisis dampak dari monitoring suhu dan kelembapan yang terkendali terhadap peningkatan keamanan di area industri migas, khususnya dalam mendukung pelaksanaan K3 (Keselamatan dan Kesehatan Kerja).
3. Untuk merancang sistem monitoring yang dapat meningkatkan keamanan dan keselamatan kerja bagi para pekerja di sekitar area industri migas.

# TINJAUAN PUSTAKA

## Hasil Penelitian Terdahulu

**Tabel 2.1 State of The Art**

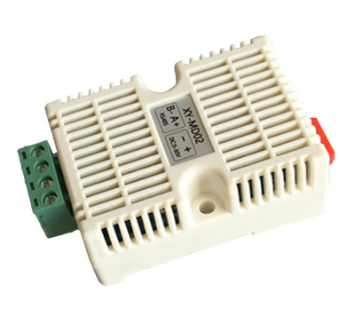
|  |  |  |
| --- | --- | --- |
| **Judul, penulis & Tahun** | **Metode** | **Hasil** |
| |  | | --- | | *Industrial Symbiosis: Context-Aware Strategies for Automated Negotiation of Smart Contracts in Peer-to-Peer Markets of Prosumers* (Dan E. Kröhling, Ernesto C. Martín 2020) | | |  | | --- | | perusahaan prosumer memproduksi barang dan menyediakan layanan melalui negosiasi otomatis Kontrak Cerdas melalui teknologi baru seperti *Internet of Things* (IoT) dan Blockchain di Pasar *Peer-to-Peer* (P2P). | | Negotiation outcomes: Agen yang menggabungkan model lawan dengan prior yang informatif (berbasis konteks, preferensi privat) menghasilkan keuntungan |
| *Evaluating the Potential Adoption of Blockchain and IoT in Nigeria's Oil and Gas Industry Sector: A Technology Acceptance Model Approach* (Yusuf, Nabil, & Liao, 2024) | Penelitian ini mengevaluasi minat dan potensi adopsi teknologi *blockchain* dan *Internet of Things* (IoT) di sektor minyak dan gas Nigeria melalui Technology Acceptance Model (TAM). | Hasil penelitian memperkaya TAM dengan memasukkan konteks industri dan lingkungan operasional serta dimensi *industry dynamics* tambahan, membuat model lebih relevan di konteks negara berkembang. |
| *Perkembangan Internet of Things (IoT) pada Sektor Energi : Sistematik Literatur Review* (Anggy Giri Prawiyogi, 2023) | mengenai penggunaan IoT dalam sistem energi secara umum dan *smart grid* secara spesifik. Studi ini mengkaji penerapan IoT di seluruh sektor energi | privasi dan keamanan yang muncul dengan implementasi IoT di sektor energi, serta beberapa solusi potensial seperti teknologi *blockchain* |

## Komponen dan Protocol Program

### Sensor MD02/SHT 40

Sensor SHT40 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°C serta kelembapan relatif dari 0 hingga 100% RH. Akurasi pengukuran suhu mencapai ±0,2°C dan kelembapan ±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.



### 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 commond 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. Sedangkan perbedaan tegangan dibawah 200mV akan mengakibatkan keluaran yang tidak terdefinisi, atau diabaikan



### TCP (Transmission Control Protocol)

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 *std::net::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.

### InfluxDB sebagai *Database Time-Series*

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.

### Visualisasi Real-Time Menggunakan Grafana

Grafana adalah perangkat lunak visualisasi dan analitik yang bersifat opensource. Grafana memungkinkanuntuk memvisualisasikan, mengingatkan, dan menjelajahi metrik disimpan (Dede Rahman, 2020). Alat untuk mengubah data *timeseries Database* (TSDB) menjadi grafik dan visualisasi.Grafana sangat cocok untuk membuat Dashboard yang dinamis dengan berbagai menu bawaan. Grafana jugamemiliki dashboard template yang bisa digunakan untuk mengumpulkan variabel data yang digunakan .Dalam paparan ini dijelaskan bahwa Grafana sangat support dalam visualisasi data dalam bentuk time series .

Grafana digunakan untuk menampilkan status service yang berjalan pada aplikasi maupun server yangdigunakan. Namun Grafana bukan hanya digunakan untuk itu saja tetapi juga dapat digunakan dalam visualisasisensor industri, pengimplementasian *Internet of thing*(IoT), pengamatan cuaca dan pengontrolan proses yang sedang berjalan. Terdapat berbagai macam pilihan untuk membuat dashboard yang bagus . Grafana memilikibeberapa panel yang dapat digunakan seperti grafik, singlestat, dashlist, tabel dan teks. Panel grafik digunakan dalam pembuatan grafik metric dan seri sebanyak yang diinginkan.

Beberapa data source yang didukung dari Grafana, antara lain *: Graphite, InfluxDB, OpenTSDB, Prometheus,Elasticsearch, dan CloudWatch*. Grafana hadir dengan plugin data source yang sangat lengkap untuk InfluxDb.Grafana mendukung berbagai macam editor query dengan fitur yang kaya, anotasi dan templating queries

### Blockchain

Blockchain adalah sebuah inovasi teknologi yang memungkinkan terbentuknya kepercayaan dalam sistem yang tersebar luas tanpa bergantung pada pihak otoritatif pusat. Teknologi ini pertama kali diperkenalkan oleh tokoh anonim bernama Satoshi Nakamoto melalui publikasi berjudul *"Bitcoin: A Peer-to-Peer Electronic Cash System"*, yang menjadi dasar terciptanya mata uang digital Bitcoin. Seiring waktu, aplikasi blockchain tidak hanya terbatas pada dunia kripto, tetapi juga telah meluas ke sektor lain seperti keuangan, pengelolaan hak kepemilikan, hingga implementasi kontrak pintar (*smart contract*). (Amit Kumar Das, 2024)

Secara konseptual, blockchain merupakan struktur data berbentuk rangkaian blok yang saling terhubung. Setiap blok menyimpan informasi transaksi, penanda waktu, dan nilai hash kriptografi dari blok sebelumnya. Hubungan berantai antarblok ini membentuk sistem pencatatan yang sulit dimanipulasi, karena perubahan pada satu blok akan memengaruhi blok-blok selanjutnya, menjadikan manipulasi data mudah terdeteksi.

Salah satu kekuatan utama dari blockchain adalah arsitektur pencatatannya yang bersifat desentralisasi. Tidak seperti sistem penyimpanan data terpusat, blockchain menyebarkan salinan data transaksi ke seluruh node dalam jaringan. Karena itu, sistem ini lebih tangguh terhadap kerusakan, manipulasi, atau gangguan, karena setiap node dapat saling memverifikasi keaslian data yang tersimpan.

### Web 3

Web3 merupakan tahap lanjutan dari perkembangan teknologi web, dikenal sebagai generasi ketiga dari arsitektur internet, yang menekankan desentralisasi, keterbukaan, dan kepemilikan data secara individu. Evolusi ini hadir setelah era Web1 yang bersifat statis (hanya bisa membaca), serta Web2 yang memperkenalkan kemampuan interaktif antara pengguna dan sistem (membaca dan menulis). Dalam era Web3, pengguna memiliki kendali lebih besar terhadap data yang mereka hasilkan melalui sistem terdesentralisasi berbasis blockchain. (heridan, 2022)

Teknologi ini dibangun di atas fondasi beberapa komponen utama, antara lain blockchain publik seperti Ethereum dan Bitcoin, kontrak pintar (*smart contracts*), dompet digital yang berfungsi sebagai identitas pengguna, serta pustaka pengembangan Web3 seperti Web3.js, Ethers.js, dan Truffle Suite. Seluruh elemen ini berkontribusi dalam membentuk ekosistem aplikasi terdesentralisasi (*DApps*) yang tidak lagi bergantung pada server pusat atau perusahaan penyedia layanan konvensional. Pengguna dapat berinteraksi langsung dengan sistem dan data secara aman melalui mekanisme terbuka yang tertanam dalam jaringan blockchain. Keunggulan utama dari Web3 terletak pada kemampuannya menjalankan proses digital yang aman dan otonom menggunakan smart contracts, yaitu kode yang berperan sebagai perjanjian otomatis dan dieksekusi secara transparan oleh setiap node dalam jaringan. Hal ini membuka peluang untuk menciptakan sistem yang lebih adil, efisien, dan bebas dari kendali tunggal.

