

LAPORAN INTERKONEKSI SISTEM INSTRUMENTASI - VI231418

PENERAPAN TEKNOLOGI *BLOCKCHAIN DAN SMART CONTRACT* DALAM RANTAI PASOK *MIDSTREAM MIGAS* UNTUK MENGURANGI DUPLIKASI TRANSAKSI DAN MENINGKATKAN AKURASI DATA

MUHAMMAD HADID QUSHAIRI

NRP 2042231025

KARINA LAILATUL MAGHFIROH MAHARANI

NRP 2042231035

GREISTA TEZAR RIZKI SAPUTRA

NRP 2042231079

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



LAPORAN INTERKONEKSI SISTEM INSTRUMENTASI - VI231418

PENERAPAN TEKNOLOGI *BLOCKCHAIN* DAN *SMART CONTRACT*
DALAM RANTAI PASOK *MIDSTREAM* MIGAS UNTUK MENGURANGI
DUPLIKASI TRANSAKSI DAN MENINGKATKAN AKURASI DATA

MUHAMMAD HADID QUSHAIRI

NRP 2042231025

KARINA LAILATUL MAGHFIROH MAHARANI

NRP 2042231035

GREISTA TEZAR RIZKI SAPUTRA

NRP 2042231079

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



INSTRUMENTATION SYSTEM INTERCONNECTION REPORT - VI231418

IMPLEMENTATION OF BLOCKCHAIN TECHNOLOGY AND SMART CONTRACTS IN THE MIDSTREAM OIL AND GAS SUPPLY CHAIN TO REDUCE TRANSACTION DUPLICATION AND IMPROVE DATA ACCURACY

MUHAMMAD HADID QUSHAIRI

Student ID: 2042231025

KARINA LAILATUL MAGHFIROH MAHARANI

Student ID: 2042231035

GREISTA TEZAR RIZKI SAPUTRA

Student ID: 2042231079

Supervising Lecturer

Ahmad Radhy, S.Si., M.Si

Lecturer ID: 2022198911049

Study Program of Instrumentation Technology Engineering

Department of Instrumentation Engineering

Vocational Faculty

Institut Teknologi Sepuluh Nopember

Surabaya

2025

LEMBAR PENGESAHAN

PENERAPAN TEKNOLOGI *BLOCKCHAIN* DAN *SMART CONTRACT* DALAM RANTAI PASOK *MIDSTREAM* MIGAS UNTUK MENGURANGI DUPLIKASI TRANSAKSI DAN MENINGKATKAN AKURASI DATA

LAPORAN AKHIR

Diajukan untuk memenuhi salah satu syarat
menyelesaikan tugas Interkoneksi Listem Instrumentasi
Program Studi D-4 Rekayasa Teknologi Instrumentasi
Departemen Teknik Instrumentasi

Fakultas Vokasi

Institut Teknologi Sepuluh Nopember Institut Teknologi Sepuluh Nopember

Oleh : **MUHAMMAD HADID QUSHAIRI**
NRP 2042231025

KARINA LAILATUL MAGHFIROH MAHARANI
NRP 2042231035

GREISTA TEZAR RIZKI SAPUTRA
NRP 2042231079

Disetujui oleh Dosen Pengampu :

1. Ahmad Radhy, S.Si., M.Si Dosen Pengampu

SURABAYA

JUNI 2025

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP : Muhammad Hadid Qushairi/2042231025
Karina Lailatul Maghfiroh Maharani/2042231035
Greista Tezar Rizki Saputra/2042231079
Program studi : Teknik Instrumentasi-FV
Dosen Pembimbing / NIP : Ahmad Radhy, S.Si., M.Si /2022198911049

dengan ini menyatakan bahwa Laporan Akhir dengan judul "**PENERAPAN TEKNOLOGI BLOCKCHAIN DAN SMART CONTRACT DALAM RANTAI PASOK MIDSTREAM MIGAS UNTUK MENGURANGI DUPLIKASI TRANSAKSI DAN MENINGKATKAN AKURASI DATA**" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 20 Juni 2025

Mahasiswa,

Mahasiswa,

Muhammad Hadid Qushairi
NRP. 2042231025

Karina Lailatul Maghfiroh Maharani
NRP. 2042231035

Mengetahui
Dosen Pembimbing

Mahasiswa,

Ahmad Radhy, S.Si., M.Si NPP.
NIP. 2022198911049

Greista Tezar Rizki Saputra
NRP. 2042231079

ABSTRAK

PENERAPAN TEKNOLOGI *BLOCKCHAIN* DAN *SMART CONTRACT* DALAM RANTAI PASOK *MIDSTREAM* MIGAS UNTUK MENGURANGI DUPLIKASI TRANSAKSI DAN MENINGKATKAN AKURASI DATA

Nama mahasiswa / NRP : Muhammad Hadid Qushairi/2042231025
Karina Lailatul Maghfiroh Maharani/2042231035
Greista Tezar Rizki Saputra/2042231079

Program studi : Teknik Instrumentasi FV-ITS

Dosen Pembimbing / NIP : Ahmad Radhy, S.Si., M.Si /2022198911049

Abstrak

Sektor midstream dalam industri minyak dan gas (migas) sering mengalami kendala berupa duplikasi transaksi dan ketidakakuratan data logistik. Penelitian ini merancang dan mengimplementasikan sistem *monitoring* suhu dan kelembaban berbasis sensor industri yang terintegrasi dengan teknologi *blockchain* dan *smart contract* sebagai solusi terhadap permasalahan tersebut. Sistem dibangun menggunakan bahasa pemrograman *Embedded Rust* dengan protokol komunikasi TCP/IP dan Modbus RTU, serta memanfaatkan InfluxDB dan Grafana untuk penyimpanan dan visualisasi data secara *real-time*. Selanjutnya, data dicatat ke dalam *blockchain* Ethereum lokal melalui *smart contract* berbasis Web3. Hasil pengujian menunjukkan bahwa sistem mampu mencatat data secara transparan dan tidak dapat diubah (*immutable*), mengurangi risiko kesalahan pencatatan, serta meningkatkan kepercayaan antar pemangku kepentingan dalam rantai distribusi migas. Sistem ini memberikan kontribusi nyata terhadap otomatisasi dan akuntabilitas dalam proses distribusi sektor *midstream* migas.

Kata kunci: *Blockchain*, *Smart Contract*, Sensor Industri, Rantai Pasok Migas.

ABSTRACT

ANALYSIS OF THE EFFECT OF LINKAGE LENGTH ON SERIES ACTIVE VARIABLE GEOMETRY SUSPENSION (SAVGS) RESPONSE

Student Name / NRP	: Muhammad Hadid Qushairi/2042231025 Karina Lailatul Maghfiroh Maharani/2042231035 Greista Tezar Rizki Saputra/2042231079
Department Advisor	: Teknik Instrumentasi FV-ITS : Ahmad Radhy, S.Si., M.Si /2022198911049

Abstract

The midstream sector of the oil and gas industry frequently encounters issues such as transaction duplication and inaccurate logistics data. This research designs and implements a temperature and humidity monitoring system based on industrial sensors integrated with blockchain and smart contract technology to address these challenges. The system is developed using the Embedded Rust programming language with TCP/IP and Modbus RTU communication protocols, and utilizes InfluxDB and Grafana for real-time data storage and visualization. The collected data is then recorded into a local Ethereum blockchain through Web3-based smart contracts. Testing results show that the system can transparently and immutably record data, reduce recording errors, and enhance trust among stakeholders in the midstream oil and gas supply chain. This system provides a practical contribution to automation and accountability in midstream distribution processes.

Keywords: Blockchain, Smart Contract, Industrial Sensor, Oil And Gas Supply Chain.

DAFTAR ISI

LEMBAR PENGESAHAN	i
PERNYATAAN ORISINALITAS	ii
ABSTRAK	iii
ABSTRACT	iv
DAFTAR ISI	v
DAFTAR GAMBAR	vii
DAFTAR TABEL	viii
BAB 1 PENDAHULUAN	9
1.1 Latar Belakang	9
1.2 Rumusan Masalah	9
1.3 Batasan Masalah	10
1.4 Tujuan	10
1.5 Manfaat	11
BAB 2 TINJAUAN PUSTAKA	12
2.1 Hasil Penelitian Terdahulu	12
2.2 Dasar Teori	12
2.2.1 Sistem <i>Monitoring</i> Industri	12
2.2.2 Protokol Komunikasi TCP/IP dan Modbus RTU	13
2.2.3 Penyimpanan Data dengan InfluxDB dan Visualisasi <i>Real-Time</i> melalui Grafana	13
2.2.4 Pengembangan Aplikasi Desktop dengan Qt	13
2.2.5 Teknologi Blockchain dalam Sistem Industri	13
2.2.6 Smart Contract dan Web3	14
BAB 3 METODOLOGI	15
3.1 Flowchart Sistem	15
3.2 Pembuatan Program	15
3.2.1 Pembuatan Program Client Sensor	16
3.2.2 Pembuatan Program Server	16
3.2.3 Pembuatan GUI Desktop	16
3.2.4 Pembuatan Web3 Dashboard	16
3.2.5 Dependensi dan Konfigurasi	17
3.3 Pengujian Sistem	17

3.3.1	Metodologi Pengujian	17
3.3.2	Metode Pengujian	17
BAB 4	Hasil dan Pembahasan	19
4.1	Hasil	19
4.1.1	Tampilan Data <i>Explorer InfluxDB</i>	19
4.1.2	Tampilan <i>Dashboard Grafana</i>	19
4.1.3	Tampilan Web3 Sensor <i>Dashboard</i>	20
4.1.4	Tampilan <i>Blockchain</i>	20
4.1.5	Tampilan Eksekusi Perintah <i>Cargo Run</i>	21
4.1.6	Tampilan <i>Smartcontract</i> Pada Saat <i>Cargo Run</i>	22
4.1.7	Tampilan Perintah <i>Ganache</i>	22
4.1.8	Tampilan Pembuatan Server Web	23
4.1.9	Tampilan PyQT	23
4.2	Pembahasan	24
BAB 5	Kesimpulan dan Saran	25
5.1	Kesimpulan	25
5.2	Saran	25
DAFTAR PUSTAKA		26
LAMPIRAN		27
BIODATA PENULIS		43

DAFTAR GAMBAR

Gambar 3.1 Flowchart Sistem	15
Gambar 4.1 Data Explorer InfluxDB	19
Gambar 4.2 Dashboard Grafana	19
Gambar 4.3 Web3 Sensor Dashboard	20
Gambar 4.4 Blockchain Container Purchase Dashboard	20
Gambar 4.5 Tabel data sensor	21
Gambar 4.6 Eksekusi Perintah Cargo Run	21
Gambar 4.7 Smartcontract Pada Saat Cargo Run.....	22
Gambar 4.8 Perintah Ganache	22
Gambar 4.9 Pembuatan Server Web	23
Gambar 4.10 Tampilan PyQt	23

DAFTAR TABEL

Tabel 2.1 State of The Art	12
---	-----------

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Industri minyak dan gas bumi (migas) merupakan sektor vital dalam mendukung pertumbuhan ekonomi global, memenuhi kebutuhan energi, dan menjaga kestabilan geopolitik. Rantai pasok industri migas terbagi menjadi tiga segmen utama, yakni sektor hulu (*upstream*), tengah (*midstream*), dan hilir (*downstream*). Sektor *midstream*, yang berfokus pada transportasi dan penyimpanan minyak mentah maupun gas dari titik produksi ke kilang atau fasilitas ekspor, memainkan peran strategis sebagai penghubung antara produksi dan konsumsi energi (Lu et al., 2019).

Namun, sektor ini kerap menghadapi tantangan serius, terutama dalam hal pengelolaan informasi dan validitas transaksi antar pelaku industri. Penggunaan sistem logistik tradisional berbasis dokumentasi fisik atau terpusat telah menyebabkan berbagai permasalahan, seperti duplikasi transaksi, kontrak ganda antar pihak, dan kesalahan pencatatan yang dapat memicu keterlambatan distribusi serta pemborosan biaya operasional (Haque, Hasan, & Zihad, 2021). Situasi ini diperparah dengan tidak adanya sistem tunggal yang menjamin transparansi dan akurasi data secara menyeluruh dalam ekosistem *midstream* migas.

Sebagai solusi potensial, teknologi *blockchain* hadir dengan menawarkan sistem pencatatan transaksi yang bersifat terdesentralisasi, transparan, dan tidak dapat diubah (*immutable*). *Blockchain* menyediakan *distributed ledger* yang memungkinkan seluruh pihak dalam rantai pasok untuk mengakses informasi yang sama secara *real-time*, sehingga meminimalkan risiko manipulasi dan duplikasi data (Lu, Huang, Azimi, & Guo, 2019; Wang et al., 2019).

Lebih lanjut, fitur *smart contract* dalam *blockchain* dapat mengotomatiskan eksekusi perjanjian berdasarkan parameter yang telah ditentukan, tanpa keterlibatan pihak ketiga. Hal ini sangat relevan dalam konteks kontrak logistik migas yang sering melibatkan berbagai entitas dan berisiko tumpang tindih. *Smart contract* memungkinkan eksekusi transaksi yang cepat, aman, dan bebas kesalahan, sekaligus menjaga keabsahan kontrak secara otomatis (Haque et al., 2021; Al-Fuqaha et al., 2015).

Penelitian sebelumnya oleh Haque et al. (2021) memperkenalkan model *SmartOil* sebagai sistem *blockchain* dan *smart contract* untuk mengelola rantai pasok minyak secara efisien. Sementara itu, studi oleh Lu et al. (2019) menunjukkan bahwa adopsi *blockchain* di sektor migas dapat menurunkan risiko korupsi data serta meningkatkan akurasi dokumentasi dalam proses distribusi. Selain itu, Wang et al. (2019) dalam ulasannya tentang *blockchain* dan IoT juga menekankan bagaimana integrasi teknologi ini dapat meningkatkan keamanan data serta efisiensi sistem.

Oleh karena itu, penerapan teknologi *blockchain* dan *smart contract* dalam sektor *midstream* migas menjadi sangat relevan dalam upaya untuk mengurangi duplikasi transaksi dan meningkatkan akurasi pencatatan data logistik. Penelitian ini diharapkan mampu memberikan kontribusi nyata dalam modernisasi infrastruktur informasi logistik migas dan mendorong praktik distribusi yang lebih efisien, transparan, dan dapat dipertanggungjawabkan.

1.2 Rumusan Masalah

Berdasarkan latar belakang dan deskripsi proyek yang diberikan, maka rumusan masalah dalam penelitian ini dapat dirumuskan sebagai berikut:

1. Bagaimana merancang sistem *monitoring* suhu dan kelembaban berbasis sensor industri yang dapat diprogram menggunakan bahasa *Embedded Rust* dan diintegrasikan dengan protokol komunikasi TCP dan Modbus RTU untuk kebutuhan sektor *midstream* migas?
2. Bagaimana cara menyimpan dan menampilkan data sensor secara *real-time* melalui InfluxDB dan Grafana, serta mengembangkan antarmuka *desktop* menggunakan *framework* Qt yang mendukung pemantauan distribusi migas secara efisien?
3. Bagaimana mengimplementasikan teknologi *blockchain* dan Web3 (*smart contract* dan DApp) ke dalam sistem *monitoring* untuk menjamin transparansi, keutuhan, dan akuntabilitas data dalam rantai pasok distribusi migas?
4. Bagaimana efektivitas penerapan sistem terintegrasi ini dalam mengurangi risiko duplikasi transaksi, kesalahan pencatatan, serta meningkatkan kepercayaan antar pemangku kepentingan di sektor *midstream* migas?