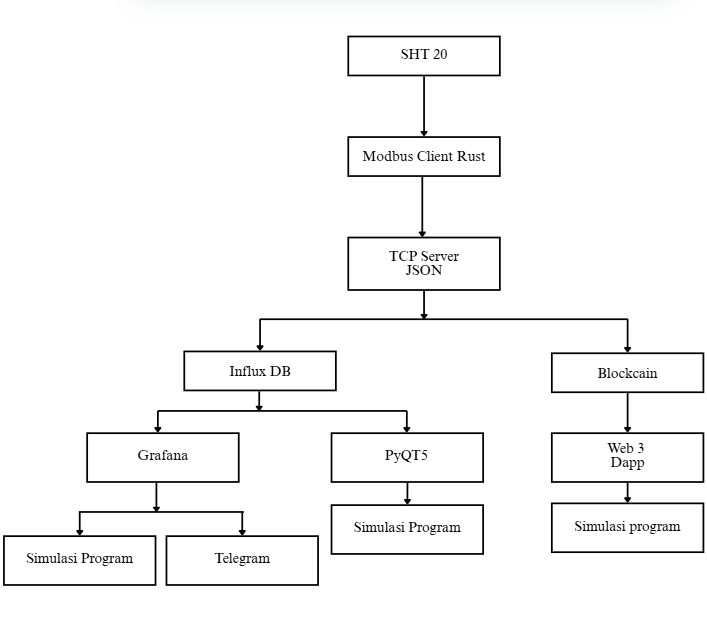
### Metamask

MetaMask merupakan ekstensi dompet digital berbasis web yang memungkinkan pengguna untuk terhubung langsung ke jaringan blockchain melalui browser seperti *Chrome, Firefox, Opera, dan Brave*. Ekstensi ini mengubah browser biasa menjadi "blockchain browser", memungkinkan pengguna untuk menjalankan aplikasi terdesentralisasi (DApp) di jaringan Ethereum tanpa harus menjalankan seluruh node Ethereum secara lokal. (Khan, Nand, Bhushan, Hameed, & Jamil, 2024)

MetaMask bekerja dengan membuat akun Ethereum yang terdiri atas alamat publik dan *private key*, yang digunakan untuk menandatangani transaksi digital secara aman. Selain itu, MetaMask juga menyimpan dan mengelola riwayat transaksi pengguna serta memungkinkan impor dan ekspor *private key*. Untuk mendukung pengalaman pengguna, MetaMask menyediakan antarmuka yang ramah pengguna, sekaligus menyediakan fungsi-fungsi penting seperti verifikasi tanda tangan digital, pengelolaan nonce, serta otentikasi berbasis tanda tangan (*signature-based authentication*) terhadap *DApp* yang terhubung dengannya

# METODOLOGI

## Diagram Alur Sistem



**Gambar 3.1** diagram Alur Sistem

Diagram alur pada gambar di atas menggambarkan sistem pemantauan suhu dan kelembapan yang menggunakan sensor SHT20 sebagai sumber data utama. Data dari sensor ini dikomunikasikan melalui protokol Modbus yang diimplementasikan dalam bahasa pemrograman Rust, bertindak sebagai Modbus Client. Selanjutnya, data diteruskan ke sebuah TCP Server berbasis JSON yang juga dikembangkan dengan Rust untuk menangani distribusi data secara real-time. Dari TCP Server, data disalurkan ke dua jalur pemrosesan utama. Jalur pertama mengarahkan data ke InfluxDB, yaitu basis data time-series yang memungkinkan pencatatan data historis secara efisien. InfluxDB kemudian terhubung dengan Grafana untuk visualisasi data dalam bentuk dashboard yang dapat diakses pengguna, serta dengan Telegram untuk pengiriman notifikasi otomatis. Selain itu, antarmuka GUI juga disediakan melalui PyQt5 yang memungkinkan pengguna menjalankan simulasi program secara lokal. Jalur kedua memfokuskan pada pencatatan data ke dalam sistem blockchain guna menjamin integritas, transparansi, dan keamanan data. Data ini kemudian dapat diakses melalui platform Web3 dalam bentuk aplikasi terdesentralisasi (DApp) yang juga menyediakan antarmuka simulasi program.

## Pembuat Program

Dalam tugas ini, kami membagi 3 bagian

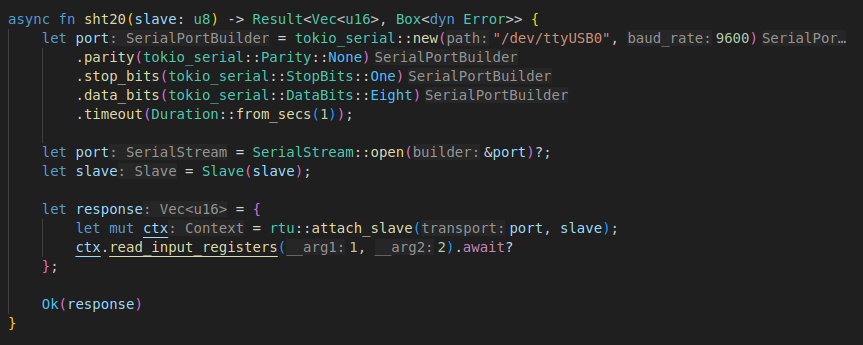
1. Program code Rush Modbus Client yang digunakan untuk membaca program data yang dikirim oleh RS-458 melalui Modbus RTU.
2. Program Rush mengirim data ke TCP dalam bentuk format JSON dan kemudian mengirim influx DB untu ditampilkan ke grafana dan blockchain
3. Visualisasi dimana terdiri atas beberapa visual, antara lain visual pada grafana, visual pada *Dapp* dan visual deaktop QT.

### Kode Rust Modbus Client

Untuk membaca data dari sensor menggunakan protokol Modbus RTU, langkah pertama yang dilakukan adalah menghubungkan sensor ke sistem melalui koneksi serial, yaitu menggunakan antarmuka RS-485 to USB. Pada sistem operasi Linux, koneksi ini biasanya dikenali sebagai port serial dengan path /dev/ttyUSB0. Sensor yang digunakan, seperti sensor SHT20, umumnya memiliki alamat slave sebesar 0x01 dalam jaringan Modbus.

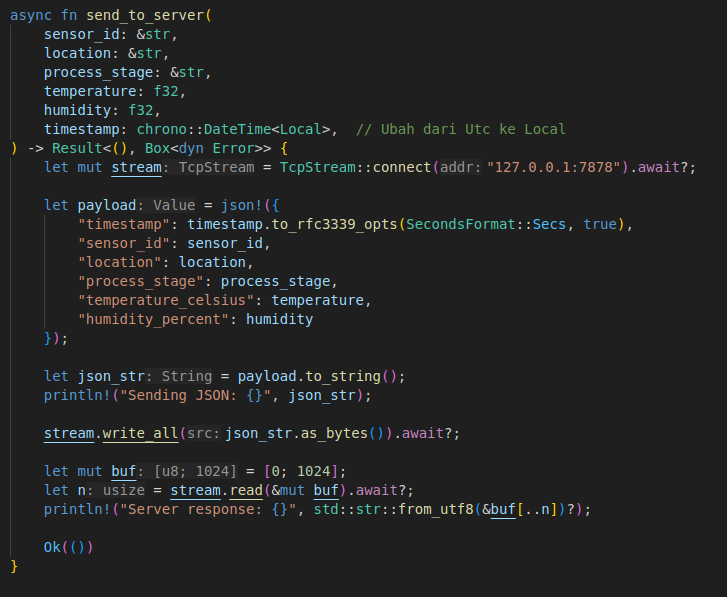
Selanjutnya, proses pembacaan data dilakukan dengan mengakses register tertentu pada sensor. Untuk parameter suhu, data dibaca dari register alamat 0x0000, sedangkan untuk kelembaban dibaca dari alamat register 0x0001. Nilai yang diperoleh dari sensor biasanya masih dalam bentuk integer dan perlu dikonversi ke bentuk float dengan membaginya dengan angka 10, karena sensor umumnya mengirimkan data dalam format nilai dikali 10.