1.3 Batasan Masalah

Agar penelitian ini terfokus dan terarah, maka ditetapkan beberapa batasan masalah sebagai berikut:

1. Sistem hanya akan memantau dua parameter lingkungan, yaitu suhu dan kelembaban, menggunakan sensor industri yang sesuai dengan lingkungan distribusi sektor *midstream* migas.
2. Komunikasi data sensor dibatasi hanya pada dua protokol, yaitu TCP/IP dan Modbus RTU, sebagai perwakilan komunikasi modern dan industri konvensional.
3. Data yang diterima dari sensor hanya akan disimpan di InfluxDB dan divisualisasikan secara *real-time* menggunakan Grafana.
4. Studi kasus sistem ini difokuskan pada rantai pasok sektor *midstream* migas, khususnya dalam konteks pemantauan distribusi bahan bakar atau minyak mentah, tanpa mencakup sektor hulu atau hilir secara spesifik.

1.4 Tujuan

Tujuan dari perancangan sistem ini yaitu sebagai berikut

1. Merancang dan mengembangkan sistem *monitoring* suhu dan kelembaban dengan program menggunakan bahasa *Embedded Rust*, serta mampu berkomunikasi melalui protokol TCP/IP dan Modbus RTU, guna memenuhi kebutuhan *monitoring* lingkungan pada rantai distribusi sektor *midstream* migas.
2. Mengimplementasikan sistem penyimpanan dan visualisasi data sensor secara *real-time* dengan memanfaatkan InfluxDB sebagai *time-series database* dan Grafana sebagai *platform* visualisasi, serta mengembangkan aplikasi *desktop* berbasis Qt sebagai antarmuka pengguna.
3. Menerapkan teknologi *blockchain* dan Web3, khususnya melalui *smart contract* dan *decentralized application* (DApp), untuk mencatat dan memverifikasi data suhu dan kelembaban secara terdesentralisasi, transparan, dan tidak dapat diubah.
4. Mengevaluasi efektivitas sistem terintegrasi dalam mengurangi risiko duplikasi transaksi, meningkatkan keakuratan data distribusi, serta membangun sistem *monitoring* yang transparan dan akuntabel bagi berbagai pihak dalam rantai pasok sektor *midstream* migas.

1.5 Manfaat

Manfaat yang diharapkan dengan adanya perancangan sistem ini yaitu

1. Memberikan contoh implementasi multidisipliner antara teknologi sensor industri, jaringan komunikasi, *database time-series*, visualisasi data, dan *blockchain*.
2. Menghasilkan sistem *monitoring* suhu dan kelembaban yang *real-time* dan akurat, yang dapat digunakan dalam kondisi lingkungan industri yang kompleks seperti pada proses distribusi *midstream* migas.
3. Memberikan solusi otomatisasi pencatatan dan pelaporan data sensor melalui *smart contract* berbasis *blockchain*, yang sulit dimanipulasi dan mudah diverifikasi antar pihak.

BAB 2 TINJAUAN PUSTAKA

2.1 Hasil Penelitian Terdahulu

Tabel 2.1 *State of The Art*

Judul, penulis & Tahun	Metode	Hasil
<i>SmartOil: Blockchain and Smart Contract-based Oil Supply Chain Management</i> (Haque, Hasan, & Zihad 2021)	Pengembangan arsitektur sistem rantai pasok minyak menggunakan <i>blockchain</i> dan <i>smart contract</i>	Sistem <i>SmartOil</i> mampu mencatat dan memverifikasi transaksi distribusi secara otomatis, meningkatkan transparansi dan mengurangi duplikasi data.
<i>Blockchain Technology in the Oil and Gas Industry: A Review of Applications, Opportunities, Challenges, and Risks</i> (Lu, Huang, Azimi, & Guo 2019)	Tinjauan pustaka mendalam tentang penerapan <i>blockchain</i> di industri migas	<i>Blockchain</i> sangat potensial dalam meningkatkan efisiensi, mengurangi risiko manipulasi data, dan memperkuat keamanan logistik migas.
<i>Survey on Blockchain for Internet of Things</i> (Wang et al. 2019)	Studi komprehensif integrasi <i>blockchain</i> dengan IoT	Integrasi <i>blockchain-IoT</i> meningkatkan keamanan, otentikasi data sensor, dan sangat cocok untuk sistem <i>monitoring</i> industri.
<i>Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications</i> (Al-Fuqaha et al. 2015)	Survei tentang teknologi dan protokol pendukung IoT	Memberikan dasar teknis penting untuk pengembangan sistem komunikasi berbasis TCP/IP dan Modbus RTU dalam pemantauan sensor.
<i>IoT in Social, Mobile, Analytics and Cloud (ISMAC 2020 Proceedings)</i> (Sangeetha & Shunmugan 2020)	Kompilasi riset IoT dan cloud dalam konferensi	Mendukung penerapan platform <i>time-series database</i> (InfluxDB) dan visualisasi data <i>real-time</i> dalam sistem industri modern
<i>RAIN: A Bio-Inspired Communication and Data Storage Infrastructure</i> (Monti & Rasmussen 2017)	Desain sistem penyimpanan dan komunikasi terdistribusi	Memberikan inspirasi dalam arsitektur penyimpanan data berbasis desentralisasi dan efisiensi komunikasi.

2.2 Dasar Teori

2.2.1 Sistem *Monitoring* Industri

Sistem *monitoring* industri merupakan komponen vital dalam pengawasan kondisi proses, baik secara lokal maupun terdistribusi. Dalam konteks distribusi migas sektor *midstream*, parameter lingkungan seperti suhu dan kelembaban menjadi penting karena berkaitan langsung dengan keamanan, kualitas, dan efisiensi transportasi bahan bakar. Sensor industri yang digunakan harus memiliki tingkat akurasi tinggi, ketahanan terhadap lingkungan ekstrim, dan kemampuan integrasi dengan sistem digital.

Dalam penelitian ini, sensor suhu dan kelembaban digunakan untuk memantau kondisi lingkungan di sekitar tangki atau pipa distribusi migas. Data sensor dibaca oleh perangkat *embedded system* yang diprogram menggunakan bahasa *Embedded Rust*. *Embedded Rust* dipilih karena kemampuannya dalam menghasilkan program *bare-metal* yang aman, efisien memori, dan bebas dari *data race*, sangat cocok untuk lingkungan industri yang menuntut ketebalahan tinggi (Al-Fuqaha et al., 2015).

2.2.2 Protokol Komunikasi TCP/IP dan Modbus RTU

Dalam sistem otomasi industri, komunikasi antar perangkat merupakan hal yang krusial. Dua protokol utama yang digunakan dalam penelitian ini adalah TCP/IP dan Modbus RTU. Modbus RTU merupakan protokol komunikasi serial yang banyak digunakan di lingkungan industri karena kesederhanaannya, keandalan tinggi, dan kompatibilitas dengan berbagai perangkat seperti PLC dan HMI.

Di sisi lain, TCP/IP digunakan untuk memungkinkan transmisi data ke *server* melalui jaringan *Ethernet* atau Wi-Fi. Protokol ini bersifat *connection-oriented* dan mendukung pengiriman data secara *real-time* dengan latensi rendah, sehingga ideal untuk menghubungkan sistem *monitoring* ke *server* pusat atau *cloud platform* (Al-Fuqaha et al., 2015; Wang et al., 2019).