Setelah data suhu dan kelembaban berhasil dibaca dan dikonversi, data tersebut diubah ke dalam format JSON agar mudah diproses dan ditransmisikan. Kemudian, data JSON tersebut dikirimkan ke server melalui koneksi TCP menggunakan port 7878. Proses pengiriman ini dilakukan secara periodik setiap 10 detik, sehingga server akan terus menerima pembaruan data sensor secara real-time.



**Gambar 3.2** Kode Rush Modbus

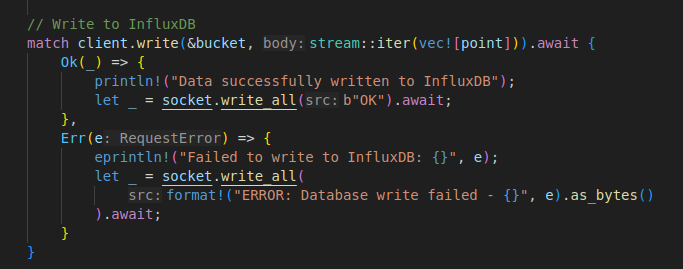
Data yang masuk akan dikirim ke TCP Server



**Gambar 3.3** Kode modbus untuk dikirim ke TCP Server

### Kode Rust TCP Server

Tahap kedua adalah menerima data dari client menggunakan program Rust TCP Server. Server ini terus-menerus mmebaca koneksi pada port USB. Ketika data JSON diterima, program memprosesnya menggunakan pustaka serde\_json untuk memparsing informasi yang dikirim oleh client. Setelah parsing, data disiapkan dan dikirimkan ke InfluxDB, sebuah database time-series yang dirancang untuk menyimpan data yang berurutan berdasarkan waktu. Di sini, data disimpan dengan struktur measurement untuk mencatat nama data, tags untuk metadata seperti ID sensor, dan fields untuk nilai aktual seperti suhu dan kelembaban.



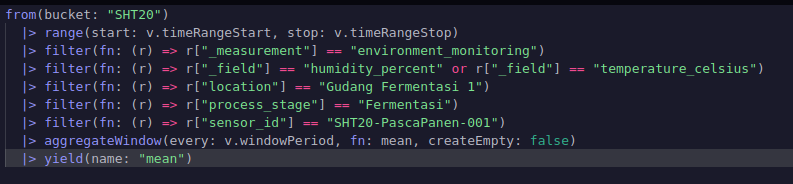
**Gambar 3.4** Kode Rush TCP server

### Konfigurasi InfluxDB dan Integrasi

Server dibangun untuk berjalan di port 7878 pada *localhost*, dan dirancang agar dapat menerima koneksi dari beberapa client secara asynchronous. Hal ini memungkinkan sistem untuk menangani banyak koneksi secara bersamaan tanpa mengganggu performa atau menghambat proses lainnya.

Setelah koneksi diterima, data yang dikirim oleh client diterima dalam bentuk string dengan format JSON. Data tersebut kemudian diparsing menggunakan library serde\_json, yang memungkinkan proses deserialisasi data JSON menjadi objek terstruktur. Dari hasil parsing, sistem mengekstrak nilai-nilai penting, yaitu temperature, humidity, serta informasi location yang menunjukkan asal data sensor.

Selanjutnya, data yang telah diproses disiapkan untuk disimpan ke dalam database time-series InfluxDB. Struktur penyimpanan data dibuat dalam bentuk data point, dengan measurement bernama "fermentation". Untuk mempermudah proses query dan filtering, location digunakan sebagai tag, sedangkan temperature dan humidity disimpan sebagai field yang berisi nilai aktual pengukuran. Proses penulisan data ke InfluxDB dilakukan menggunakan client InfluxDB versi 2, yang mendukung operasi *write* secara efisien dan kompatibel dengan format data time-series.



**Gambar 3.5** Hasil dari query di Influx DB

### Konfiguasi pada *Dapp* untuk blockchain



**Gambar 3.6** Dapp.js program untuk blockchain

Program daiatas merupakan bagian awal dari aplikasi frontend berbasis React yang dirancang untuk berinteraksi dengan jaringan *blockchain Ethereum*. Aplikasi ini memanfaatkan library ethers.js untuk menghubungkan dan berinteraksi dengan dompet digital pengguna seperti MetaMask, termasuk membaca saldo, mengelola transaksi, atau mengakses smart contract. Selain itu, aplikasi ini menggunakan *react-chartjs-2* untuk menampilkan data dalam bentuk grafik garis, yang kemungkinan digunakan untuk menyajikan histori transaksi, harga token, atau aktivitas jaringan blockchain.

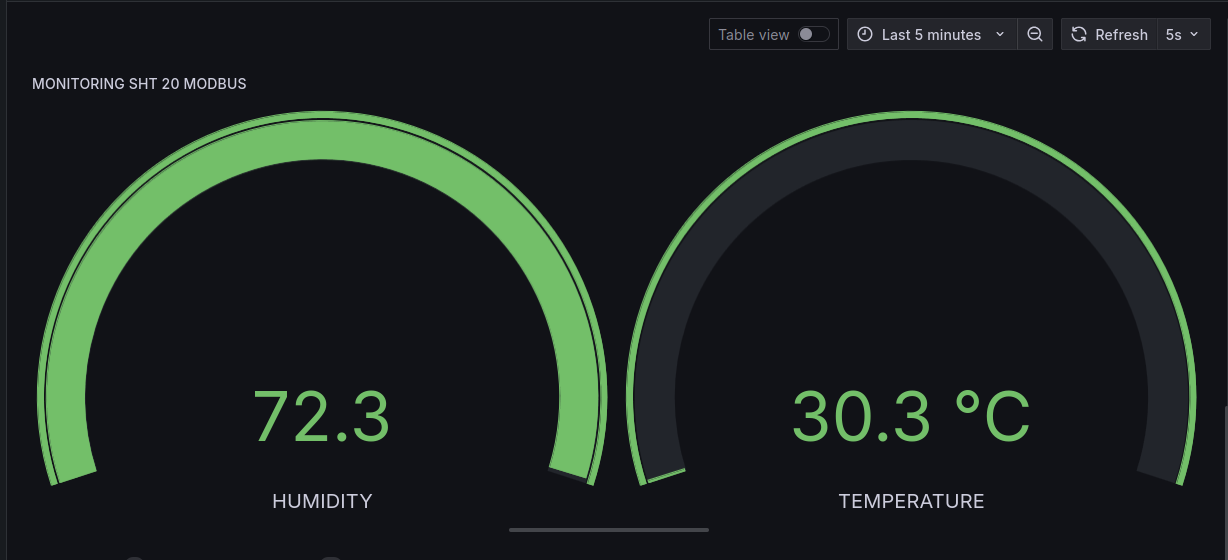
Tampilan antarmuka dibangun menggunakan berbagai komponen dari Material UI (MUI), seperti *Card, Grid, Typography, Button, Table,* dan *CircularProgress*. Komponen-komponen ini membantu menyusun tampilan aplikasi menjadi lebih responsif, terstruktur, dan ramah pengguna. Penggunaan Tabs dan Tab menunjukkan bahwa aplikasi memiliki sistem navigasi antar bagian, sementara Table dan turunannya digunakan untuk menampilkan data dalam format tabular, seperti daftar transaksi atau status kontrak. Elemen interaktif seperti *TextField, Slider*, dan *Button* juga disediakan agar pengguna dapat memasukkan data atau mengirim transaksi.

# HASIL DAN PEMBAHASAN

## Hasil

Berikut ini merupakan hasil tampilan setelah di generate semua program yang ada

### Tampilan *Dashboard* Grafana

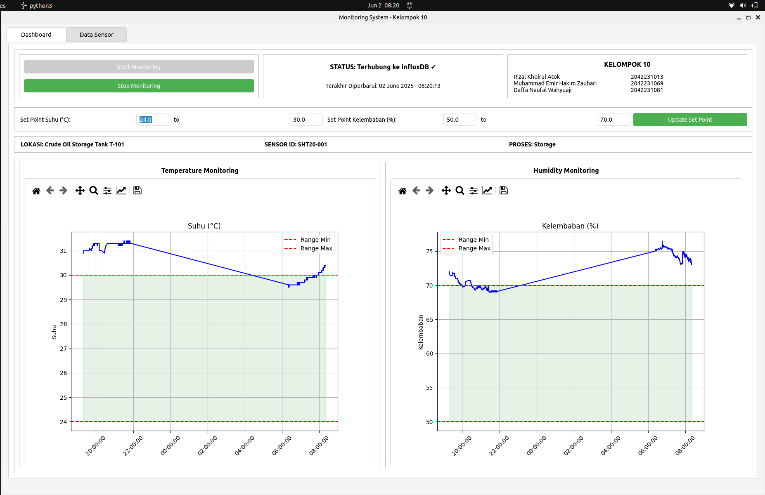




**Gambar 4.1** Data Dasboard Grafana

Gambar tersebut menunjukkan tampilan InfluxDB *Data Explorer* yang digunakan untuk melakukan *query* data sensor berbasis waktu. Dalam tampilan tersebut, *user* sedang menuliskan perintah untuk mengambil data dari *bucket* bernama "tugas3", menyaring data berdasarkan *measurement* "*monitoring*" dan *field* "*temperature*", serta menampilkan hasilnya. Grafik di bagian atas belum menampilkan visualisasi karena kemungkinan data belum tersedia dalam rentang waktu yang ditentukan. Antarmuka ini digunakan untuk menganalisis data suhu dan kelembaban yang dikirim oleh sensor secara real-time maupun historis.

### Tampilan *Dashboard* QT



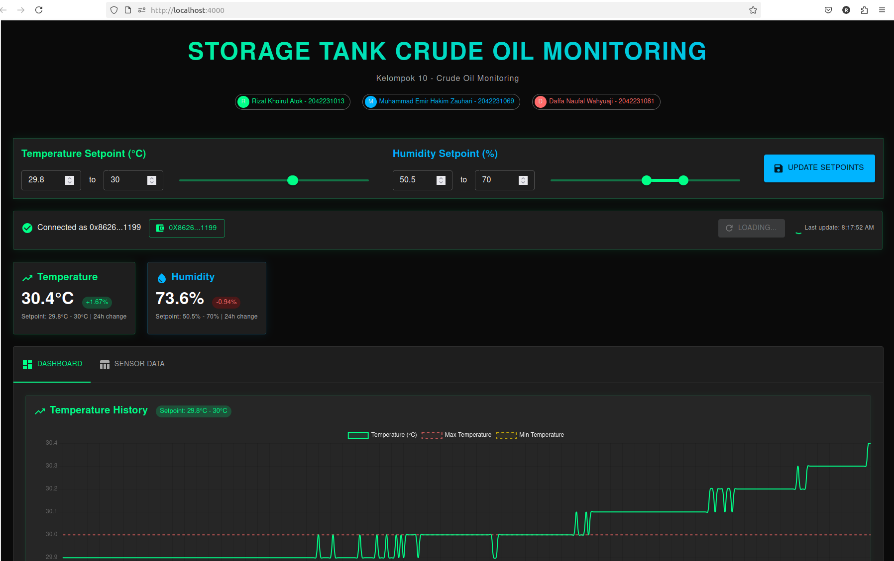
**Gambar 4.2** TampilanDashboard QT

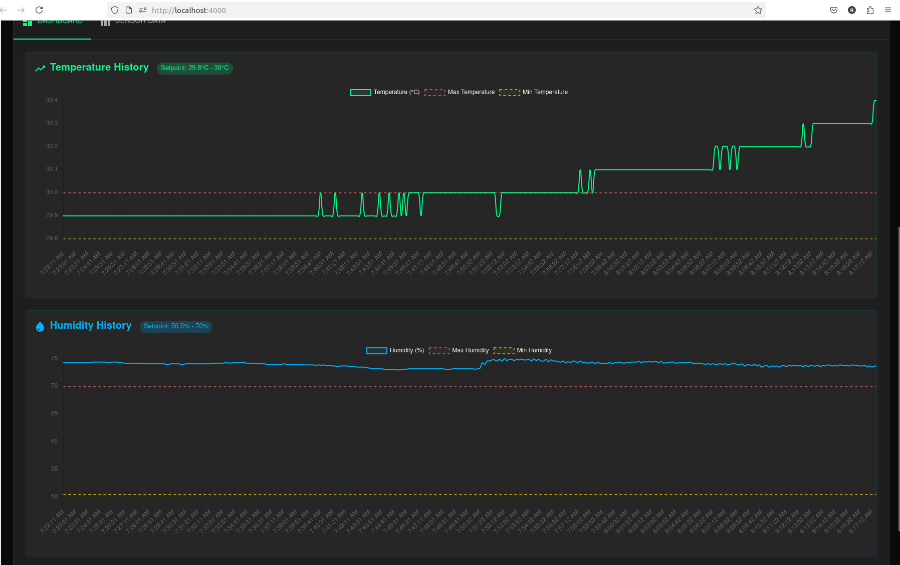
Berdasarkan data grafik suhu dan kelembaban dari sistem monitoring fermentasi kopi yang ditampilkan pada dashboard, dapat dilakukan analisis terhadap kestabilan lingkungan fermentasi dibandingkan dengan rentang optimal yang dianjurkan, yaitu suhu 24–30 °C dan kelembaban 50–70 %. Grafik suhu menunjukkan bahwa pada awal proses, suhu berada di sekitar 30 °C, kemudian meningkat secara bertahap hingga mencapai puncak sekitar 34 °C pada pukul 14:30. Setelah itu, suhu menurun dan menjelang pukul 18:00 berada di kisaran 31 °C. Artinya, suhu fermentasi sempat berada di luar batas atas rentang optimal selama beberapa jam, khususnya antara pukul 12:00 hingga 17:00.

Sementara itu, grafik kelembaban menunjukkan bahwa pada awal fermentasi, kelembaban relatif berada di kisaran 75–77 %, yang berada di atas ambang batas optimal. Namun, seiring waktu, kelembaban menurun dan stabil di kisaran 65–68 %, yang merupakan rentang ideal untuk proses fermentasi. Terdapat pula beberapa lonjakan kelembaban hingga 82 %, yang kemungkinan disebabkan oleh gangguan sistem, proses pembasahan ulang, atau gagguan lainnya.

Sehingga Berdasarkan analisis data suhu dan kelembaban yang ditampilkan pada dashboard, dapat dianalisa bahwa kondisi lingkungan untuk digunakan proses fermentasi belum sepenuhnya sesuai dengan rentang optimal yang dianjurkan, yaitu suhu 24–30 °C dan kelembaban 50–70 %. Suhu mengalami penyimpangan dengan puncak hingga 34 °C, yang berada di atas batas atas suhu optimal dan dapat mempengaruhi kualitas hasil fermentasi. Sementara itu, kelembaban juga sempat berada di atas 70 % di awal proses, meskipun akhirnya stabil di rentang optimal

### Tampilan Data menggunakan *decentralized application (DApp)*

****

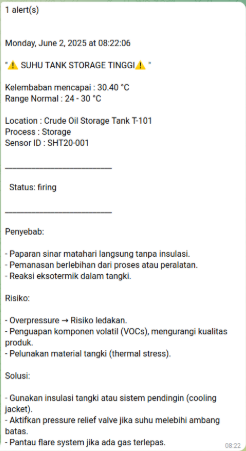
****

**Gambar 4.3** Tampilan Dashboard DApp

Gambar di atas menunjukkan tampilan dari Web3 Sensor *Dashboard* berbasis web yang terhubung dengan data dari *smart contract* di *blockchain Ethereum*. Gambar di sebelah kiri menyajikan data dalam bentuk tabel, yang berisi informasi waktu (*timestamp*), ID sensor, lokasi (misalnya Mobil Tanki 1), tahap proses (*Stage*), serta nilai suhu dan kelembaban. Tampilan ini memudahkan pengguna untuk melihat *log* data sensor secara historis.