2.2.3 Penyimpanan Data dengan InfluxDB dan Visualisasi *Real-Time* melalui Grafana

Data suhu dan kelembaban yang dikumpulkan perlu disimpan secara efisien untuk analisis jangka panjang dan pemantauan historis. Untuk itu digunakan InfluxDB, yaitu *time-series database* yang dirancang untuk data dengan cap waktu (*timestamped*), seperti data sensor industri. InfluxDB mendukung penulisan dan pembacaan data dalam jumlah besar dengan latensi rendah serta kemampuan *downsampling*.

Visualisasi data dilakukan menggunakan Grafana, sebuah *platform open-source* yang sangat populer dalam dunia industri untuk membuat *dashboard* interaktif dan *real-time*. Grafana mendukung integrasi langsung dengan InfluxDB dan memungkinkan pengguna melihat pola fluktuasi suhu dan kelembaban dari waktu ke waktu dalam bentuk grafik yang informatif (Sangeetha & Shunmugan, 2020).

2.2.4 Pengembangan Aplikasi Desktop dengan Qt

Untuk memberikan antarmuka pengguna yang intuitif dan fleksibel, aplikasi desktop dirancang menggunakan Qt *Framework*. Qt memungkinkan pembuatan GUI modern yang kompatibel dengan berbagai sistem operasi, dan mendukung integrasi langsung dengan komunikasi TCP/IP serta pengolahan data dari InfluxDB. Dalam proyek ini, aplikasi berbasis Qt digunakan untuk menampilkan nilai suhu dan kelembaban secara real-time kepada operator di lingkungan industri.

2.2.5 Teknologi Blockchain dalam Sistem Industri

Blockchain adalah teknologi *ledger* terdistribusi yang memungkinkan pencatatan transaksi secara permanen, transparan, dan tidak dapat diubah. Dalam sistem *monitoring* distribusi migas, *blockchain* dapat digunakan untuk menyimpan data sensor secara terdesentralisasi sehingga seluruh pihak dalam rantai pasok operator, transportir, dan regulator dapat mengakses informasi yang sama tanpa manipulasi.

Penerapan *blockchain* dalam sektor minyak dan gas telah dikaji oleh Lu et al. (2019), yang menyatakan bahwa teknologi ini dapat membantu mengurangi biaya administrasi, meningkatkan efisiensi logistik, dan memperkuat kepercayaan antar pihak. Dengan mencatat

parameter lingkungan seperti suhu dan kelembaban ke dalam *blockchain*, perusahaan dapat memastikan bahwa distribusi dilakukan dalam kondisi sesuai standar keselamatan.

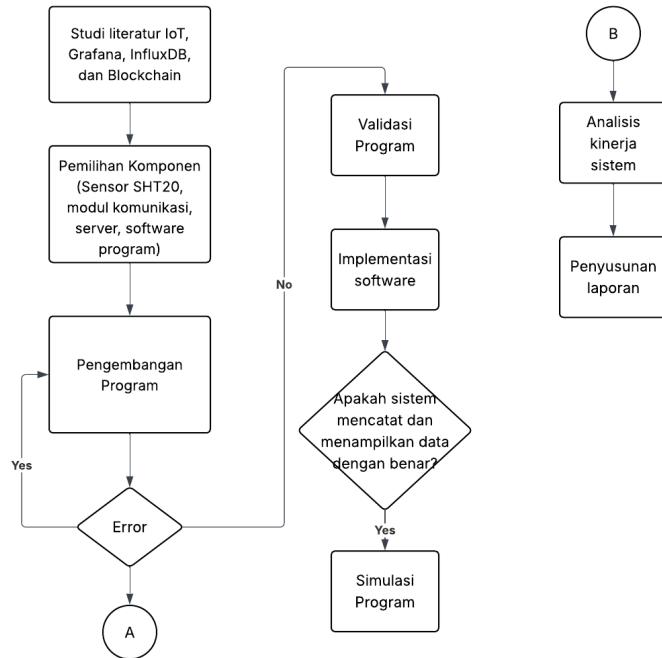
2.2.6 Smart Contract dan Web3

Smart contract adalah bagian dari ekosistem *blockchain* yang memungkinkan eksekusi kontrak digital secara otomatis berdasarkan kondisi tertentu. Dalam penelitian ini, *smart contract* digunakan untuk mencatat dan memverifikasi data sensor secara otomatis. Misalnya, jika suhu melebihi ambang batas tertentu, maka sistem secara otomatis men-trigger pencatatan ke *blockchain* disertai alarm atau notifikasi kepada pengguna. Ini mengurangi intervensi manual dan meminimalisir human error (Haque, Hasan, & Zihad, 2021).

Web3 mengacu pada evolusi internet terdesentralisasi, di mana data dan aplikasi tidak lagi dikendalikan oleh satu entitas, tetapi oleh jaringan *peer-to-peer*. Melalui *decentralized applications* (DApps), pengguna dapat mengakses data sensor yang tercatat di *blockchain* dengan transparansi penuh. Sistem ini sangat sesuai untuk sektor industri seperti migas yang melibatkan banyak pemangku kepentingan dengan kepentingan dan otorisasi berbeda (Monti & Rasmussen, 2017).

BAB 3 METODOLOGI

3.1 Flowchart Sistem



Gambar 3.1 Flowchart Sistem

Gambar 3.1 merupakan diagram alir pengembangan sistem *monitoring* suhu dan kelembaban berbasis *Internet of Things* (IoT) yang dilengkapi dengan integrasi teknologi InfluxDB, Grafana, dan blockchain. Proses pengembangan dimulai dari studi literatur yang mencakup pemahaman mengenai arsitektur sistem IoT, protokol komunikasi, serta penggunaan *time-series database* dan sistem pencatatan desentralisasi. Setelah itu, dilakukan pemilihan komponen utama seperti sensor suhu dan kelembaban SHT20, modul komunikasi RS485/USB, *server lokal/cloud*, serta *platform* pemrograman seperti Rust, Grafana, dan Web3.

Tahap berikutnya adalah pengembangan program sistem *monitoring*. Program diuji untuk memastikan tidak terdapat kesalahan (*error*), dan bila ditemukan, dilakukan proses *debugging* hingga sistem berjalan dengan baik. Setelah program tervalidasi, sistem diimplementasikan secara penuh, kemudian diuji untuk memastikan bahwa data dapat dicatat dan divisualisasikan dengan benar. Apabila sistem dapat menampilkan data secara akurat, maka dilanjutkan ke tahap simulasi program untuk mengevaluasi kinerja fungsional sistem. Tahapan akhir dari proses ini adalah analisis kinerja sistem serta penyusunan laporan sebagai dokumentasi hasil pengembangan secara menyeluruh.

3.2 Pembuatan Program

Pembuatan program dalam proyek ini terdiri dari tiga bagian utama, yaitu:

1. Program *Client Sensor (Embedded Rust)*: membaca data suhu dan kelembaban melalui Modbus RTU.
2. Program Server (Rust): menerima data via TCP, menyimpan ke InfluxDB, dan mencatat ke Blockchain.

3. Antarmuka Visualisasi: terdiri dari GUI desktop berbasis Python dan dashboard Web3 berbasis HTML-JS.

Setiap bagian dirancang untuk berjalan secara sinkron dan saling terhubung melalui protokol terbuka, seperti TCP, HTTP, dan Web3 RPC.

3.2.1 Pembuatan Program Client Sensor

Program client dikembangkan menggunakan bahasa Rust dengan pustaka tokio-modbus untuk komunikasi Modbus RTU melalui port serial (/dev/ttyUSB0). Sensor SHT20 dikonfigurasi untuk mengirimkan dua register yang masing-masing merepresentasikan suhu dan kelembaban. Data dibaca setiap 5 detik, dikonversi dari satuan *raw register* menjadi suhu dalam °C dan kelembaban dalam %RH, lalu dikemas dalam format JSON.