Gambar di seblah kiri memperlihatkan kombinasi antara tabel data dan grafik visualisasi, yang menampilkan tren suhu dan kelembaban secara linier. Grafik tersebut membantu pengguna memantau kestabilan kondisi lingkungan dalam sistem secara *real-time* maupun historis. Seluruh data ditarik langsung dari blockchain melalui pemanggilan event smart contract menggunakan Ethers.js, menjadikan *dashboard* ini transparan dan tidak dapat dimanipulasi.

### Tampilan Notifikasi pada Telegram



**Gambar 4.4** Tampilan notifikasi Telegram melalui alerts

Tampilan telegam diatas menunjukkan peringatan mengenai suhu tinggi pada tangki penyimpanan minyak mentah (Crude Oil Storage Tank T-101). Berdasarkan data sensor SHT20-001, suhu di dalam tangki tercatat mencapai 30,40 °C, yang melebihi batas atas dari rentang suhu normal yaitu 24–30 °C. hal ini memungkinkan kita untuk mengetahui data suhu secara realtime dimanasaja dan kapan saja untuk menghindar kecelakaan kerja akibat suhu berlebih

## Pembahasan

Sistem yang dirancang dalam penelitian ini menunjukkan kinerja yang memuaskan dalam memantau suhu dan kelembapan secara real-time, khususnya pada area rawan kebakaran di industri minyak dan gas. Hasil visualisasi dari dashboard Grafana memperlihatkan bahwa sistem berhasil menangkap fluktuasi suhu dan kelembapan lingkungan yang berpotensi memicu bahaya kebakaran. Data yang ditampilkan secara historis melalui InfluxDB, serta antarmuka visual dari GUI PyQt dan Web3 DApp, memberikan kemudahan pemantauan bagi operator maupun stakeholder teknis.

Keunggulan utama dari sistem ini adalah pemanfaatan teknologi blockchain untuk pencatatan data secara permanen dan transparan. Dengan mencatat data sensor melalui smart contract Ethereum, integritas data dapat dipertanggungjawabkan karena tidak dapat dimodifikasi setelah dicatat. Hal ini sangat penting untuk konteks pemantauan K3 di sektor migas yang memerlukan transparansi data untuk audit dan laporan keselamatan.

Selain itu, penggunaan bahasa pemrograman Rust memberikan performa yang tinggi dan stabil pada client dan server, sedangkan protokol komunikasi Modbus RTU dan TCP/IP memberikan fleksibilitas integrasi dengan sistem yang telah ada. Tampilan antarmuka pengguna yang lengkap, baik melalui Grafana, PyQt5, maupun Web3 DApp, menunjukkan keberhasilan integrasi sistem secara menyeluruh dari sisi akuisisi data, penyimpanan, visualisasi, hingga aspek keamanan dan transparansi.

Namun demikian, dari hasil pengamatan grafik suhu dan kelembapan, sistem menunjukkan bahwa kondisi lingkungan belum sepenuhnya berada dalam rentang optimal yang ditetapkan (24–30 °C dan 50–70 % RH). Hal ini menandakan bahwa sistem berhasil mendeteksi potensi risiko lebih awal, yang merupakan nilai tambah dalam konteks penerapan keselamatan kerja. Notifikasi Telegram yang diaktifkan juga berfungsi efektif dalam memberi peringatan dini terhadap anomali suhu yang melebihi batas.

# KESIMPULAN DAN SARAN

## Kesimpulan

Berdasarkan hasil perancangan dan uji coba sistem monitoring suhu dan kelembaban yang digunakan sebagai monitoring di industry Minyak dan gas, guna menjaga K3 , dapat disimpulkan bahwa sistem yang dibangun mampu berfungsi secara baik dalam mendukung proses monitoring dari sekitar industry migas dengan optimal. Sistem ini menggunakan sensor SHT20 dengan protokol komunikasi Modbus RTU melalui antarmuka RS-485, yang memungkinkan pembacaan data suhu dan kelembaban secara akurat dan stabil.

Protokol Modbus RTU memungkinkan komunikasi yang andal antara mikrokontroler dan sensor SHT20, sementara TCP Server berperan sebagai jembatan pengirim data menuju basis data. Bahasa pemrograman Rust dipilih karena efisien dalam pengelolaan memori dan handal.

Penggunaan InfluxDB memungkinkan penyimpanan data suhu dan kelembaban secara historis dan efisien, sedangkan Grafana menyediakan visualisasi data yang informatif dan mendukung pemantauan kondisi lingkungan gudang secara terus menerus.

kondisi lingkungan fermentasi belum sepenuhnya berada dalam rentang optimal yang ditetapkan, yaitu suhu 24–30 °C dan kelembaban 50–70 %. Selama periode pemantauan, suhu sempat melebihi ambang batas atas hingga mencapai 34 °C, yang dapat mempercepat proses fermentasi secara tidak terkendali dan berpotensi menurunkan kualitas akhir biji kopi. Sementara itu, kelembaban juga mengalami kenaikan melebihi 75 % pada awal proses, meskipun secara bertahap kembali turun dan stabil di kisaran 65–68 %, yang berada dalam batas ideal.

## Saran

* 1. Penerapan di Skala Lapangan: Sistem sebaiknya diuji dalam skala lapangan dengan kondisi operasional nyata di area industri untuk menguji kestabilan jaringan, performa sensor dalam jangka panjang, dan respons sistem terhadap anomali lingkungan secara konsisten.
  2. Optimasi Interval Pembacaan Data: Waktu sampling saat ini adalah 5–10 detik; sistem akan lebih efisien bila dioptimalkan sesuai kebutuhan aktual berdasarkan variasi suhu lingkungan dan kebutuhan kecepatan respons.
  3. Integrasi Sistem Kendali Otomatis: Selain monitoring, sistem dapat dikembangkan lebih lanjut untuk melakukan pengendalian otomatis, seperti aktivasi sistem pemadam otomatis jika terdeteksi suhu abnormal.
  4. Keamanan Digital Lanjutan: Meskipun blockchain telah digunakan, perlu dipastikan bahwa dompet digital, node Ethereum, dan koneksi Web3 menggunakan lapisan keamanan tambahan seperti enkripsi end-to-end dan autentikasi multi-faktor.

DAFTAR PUSTAKA

Amit Kumar Das, M. T. (2024). Blockchain-Based Knowledge Repository for Training Artificial Intelligence Models: Bridging AIML with Decentralized Data.

Anggy Giri Prawiyogi, A. S. (2023). Perkembangan Internet of Things (IoT) pada Sektor Energi : Sistematik Literatur Review. *Jurnal MENTARI: ManajemenPendidikan dan Teknologi Informasi*.

Dede Rahman, H. A. (2020). Monitoring Server dengan Prometheus dan Grafana serta Notifikasi Telegram . *Jurnal Ilmiah Teknologi Sistem informasi*.

heridan, D. H. (2022). Web3: Challenges and Opportunities for the Market. Kennesaw State University & University of Guelph.

Khan, A., Nand, P., Bhushan, B., Hameed, A. A., & Jamil, A. (2024). A Review of Blockchain based Decentralised Authentication Solutions and their improvement through Metamask. *2024 2nd International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings)*.