Data yang sudah terstruktur dikirim ke server melalui TCP *socket* menggunakan tokio::net::TcpStream. Jika koneksi berhasil, JSON dikirim disertai *newline* (\n) sebagai *delimiter*. Sistem mampu menangani *connection failure* dan *data loss* dengan *retry* otomatis setiap siklus pembacaan.

3.2.2 Pembuatan Program Server

Server dikembangkan dengan *Rust asynchronous* menggunakan tokio dan reqwest. Ia menerima koneksi masuk pada port 9000, membaca data sensor, dan melakukan dua proses:

1. Penyimpanan InfluxDB: Data dikonversi ke format *Line Protocol*, lalu dikirim melalui HTTP POST ke *endpoint* API InfluxDB v2.
2. Pencatatan ke *Blockchain*: *Smart contract* SensorStorage.sol di-deploy secara otomatis menggunakan ethers-rs. Metode storeData() dipanggil untuk menyimpan data sensor ke dalam blockchain Ethereum lokal (localhost:8545) melalui Web3 JSON-RPC. Program ini memanfaatkan struktur JSON deserialisasi otomatis (serde) dan parsing timestamp RFC3339 ke epoch UNIX untuk integrasi dengan time-series database.

3.2.3 Pembuatan GUI Desktop

GUI desktop dikembangkan menggunakan Python dengan Tkinter dan matplotlib. Program ini memanggil data dari InfluxDB menggunakan *query* FLUX melalui *endpoint* /api/v2/query. Data suhu dan kelembaban ditampilkan dalam format numerik dan grafik waktu nyata (*real-time chart*).

Fitur utama:

- Tampilan suhu & kelembaban secara langsung.
- Grafik historis 50 titik data terakhir.
- *Input* manual untuk melihat data historis dengan rentang waktu tertentu.

Proses update dilakukan dalam *thread* paralel untuk memastikan GUI tetap responsif selama pengambilan data dari InfluxDB.

3.2.4 Pembuatan Web3 Dashboard

Antarmuka Web3 dikembangkan menggunakan HTML, Chart.js, dan Ethers.js. *Dashboard* ini menampilkan:

- Tabel log data sensor dari *blockchain*.
- Grafik suhu dan kelembaban dari *event blockchain*.
- Simulasi pemesanan *container* dengan validasi suhu & kelembaban sesuai kriteria.

Dashboard terhubung langsung ke smart contract SensorStorage menggunakan alamat kontrak dan ABI yang telah di-fetch. Fitur tambahan seperti QR simulasi pembelian ditampilkan setelah data sesuai ditemukan di blockchain.

3.2.5 Dependensi dan Konfigurasi

Seluruh dependensi untuk proyek ini didefinisikan dalam file Cargo.toml, package.json, dan requirements.txt. Untuk Rust, digunakan pustaka berikut:

- tokio, tokio-serial, tokio-modbus (I/O dan komunikasi)
- serde, serde_json (serialisasi JSON)
- reqwest (HTTP client untuk InfluxDB)
- ethers (integrasi Web3/Ethereum)
- chrono (manajemen waktu)

3.3 Pengujian Sistem

Pengujian sistem bertujuan untuk memastikan bahwa setiap komponen dalam sistem *monitoring* suhu dan kelembaban dapat berfungsi dengan baik sesuai rancangan, mulai dari pembacaan sensor, komunikasi data, penyimpanan di *database*, visualisasi data, hingga pencatatan ke dalam *blockchain*. Pengujian ini juga dilakukan untuk mengevaluasi performa sistem dalam kondisi operasional nyata di lingkungan *midstream* migas.

Pengujian dilakukan secara berjenjang dan terpisah berdasarkan blok fungsi sistem, dengan metode sebagai berikut:

3.3.1 Metodologi Pengujian

Uji Fungsional Sensor dan Komunikasi: memastikan data dari sensor SHT20 dapat terbaca dan dikirim menggunakan Modbus RTU.

1. Uji TCP *Transmission*: menguji keberhasilan pengiriman data dari *client* ke server menggunakan protokol TCP/IP.
2. Uji Penyimpanan dan Visualisasi Data: mengecek apakah data berhasil disimpan ke InfluxDB dan divisualisasikan melalui Grafana dan PyQt GUI.
3. Uji Pencatatan ke *Blockchain*: menguji apakah data sensor berhasil tercatat ke dalam *smart contract* Ethereum dan dapat diakses melalui DApp berbasis Web3.

3.3.2 Metode Pengujian

Pengujian dilakukan berdasarkan tahapan dan komponen berikut:

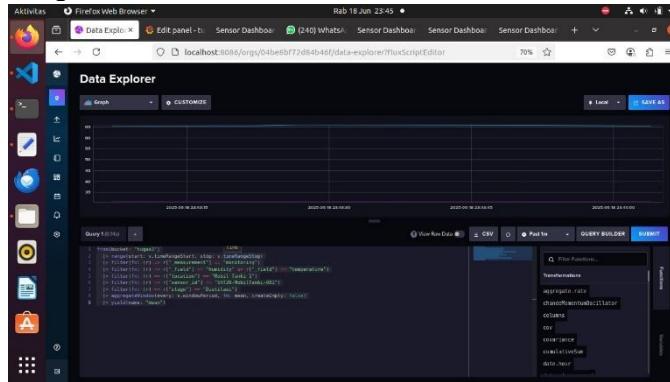
Komponen	Tujuan	Metode Pengujian
Client Sensor (Rust)	Memastikan sensor membaca suhu & kelembaban	Dibaca melalui Modbus RTU (RS485)
Komunikasi TCP	Memastikan data terkirim dari client ke server	TCP stream ditest setiap 5 detik
Server (Rust)	Parsing JSON, kirim ke InfluxDB & Smart Contract	Monitor via log terminal & Grafana
InfluxDB	Menyimpan data time-series	Dicek via dashboard & query FLUX
Ethereum Smart Contract	Mencatat log sensor	Dicek via hash transaksi dan DApp

PyQt GUI	Visualisasi real-time & historis	Dibandingkan dengan data aslinya
Web3 Dashboard	Tampilkan data & simulasi	Verifikasi tabel & chart sensor

BAB 4 Hasil dan Pembahasan

4.1 Hasil

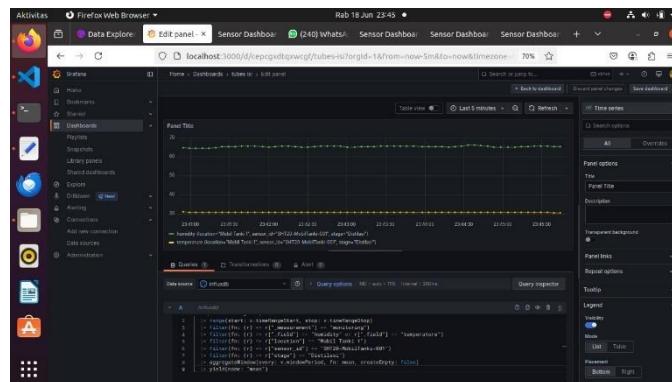
4.1.1 Tampilan Data Explorer InfluxDB



Gambar 4.1 Data Explorer InfluxDB

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.