Yusuf, S. I., Nabil, E. I., & Liao, I. Y. (2024). Evaluating the Potential Adoption of Blockchain and IoT in Nigeria's Oil and Gas Industry Sector: A Technology Acceptance Model Approach. *conferences*.

LAMPIRAN

* *Source code* Cargo.toml

|  |
| --- |
| [package]  name = "sht20"  version = "0.1.0"  edition = "2021"  [dependencies]  chrono = "0.4"  serde\_json = "1.0"  tokio = { version = "1.0", features = ["full"] }  tokio-modbus = "0.9"  tokio-serial = "5.4" |

* *Source code* mainrs

|  |
| --- |
| use chrono::{Local, SecondsFormat}; // Ubah dari Utc ke Local  use tokio\_modbus::{client::rtu, prelude::\*};  use tokio\_serial::SerialStream;  use tokio::{  net::TcpStream,  time::{sleep, Duration},  io::{AsyncReadExt, AsyncWriteExt},  };  use serde\_json::json;  use std::error::Error;  async fn sht20(slave: u8) -> Result<Vec<u16>, Box<dyn Error>> {  let port = tokio\_serial::new("/dev/ttyUSB0", 9600)  .parity(tokio\_serial::Parity::None)  .stop\_bits(tokio\_serial::StopBits::One)  .data\_bits(tokio\_serial::DataBits::Eight)  .timeout(Duration::from\_secs(1));    let port = SerialStream::open(&port)?;  let slave = Slave(slave);    let response = {  let mut ctx = rtu::attach\_slave(port, slave);  ctx.read\_input\_registers(1, 2).await?  };  Ok(response)  }  async fn send\_to\_server(  sensor\_id: &str,  location: &str,  process\_stage: &str,  temperature: f32,  humidity: f32,  timestamp: chrono::DateTime<Local>, // Ubah dari Utc ke Local  ) -> Result<(), Box<dyn Error>> {  let mut stream = TcpStream::connect("127.0.0.1:7878").await?;    let payload = json!({  "timestamp": timestamp.to\_rfc3339\_opts(SecondsFormat::Secs, true),  "sensor\_id": sensor\_id,  "location": location,  "process\_stage": process\_stage,  "temperature\_celsius": temperature,  "humidity\_percent": humidity  });  let json\_str = payload.to\_string();  println!("Sending JSON: {}", json\_str);    stream.write\_all(json\_str.as\_bytes()).await?;    let mut buf = [0; 1024];  let n = stream.read(&mut buf).await?;  println!("Server response: {}", std::str::from\_utf8(&buf[..n])?);    Ok(())  }  #[tokio::main]  async fn main() -> Result<(), Box<dyn Error>> {  let sensor\_id = "SHT20-PascaPanen-001";  let location = "Gudang Fermentasi 1";  let process\_stage = "Fermentasi";    loop {  let timestamp = Local::now(); // Ubah dari Utc::now() ke Local::now()    match sht20(1).await {  Ok(response) if response.len() == 2 => {  let temp = response[0] as f32 / 10.0;  let rh = response[1] as f32 / 10.0;    println!("[{}] {} - {}: Temp={:.1}°C, RH={:.1}%",  timestamp.format("%Y-%m-%d %H:%M:%S"),  location,  process\_stage,  temp,  rh);    if let Err(e) = send\_to\_server(  sensor\_id,  location,  process\_stage,  temp,  rh,  timestamp // Tetap menggunakan timestamp yang sama  ).await {  eprintln!("Failed to send data: {}", e);  }  }  Ok(invalid) => eprintln!("Invalid sensor response: {:?}", invalid),  Err(e) => eprintln!("Sensor read error: {}", e),  }    sleep(Duration::from\_secs(10)).await;  }  } |

* **CODINGAN TCP SERVER**
* cargo.toml

|  |
| --- |
| [package]  name = "tcp\_server"  version = "0.1.0"  edition = "2024"  [dependencies]  tokio = { version = "1.0", features = ["full"] }  serde = { version = "1.0", features = ["derive"] }  serde\_json = "1.0"  influxdb2 = "0.4.0"  chrono = "0.4.0"  futures = "0.3" |

* [main.rs](http://main.rs)

|  |
| --- |
| use influxdb2::Client;  use influxdb2::models::DataPoint;  use serde::Deserialize;  use tokio::{  io::{AsyncReadExt, AsyncWriteExt},  net::TcpListener,  };  use futures::stream;  use chrono::{Utc, DateTime};  #[derive(Debug, Deserialize)]  struct SensorData {  timestamp: String, // Required timestamp in ISO 8601 format  sensor\_id: String, // Required sensor identifier  location: String, // Required location  process\_stage: String, // Required process stage  temperature\_celsius: f64, // Temperature field with explicit unit  humidity\_percent: f64, // Humidity field with explicit unit  }  #[tokio::main]  async fn main() -> Result<(), Box<dyn std::error::Error>> {  // Configure InfluxDB connection  let influx\_url = "http://localhost:8086";  let influx\_org = "INSTITUT TEKNOLOGI SEPULUH NOPEMBER";  let influx\_token = "fXwgQMVodek-zcZDo-Q4W5RkEMMxs3yrNMFaX6hbpVyrHT\_kQdpk8TF64ul97VtgU2Ji3BPNNoMMIi6pdaE1Yg==";  let influx\_bucket = "SHT20";  let client = Client::new(influx\_url, influx\_org, influx\_token);  // Verify InfluxDB connection  match client.health().await {  Ok(health) => println!("InfluxDB connection healthy: {:?}", health),  Err(e) => {  eprintln!("Failed to connect to InfluxDB: {}", e);  return Err(e.into());  }  }  let listener = TcpListener::bind("127.0.0.1:7878").await?;  println!("Server running on 127.0.0.1:7878");  loop {  let (mut socket, \_) = listener.accept().await?;  let client = client.clone();  let bucket = influx\_bucket.to\_string();    tokio::spawn(async move {  let mut buf = [0; 1024];    match socket.read(&mut buf).await {  Ok(n) if n == 0 => return,  Ok(n) => {  let data = match std::str::from\_utf8(&buf[..n]) {  Ok(d) => d,  Err(e) => {  eprintln!("Error parsing data: {}", e);  let \_ = socket.write\_all(b"ERROR: Invalid UTF-8 data").await;  return;  }  };    println!("Received raw data: {}", data);    match serde\_json::from\_str::<SensorData>(data) {  Ok(sensor\_data) => {  println!("Data received: {:?}", sensor\_data);    // Parse the timestamp string into DateTime  let timestamp = match DateTime::parse\_from\_rfc3339(&sensor\_data.timestamp) {  Ok(dt) => dt.with\_timezone(&Utc),  Err(e) => {  eprintln!("Invalid timestamp format: {}", e);  let \_ = socket.write\_all(b"ERROR: Invalid timestamp format").await;  return;  }  };  // Konversi timestamp ke nanoseconds  let timestamp\_ns = timestamp.timestamp\_nanos\_opt().unwrap\_or\_else(|| {  eprintln!("Warning: Timestamp conversion failed, using 0 as fallback");  0  });  // Create data point with all fields  let point = DataPoint::builder("environment\_monitoring")  .tag("sensor\_id", &sensor\_data.sensor\_id)  .tag("location", &sensor\_data.location)  .tag("process\_stage", &sensor\_data.process\_stage)  .field("temperature\_celsius", sensor\_data.temperature\_celsius)  .field("humidity\_percent", sensor\_data.humidity\_percent)  .timestamp(timestamp\_ns) // Now passing i64  .build()  .unwrap();    // Write to InfluxDB  match client.write(&bucket, stream::iter(vec![point])).await {  Ok(\_) => {  println!("Data successfully written to InfluxDB");  let \_ = socket.write\_all(b"OK").await;  },  Err(e) => {  eprintln!("Failed to write to InfluxDB: {}", e);  let \_ = socket.write\_all(  format!("ERROR: Database write failed - {}", e).as\_bytes()  ).await;  }  }  }  Err(e) => {  eprintln!("Error parsing JSON: {}", e);  let \_ = socket.write\_all(  format!("ERROR: Invalid JSON - {}", e).as\_bytes()  ).await;  }  }  }  Err(e) => eprintln!("Error reading socket: {}", e),  }  });  }  } |

[main.py](http://main.py)

|  |
| --- |
| import sys  from PyQt6 import QtWidgets, QtCore  from PyQt6.QtWidgets import QMessageBox, QFileDialog  from PyQt6.QtCore import QTimer, QDateTime  from ui\_mainwindow import Ui\_MainWindow  from influxdb\_client import InfluxDBClient  from influxdb\_client.client.write\_api import SYNCHRONOUS  import matplotlib.pyplot as plt  from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg as FigureCanvas  from matplotlib.backends.backend\_qt5agg import NavigationToolbar2QT as NavigationToolbar  from matplotlib.figure import Figure  import datetime  import pytz  import mplcursors  import pandas as pd  class MonitoringApp(QtWidgets.QMainWindow, Ui\_MainWindow):  def \_\_init\_\_(self):  super().\_\_init\_\_()  self.setupUi(self)  # Konfigurasi InfluxDB  self.influx\_url = "http://localhost:8086"  self.influx\_org = "INSTITUT TEKNOLOGI SEPULUH NOPEMBER"  self.influx\_token = "oNEA7ExQ-JAaR3zVhfVbbAlAhc4AkcwMUv-QY\_8MAqXLX-Dq5jRUFyMtla-Ag92i-GaJcnWyYQ0yX4UezJ3raA=="  self.influx\_bucket = "SHT20"  # Inisialisasi variabel  self.client = None  self.query\_api = None  self.timer = QTimer()  self.update\_interval = 10000 # 10 detik  self.all\_data = pd.DataFrame() # Untuk menyimpan semua data  # Untuk menyimpan referensi garis chart  self.temp\_line = None  self.humidity\_line = None  self.temp\_cursor = None  self.humidity\_cursor = None  # Setup UI  self.setup\_ui()  # Hubungkan signal  self.startButton.clicked.connect(self.start\_monitoring)  self.stopButton.clicked.connect(self.stop\_monitoring)  # Tambahkan tombol ekspor untuk Tab 2  self.exportButton = QtWidgets.QPushButton(self.tab\_2)  self.exportButton.setGeometry(QtCore.QRect(1400, 30, 161, 31))  self.exportButton.setText("Ekspor ke Excel")  self.exportButton.clicked.connect(self.export\_to\_excel)  # Tambahkan tombol refresh untuk Tab 2  self.refreshButton = QtWidgets.QPushButton(self.tab\_2)  self.refreshButton.setGeometry(QtCore.QRect(1200, 30, 161, 31))  self.refreshButton.setText("Refresh Data")  self.refreshButton.clicked.connect(self.refresh\_table)  # Awalnya nonaktifkan tombol stop  self.stopButton.setEnabled(False)  def setup\_ui(self):  self.setup\_charts()  self.setup\_table()  def setup\_charts(self):  """Menyiapkan grafik untuk Tab 1"""  self.temp\_figure = Figure()  self.temp\_canvas = FigureCanvas(self.temp\_figure)  self.temp\_ax = self.temp\_figure.add\_subplot(111)  self.temp\_ax.set\_title('Suhu (°C) vs Waktu')  self.temp\_ax.set\_xlabel('Waktu (WIB)')  self.temp\_ax.set\_ylabel('Suhu (°C)')  self.temp\_ax.grid(True)  temp\_toolbar = NavigationToolbar(self.temp\_canvas, self)  temp\_layout = QtWidgets.QVBoxLayout()  temp\_layout.addWidget(temp\_toolbar)  temp\_layout.addWidget(self.temp\_canvas)  self.temperatureChartView.setLayout(temp\_layout)  self.humidity\_figure = Figure()  self.humidity\_canvas = FigureCanvas(self.humidity\_figure)  self.humidity\_ax = self.humidity\_figure.add\_subplot(111)  self.humidity\_ax.set\_title('Kelembaban (%) vs Waktu')  self.humidity\_ax.set\_xlabel('Waktu (WIB)')  self.humidity\_ax.set\_ylabel('Kelembaban (%)')  self.humidity\_ax.grid(True)  humidity\_toolbar = NavigationToolbar(self.humidity\_canvas, self)  humidity\_layout = QtWidgets.QVBoxLayout()  humidity\_layout.addWidget(humidity\_toolbar)  humidity\_layout.addWidget(self.humidity\_canvas)  self.humidityChartView.setLayout(humidity\_layout)  def setup\_table(self):  """Menyiapkan tabel untuk Tab 2"""  self.tableWidget.setColumnCount(5)  self.tableWidget.setHorizontalHeaderLabels([  "Waktu",  "Lokasi",  "Tahap Proses",  "Suhu (°C)",  "Kelembaban (%)"  ])  self.tableWidget.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.ResizeMode.Stretch)  self.tableWidget.setSortingEnabled(True)  def start\_monitoring(self):  """Memulai monitoring data"""  try:  self.client = InfluxDBClient(  url=self.influx\_url,  token=self.influx\_token,  org=self.influx\_org,  timeout=30\_000  )  self.query\_api = self.client.query\_api()  try:  health = self.client.health()  if health.status == "pass":  self.statusLabel.setText("STATUS: Terhubung ke InfluxDB ✔")  else:  self.statusLabel.setText("STATUS: Masalah Koneksi ⚠")  QMessageBox.warning(self, "Peringatan", f"Masalah koneksi InfluxDB: {health.message}")  return  except Exception as health\_error:  self.statusLabel.setText("STATUS: Gagal Cek Kesehatan ❌")  QMessageBox.warning(self, "Peringatan", f"Tidak bisa cek kesehatan InfluxDB: {str(health\_error)}")  return  self.startButton.setEnabled(False)  self.stopButton.setEnabled(True)  self.timer.timeout.connect(self.update\_data)  self.timer.start(self.update\_interval)  self.update\_data()  except Exception as e:  self.statusLabel.setText("STATUS: Gagal Koneksi ❌")  QMessageBox.critical(self, "Error", f"Gagal terhubung ke InfluxDB: {str(e)}")  if self.client:  self.client.close()  self.client = None  self.query\_api = None  def stop\_monitoring(self):  """Menghentikan monitoring"""  self.timer.stop()  if self.client:  self.client.close()  self.client = None  self.query\_api = None  self.statusLabel.setText("STATUS: Terputus ⛔")  self.startButton.setEnabled(True)  self.stopButton.setEnabled(False)  QMessageBox.information(self, "Info", "Monitoring dihentikan")  def update\_data(self):  """Memperbarui data dari InfluxDB"""  if not self.query\_api:  self.statusLabel.setText("STATUS: Query API tidak tersedia ❌")  QMessageBox.warning(self, "Peringatan", "Query API belum diinisialisasi")  return  try:  query = f'''  from(bucket: "{self.influx\_bucket}")  |> range(start: -24h)  |> filter(fn: (r) => r["\_measurement"] == "environment\_monitoring")  |> filter(fn: (r) => r["\_field"] == "humidity\_percent" or r["\_field"] == "temperature\_celsius")  |> filter(fn: (r) => r["location"] == "Gudang Fermentasi 1")  |> filter(fn: (r) => r["process\_stage"] == "Fermentasi")  |> filter(fn: (r) => r["sensor\_id"] == "SHT20-PascaPanen-001")  |> aggregateWindow(every: 1m, fn: mean, createEmpty: false)  |> yield(name: "mean")  '''  try:  result = self.query\_api.query(query)  except Exception as query\_error:  self.statusLabel.setText("STATUS: Error Query ⚠")  QMessageBox.warning(self, "Error Query", f"Gagal menjalankan query: {str(query\_error)}")  return  temp\_data = []  humidity\_data = []  temp\_times = []  humidity\_times = []  records\_list = [] # Untuk menyimpan data ke tabel  for table in result:  for record in table.records:  if record.get\_field() == "temperature\_celsius":  