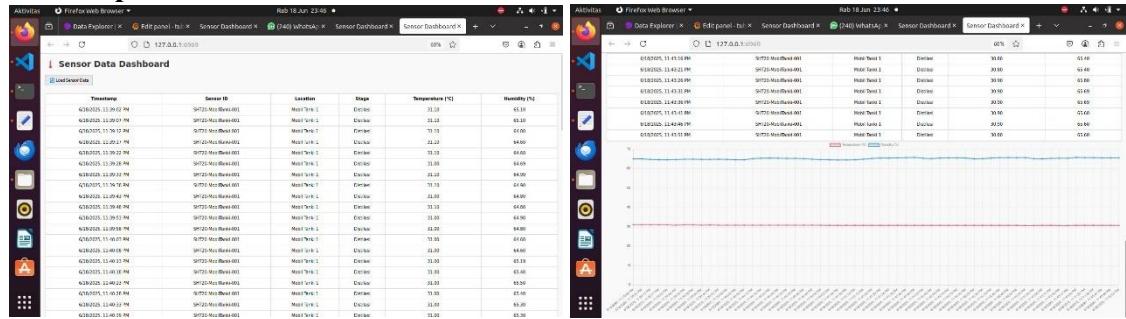
4.1.2 Tampilan Dashboard Grafana



Gambar 4.2 Dashboard Grafana

Gambar tersebut menunjukkan tampilan Grafana *Dashboard* yang digunakan untuk memvisualisasikan data suhu dan kelembaban secara *real-time*. Pada panel grafik di bagian atas, terlihat dua garis data yang menunjukkan nilai suhu dan kelembaban yang stabil selama rentang waktu lima menit terakhir. Di bagian bawah terdapat *editor query* yang menggunakan untuk mengambil data dari *bucket* "tubes", dengan *filter field* "temperature" dan "humidity". Tampilan ini membantu pengguna memantau kondisi lingkungan secara langsung dan mudah dipahami dalam bentuk grafik.

4.1.3 Tampilan Web3 Sensor Dashboard

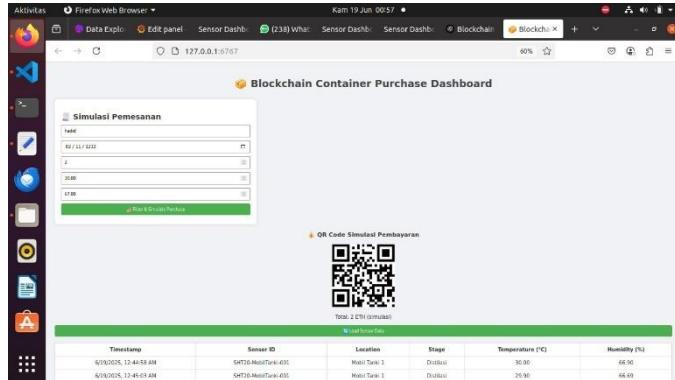


Gambar 4.3 Web3 Sensor Dashboard

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 sebelah 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.

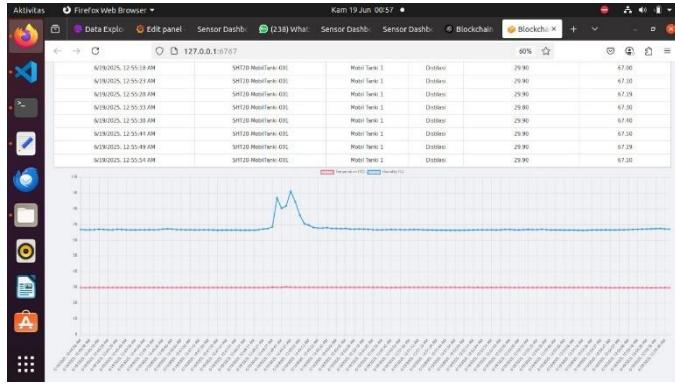
4.1.4 Tampilan Blockchain



Gambar 4.4 Blockchain Container Purchase Dashboard

Gambar diatas merupakan tampilan dari *Blockchain Container Purchase Dashboard*, yaitu antarmuka berbasis web yang dirancang untuk mensimulasikan pemesanan kontainer berdasarkan kondisi lingkungan (suhu dan kelembaban) yang dicatat oleh sensor dan disimpan ke dalam *blockchain*.

Pada *dashboard* terdapat *form* simulasi pemesanan, di mana pengguna dapat memasukkan nama, tanggal, jumlah container, serta suhu dan kelembaban yang diinginkan. Setelah tombol "*Filter & Simulate Purchase*" ditekan, sistem akan mencari data sensor dari *blockchain* yang sesuai dengan kriteria tersebut. Jika data cocok ditemukan, akan muncul QR Code simulasi pembayaran dan informasi jumlah total ETH yang harus dibayar sebagai bagian dari simulasi transaksi.



Gambar 4.5 Tabel data sensor

Gambar diatas menunjukkan tampilan tabel data sensor yang memuat *timestamp*, ID sensor, lokasi, tahap proses, suhu, dan kelembaban dari hasil pembacaan sensor. Seluruh data ini berasal dari event `DataStored` dalam *smart contract* Ethereum. Di bagian paling bawah, terdapat grafik suhu dan kelembaban yang divisualisasikan secara *real-time*, memudahkan pengguna memantau kondisi aktual dan mendeteksi perubahan yang signifikan seperti lonjakan suhu. Tampilan tersebut tidak hanya menyajikan informasi *monitoring* lingkungan secara transparan, tetapi juga mengilustrasikan integrasi antara data IoT dan sistem transaksi berbasis blockchain dalam konteks logistik dan pengiriman.

4.1.5 Tampilan Eksekusi Perintah *Cargo Run*

Gambar 4.6 Eksekusi Perintah *Cargo Run*

Tampilan tersebut menunjukkan hasil eksekusi perintah `cargo run` pada program sensor berbasis Rust. Program ini mencoba membaca data suhu dan kelembaban dari sensor melalui port serial `/dev/ttyUSB0` menggunakan protokol Modbus RTU, kemudian mengirimkannya ke server melalui koneksi TCP. Namun, muncul dua jenis pesan error, yaitu "*No such file or directory*" yang menandakan port sensor belum terdeteksi atau salah, dan "*Connection refused*" yang menunjukkan bahwa program gagal terhubung ke server TCP karena belum dijalankan atau alamatnya tidak sesuai. Meskipun demikian, program sempat berhasil membaca dan mencetak nilai suhu dan kelembaban, artinya sebagian fungsi berjalan dengan baik namun memerlukan perbaikan konektivitas untuk berfungsi secara penuh.

4.1.6 Tampilan Smartcontract Pada Saat Cargo Run

```

greista@greista-VirtualBox:~/Unduhan/Tubes ISI ke1/Tubes ISI/Server$ cargo run
Kam 19 Jun 02:16 • greista@greista-VirtualBox:~/Unduhan/Tubes ISI ke1/Tubes ISI/Server
greista@greista-VirtualBox:~/Unduhan/Tubes ISI ke1/Tubes ISI/Server$ cd Server
greista@greista-VirtualBox:~/Unduhan/Tubes ISI ke1/Tubes ISI/Server$ cargo run
Compiling sensor_server blockchain v0.1.0 (/home/greista/Unduhan/Tubes ISI ke1/Tubes ISI/Server)
    Finished dev profile [unoptimized + debuginfo] target(s) in 8.48s
     Running target/debug/sensor_server.blockchain
Error: (code: -32003, message: insufficient funds for gas * price = value, data: None)
greista@greista-VirtualBox:~/Unduhan/Tubes ISI ke1/Tubes ISI/Server$ cargo run
Compiling sensor_server blockchain v0.1.0 (/home/greista/Unduhan/Tubes ISI ke1/Tubes ISI/Server)
    Finished dev profile [unoptimized + debuginfo] target(s) in 8.48s
     Running target/debug/sensor_server.blockchain
Error: (code: -32003, message: insufficient funds for gas * price = value, data: None)
greista@greista-VirtualBox:~/Unduhan/Tubes ISI ke1/Tubes ISI/Server$ cargo run
Compiling sensor_server blockchain v0.1.0 (/home/greista/Unduhan/Tubes ISI ke1/Tubes ISI/Server)
    Finished dev profile [unoptimized + debuginfo] target(s) in 8.48s
     Running target/debug/sensor_server.blockchain
Smart contract deployed at 0x6fecd9887c7bacc5d0e0065057dec0b367570
TCP Server listening on port 9000...
Mobile Tanki 1' received data: SensorData { timestamp: "2025-06-18T18:48:04.475993384+00:00", sensor_id: "SH720-MobilTanki-001", location: "Mobile Tanki 1", process_stage: "Distlast", temperature_celcius: 29.4, humidity_percent: 66.2 }
Mobile Tanki 1' received data: SensorData { timestamp: "2025-06-18T18:48:09.516669758+00:00", sensor_id: "SH720-MobilTanki-001", location: "Mobile Tanki 1", process_stage: "Distlast", temperature_celcius: 29.4, humidity_percent: 67.8 }
InfluxDB: data written
Ethereum: tx sent: PendingTransaction { tx hash: 0x923a61744310e1e7979dc3b1fs1961b4b6d5387afad20f3fd4a15glef407a, confirmation: 5! 1, state: PendingTxState { state: "InitialDelay" } }
InfluxDB: data written
Ethereum: tx sent: PendingTransaction { tx hash: 0x1ff25eef9fa2421881407927bf37c2cc5194394911fcf9a81b97ce408594ab7, confirmation: 5! 1, state: PendingTxState { state: "InitialDelay" } }
InfluxDB: data written
Ethereum: tx sent: PendingTransaction { tx hash: 0xd9f21ca944170482fb9eb2ef2a8d7cc17f76a66662372fbcb8e396430ab, confirmation: 5! 1, state: PendingTxState { state: "InitialDelay" } }
New connection from 127.0.0.1:53535
InfluxDB: data written
Ethereum: tx sent: PendingTransaction { tx hash: 0x205-06-18T18:48:14.6157876674+00:00", sensor_id: "SH720-MobilTanki-001", location: "Mobile Tanki 1", process_stage: "Distlast", temperature_celcius: 29.4, humidity_percent: 67.6 }
InfluxDB: data written
Ethereum: tx sent: PendingTransaction { tx hash: 0x0df21ca944170482fb9eb2ef2a8d7cc17f76a66662372fbcb8e396430ab, confirmation: 5! 1, state: PendingTxState { state: "InitialDelay" } }

```

Gambar 4.7 Smartcontract Pada Saat Cargo Run

Tampilan tersebut menunjukkan proses saat program server berhasil dijalankan menggunakan perintah `cargo run`, yang merupakan bagian penting dari sistem *monitoring* suhu dan kelembaban berbasis Rust, InfluxDB, dan *blockchain* Ethereum. Pada terminal, terlihat bahwa *smart contract* berhasil *dideploy* ke jaringan Ethereum lokal dengan ditandai pesan "*Smart contract deployed at...*". Setelah itu, server membuka koneksi TCP pada port 9000 dan menerima koneksi dari *client* sensor. Ketika data sensor diterima dalam format JSON, server mencatat bahwa data berhasil diterima ("Received sensor data") dan langsung diproses ke dua jalur: pertama, disimpan ke InfluxDB menggunakan format *Line Protocol* dan berhasil tercatat ("InfluxDB: data written"); kedua, data dikirim ke *blockchain* melalui metode `storeData` dalam *smart contract* dan ditandai dengan transaksi yang sukses ("Ethereum: tx sent"). Tampilan ini menandakan bahwa seluruh alur komunikasi antara sensor, server, *database*, dan *blockchain* berjalan dengan lancar dan tanpa *error*.

4.1.7 Tampilan Perintah Ganache

```

greista@greista-VirtualBox:~/Unduhan/Tubes ISI ke1/Tubes ISI$ ganache
Kam 19 Jun 02:18 • greista@greista-VirtualBox:~/Unduhan/Tubes ISI ke1/Tubes ISI
greista@greista-VirtualBox:~/Unduhan/Tubes ISI ke1/Tubes ISI$ ganache
This version of ganache is not compatible with your Node.js build:
Your version of Node.js is incompatible with this version of ganache.
Error: Cannot find module './uws/lib/node_uws_x64_127.node'
Require stack:
- /home/greista/_myver/versions/node/v21.15.1/lib/node_modules/ganache/node_modules/@trufflesuite/uws-js-unofficial/src/uws.js
- /home/greista/_myver/versions/node/v21.15.1/lib/node_modules/ganache/node_modules/@trufflesuite/uws-js-unofficial/src/index.js
Falling back to a NodeJS implementation; performance may be degraded.

ganache v7.9.2 (@ganache/clt: 0.10.2, @ganache/core: 0.10.2)
Starting RPC server

Available Accounts
=====
(0) 0x39239571d99fb2e9081908cA9a971f081B8C (1000 ETH)
(1) 0x93203571d99fb2e9081908cA9a971f081B8C (1000 ETH)
(2) 0xb9894e48BC1330013D01146221D4b72a7f44155 (1000 ETH)
(3) 0x1f203571d99fb2e9081908cA9a971f081B8C (1000 ETH)
(4) 0x91203571d99fb2e9081908cA9a971f081B8C (1000 ETH)
(5) 0x3f9c9e903a176201375530C810cc2726a593 (1000 ETH)
(6) 0x1f203571d99fb2e9081908cA9a971f081B8C (1000 ETH)
(7) 0x520bb58cb562537aa605b4d1c51760C7984 (1000 ETH)
(8) 0xd08d335feBA50C757FB5B801fC4488A009 (1000 ETH)
(9) 0x204f18a90f72c2892e208c909320598c3300b (1000 ETH)

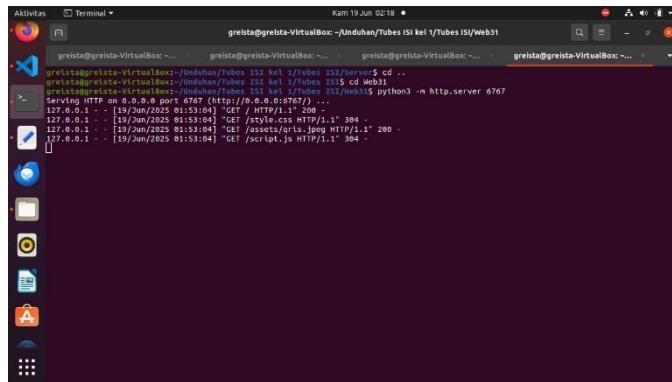
Private Keys
=====
(0) 0xdeadbeef000000000000000000000000000000000000000000000000000000000000000
(1) 0xf0fb400dfccca9a0821a5d6ceca7a07be9e310bbe0fe0fa962d92e40a7c54d
(2) 0x0edcf2c2010b1b0602e08e001fd7c104e1b080be0bf302e0ad0ffcc1d821
(3) 0x0edcf2c2010b1b0602e08e001fd7c104e1b080be0bf302e0ad0ffcc1d821
(4) 0x0d5e7aceec170d3469e15e793a5511602d975df9e0adfd155e104c9c8892

```

Gambar 4.8 Perintah Ganache

Tampilan tersebut menunjukkan hasil dari menjalankan perintah `ganache` di terminal, yang berfungsi untuk memulai jaringan *blockchain* lokal berbasis Ethereum menggunakan *Ganache* CLI. *Ganache* digunakan untuk menguji dan melakukan simulasi transaksi *smart contract* tanpa harus terkoneksi ke jaringan Ethereum publik. Pada awalnya muncul peringatan bahwa modul `uws_linux_x64_127.node` tidak kompatibel, namun *Ganache* tetap berhasil dijalankan dengan *fallback* ke implementasi NodeJS meskipun dengan performa yang mungkin menurun. Setelah itu, *Ganache* menampilkan 10 akun Ethereum lengkap dengan saldo awal masing-masing 1000 ETH dan private key-nya, yang bisa digunakan untuk pengujian transaksi atau *deployment* *smart contract* selama proses pengembangan. Jaringan ini berjalan di RPC server lokal dan sangat berguna dalam tahap pengujian sistem berbasis *blockchain*.