temp\_data.append(record.get\_value())  temp\_times.append(record.get\_time())  elif record.get\_field() == "humidity\_percent":  humidity\_data.append(record.get\_value())  humidity\_times.append(record.get\_time())  if not self.locationLabel.text().startswith("LOCATION:"):  self.locationLabel.setText(f"LOKASI: {record.values.get('location', 'N/A')}")  self.processStageLabel.setText(f"PROSES: {record.values.get('process\_stage', 'N/A')}")  self.sensorIdLabel.setText(f"SENSOR ID: {record.values.get('sensor\_id', 'N/A')}")  # Simpan data untuk tabel  records\_list.append({  'time': record.get\_time(),  'location': record.values.get('location', 'N/A'),  'process\_stage': record.values.get('process\_stage', 'N/A'),  'field': record.get\_field(),  'value': record.get\_value()  })  if temp\_data and temp\_times:  self.update\_chart(self.temp\_ax, self.temp\_canvas, temp\_times, temp\_data, 'Suhu (°C)')  if humidity\_data and humidity\_times:  self.update\_chart(self.humidity\_ax, self.humidity\_canvas, humidity\_times, humidity\_data, 'Kelembaban (%)')  # Perbarui data tabel  if records\_list:  self.update\_data\_table(records\_list)  now = QDateTime.currentDateTime()  self.updateLabel.setText(f"Terakhir Diperbarui: {now.toString('dd MMMM yyyy - hh:mm:ss')}")  except Exception as e:  self.statusLabel.setText("STATUS: Error Pembaruan ⚠")  QMessageBox.warning(self, "Error", f"Error memperbarui data: {str(e)}")  def update\_chart(self, ax, canvas, times, values, title):  """Memperbarui grafik dengan data baru"""  try:  ax.clear()  local\_tz = pytz.timezone('Asia/Jakarta')  local\_times = [t.astimezone(local\_tz) for t in times]  line, = ax.plot(local\_times, values, 'b-')  if title == 'Suhu (°C)':  if self.temp\_cursor:  self.temp\_cursor.remove()  self.temp\_line = line  else:  if self.humidity\_cursor:  self.humidity\_cursor.remove()  self.humidity\_line = line  ax.set\_title(title)  ax.set\_xlabel('Waktu (WIB)')  ax.set\_ylabel(title.split(' ')[0])  ax.grid(True)  ax.xaxis.set\_major\_formatter(plt.matplotlib.dates.DateFormatter('%H:%M:%S', tz=local\_tz))  plt.setp(ax.get\_xticklabels(), rotation=45)  formatted\_times = [t.strftime('%Y-%m-%d %H:%M:%S') for t in local\_times]  if title == 'Suhu (°C)':  self.temp\_cursor = mplcursors.cursor(line, hover=True)  def on\_add\_temp(sel):  idx = sel.target.index  sel.annotation.set\_text(  f"{title.split(' ')[0]}: {values[idx]:.2f}\nWaktu: {formatted\_times[idx]}"  )  self.temp\_cursor.connect("add", on\_add\_temp)  else:  self.humidity\_cursor = mplcursors.cursor(line, hover=True)  def on\_add\_humidity(sel):  idx = sel.target.index  sel.annotation.set\_text(  f"{title.split(' ')[0]}: {values[idx]:.2f}\nWaktu: {formatted\_times[idx]}"  )  self.humidity\_cursor.connect("add", on\_add\_humidity)  canvas.draw()  except Exception as e:  QMessageBox.warning(self, "Error Grafik", f"Error memperbarui grafik: {str(e)}")  def update\_data\_table(self, new\_records):  """Memperbarui tabel dengan data baru"""  try:  # Konversi records ke DataFrame  new\_df = pd.DataFrame(new\_records)    # Gabungkan dengan data yang sudah ada  if not self.all\_data.empty:  # Gabungkan dan hapus duplikat  self.all\_data = pd.concat([self.all\_data, new\_df]).drop\_duplicates(  subset=['time', 'field'],  keep='last'  )  else:  self.all\_data = new\_df    # Pivot data untuk tampilan tabel  df\_pivot = self.all\_data.pivot\_table(  index=['time', 'location', 'process\_stage'],  columns='field',  values='value'  ).reset\_index()    # Konversi waktu ke timezone lokal  local\_tz = pytz.timezone('Asia/Jakarta')  df\_pivot['time'] = pd.to\_datetime(df\_pivot['time']).dt.tz\_convert(local\_tz)    # Format waktu untuk tampilan  df\_pivot['time\_str'] = df\_pivot['time'].dt.strftime('%Y-%m-%d %H:%M:%S')    # Simpan data lengkap untuk ekspor  self.export\_data = df\_pivot.copy()    # Perbarui tabel  self.refresh\_table()  except Exception as e:  QMessageBox.warning(self, "Error Tabel", f"Error memperbarui tabel: {str(e)}")  def refresh\_table(self):  """Menyegarkan tampilan tabel dengan data terbaru"""  try:  if hasattr(self, 'export\_data') and not self.export\_data.empty:  # Nonaktifkan sorting sementara untuk performa  self.tableWidget.setSortingEnabled(False)    # Set jumlah baris sesuai data  self.tableWidget.setRowCount(len(self.export\_data))    # Isi tabel dengan data  for row\_idx, row in self.export\_data.iterrows():  self.tableWidget.setItem(row\_idx, 0, QtWidgets.QTableWidgetItem(row['time\_str']))  self.tableWidget.setItem(row\_idx, 1, QtWidgets.QTableWidgetItem(row['location']))  self.tableWidget.setItem(row\_idx, 2, QtWidgets.QTableWidgetItem(row['process\_stage']))  self.tableWidget.setItem(row\_idx, 3, QtWidgets.QTableWidgetItem(f"{row.get('temperature\_celsius', 'N/A'):.2f}"))  self.tableWidget.setItem(row\_idx, 4, QtWidgets.QTableWidgetItem(f"{row.get('humidity\_percent', 'N/A'):.2f}"))    # Aktifkan kembali sorting  self.tableWidget.setSortingEnabled(True)    except Exception as e:  QMessageBox.warning(self, "Error", f"Gagal menyegarkan tabel: {str(e)}")  def export\_to\_excel(self):  """Mengekspor data ke file Excel"""  if not hasattr(self, 'export\_data') or self.export\_data.empty:  QMessageBox.warning(self, "Peringatan", "Tidak ada data untuk diekspor")  return    try:  # Gunakan file dialog untuk memilih lokasi penyimpanan  file\_name, \_ = QFileDialog.getSaveFileName(  self,  "Simpan File Excel",  "",  "File Excel (\*.xlsx);;Semua File (\*)"  )    if file\_name:  # Pastikan ekstensi .xlsx  if not file\_name.endswith('.xlsx'):  file\_name += '.xlsx'    # Buat salinan data untuk ekspor  export\_df = self.export\_data.copy()    # Konversi kolom waktu ke timezone naive (tanpa timezone)  if 'time' in export\_df.columns:  export\_df['time'] = export\_df['time'].dt.tz\_localize(None)    # Siapkan data untuk ekspor  export\_df = export\_df[['time', 'location', 'process\_stage',  'temperature\_celsius', 'humidity\_percent']]  export\_df.columns = ['Waktu', 'Lokasi', 'Tahap Proses',  'Suhu (°C)', 'Kelembaban (%)']    # Ekspor ke Excel  export\_df.to\_excel(file\_name, index=False)  QMessageBox.information(self, "Sukses", "Data berhasil diekspor ke Excel")    except Exception as e:  QMessageBox.warning(self, "Error Ekspor", f"Gagal mengekspor data: {str(e)}")  def main():  app = QtWidgets.QApplication(sys.argv)  window = MonitoringApp()  window.show()  sys.exit(app.exec())  if \_\_name\_\_ == "\_\_main\_\_":  main() |