4.1.8 Tampilan Pembuatan Server Web



```
greista@greista-VirtualBox:~/Unduhan/Tubes ISI ke1/Tubes ISI/Web31$ cd ..
greista@greista-VirtualBox:~/Unduhan/Tubes ISI ke1/Tubes ISI$ cd Web31
greista@greista-VirtualBox:~/Unduhan/Tubes ISI ke1/Tubes ISI/Web31$ python3 -m http.server 6767
Serving HTTP on 0.0.0.0 port 6767 (HTTP://0.0.0.0:6767) ...
127.0.0.1 - - [19/Jun/2025 01:53:04] "GET /index.html HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2025 01:53:04] "GET /style.css HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2025 01:53:04] "GET /assets/qrис.jpeg HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2025 01:53:04] "GET /script.js HTTP/1.1" 200 -
```

Gambar 4.9 Pembuatan Server Web

Tampilan tersebut menunjukkan bahwa telah berhasil menjalankan *server web* lokal menggunakan perintah `python3 -m http.server 6767`. Perintah ini membuat *server HTTP* sederhana yang menyajikan file dari direktori `Web31` pada port `6767`. Terminal menampilkan log aktivitas akses web, termasuk permintaan HTTP untuk halaman utama (`GET /`), stylesheet (`style.css`), gambar QR (`qrис.jpeg`), dan skrip JavaScript (`script.js`). Semua permintaan berasal dari `127.0.0.1`, yang menunjukkan bahwa akses dilakukan secara lokal melalui *browser*. Ini adalah langkah awal penting dalam menyajikan *dashboard* Web3 berbasis HTML/JS agar dapat diakses dan diuji melalui *browser* tanpa harus mengunggahnya ke server eksternal.

4.1.9 Tampilan PyQt



Gambar 4.10 Tampilan PyQt

Gambar tersebut merupakan tampilan antarmuka GUI PyQt dari aplikasi *monitoring* sensor SHT20 yang terhubung ke InfluxDB. Aplikasi ini menampilkan data suhu dan kelembaban secara *real-time* di bagian atas dengan nilai numerik yang diperbarui terus-menerus. Di bawahnya terdapat dua grafik: Grafik Suhu dengan titik-titik merah dan Grafik Kelembaban

dengan garis dan simbol biru, masing-masing menampilkan tren nilai selama beberapa detik terakhir.

Selain itu, tersedia fitur untuk menampilkan data historis berdasarkan waktu dengan input *Start* dan *End* dalam format RFC3339. Tombol "Tampilkan Riwayat" memungkinkan pengguna untuk memuat grafik historis sesuai waktu yang ditentukan. Tampilan ini sangat berguna untuk memantau kondisi lingkungan dan analisis tren data sensor secara visual dan interaktif dalam satu antarmuka *desktop*.

4.2 Pembahasan

Sistem *monitoring* suhu dan kelembaban yang dirancang dalam proyek ini telah berhasil diimplementasikan dengan baik, mulai dari pembacaan data sensor hingga pencatatan ke *blockchain*. Proses diawali dari *client* sensor yang menggunakan protokol Modbus RTU untuk membaca nilai suhu dan kelembaban dari sensor SHT20, kemudian mengirimkannya ke server melalui koneksi TCP dalam format JSON. Selanjutnya, server yang dibangun dengan bahasa Rust menerima data tersebut, menyimpannya ke dalam InfluxDB dengan format *Line Protocol*, serta mencatat data yang sama ke dalam blockchain Ethereum melalui *smart contract* menggunakan *library* 'ethers-rs'.

Pengujian menunjukkan bahwa data dari sensor dapat dikirim secara berkala dan ditampilkan baik dalam bentuk teks maupun grafik. Pada sisi visualisasi, data *real-time* dapat dimonitor menggunakan aplikasi desktop PyQt, yang menampilkan nilai suhu dan kelembaban secara langsung serta dalam bentuk grafik waktu nyata. Selain itu, data historis juga dapat diakses dengan memasukkan rentang waktu tertentu, dan hasilnya divisualisasikan dengan rapi menggunakan Matplotlib. Pada sisi web, *dashboard* Web3 berhasil menampilkan data yang tersimpan di *smart contract*, baik dalam bentuk tabel maupun grafik, serta menyediakan fitur simulasi pemesanan *container* berdasarkan data suhu dan kelembaban. QR code ditampilkan sebagai bagian dari simulasi pembayaran, menambah kesan realistik terhadap proses transaksi berbasis *blockchain*.

Dalam hal infrastruktur *blockchain*, *Ganache* digunakan sebagai jaringan Ethereum lokal untuk pengujian transaksi dan *deployment smart contract*. Meskipun muncul peringatan kompatibilitas saat menjalankan *Ganache*, jaringan tetap berfungsi dengan baik dan memungkinkan *server Rust* untuk terhubung serta melakukan transaksi. Secara keseluruhan, seluruh proses dari sensor hingga pencatatan di *blockchain* berjalan stabil dan akurat. Tidak ditemukan error fatal dalam sistem selama pengujian, dan setiap komponen dapat berfungsi sesuai perannya.

Sistem ini menunjukkan kemampuan nyata dalam menerapkan teknologi terkini untuk meningkatkan transparansi dan akurasi dalam pencatatan data lingkungan. Kombinasi antara sensor industri, penyimpanan *time-series*, visualisasi *real-time*, dan pencatatan desentralisasi memberikan solusi yang inovatif untuk *monitoring* dan pelacakan kondisi lingkungan di sektor-sektor seperti logistik, pertanian, atau *midstream* migas.

BAB 5 Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem *monitoring* suhu dan kelembaban yang dilakukan, dapat disimpulkan bahwa sistem berhasil bekerja secara terpadu dan efisien. Sensor SHT20 yang dikendalikan melalui protokol Modbus RTU mampu membaca data lingkungan secara akurat dan stabil. Data yang dikirim melalui jaringan TCP ke server Rust kemudian berhasil disimpan ke dalam *database* InfluxDB dan secara bersamaan dicatat ke dalam *blockchain* Ethereum melalui *smart contract*. Hal ini membuktikan bahwa sistem mendukung pencatatan data yang bersifat transparan, terdistribusi, dan tidak dapat diubah (*immutable*).

Visualisasi data berhasil ditampilkan dalam dua bentuk antarmuka: pertama, GUI berbasis PyQt untuk *monitoring* lokal yang mendukung *real-time* dan histori data; kedua, *dashboard* Web3 yang menampilkan data dari *blockchain* dan menyediakan fitur simulasi pemesanan *container* berbasis parameter suhu dan kelembaban. Sistem *blockchain* lokal menggunakan *Ganache* CLI berjalan lancar sebagai jaringan Ethereum pengujian, dan transaksi *smart contract* berhasil dieksekusi sesuai fungsi yang ditentukan.

Dengan demikian, sistem ini membuktikan bahwa integrasi antara teknologi IoT, InfluxDB, visualisasi modern, dan *blockchain* dapat diterapkan secara nyata dalam industri, khususnya untuk meningkatkan akurasi data dan transparansi dalam rantai pasok seperti pada sektor *midstream* migas, pertanian, dan logistik.

5.2 Saran

1. Peningkatan Keamanan dan Otorisasi

Sistem saat ini belum dilengkapi dengan mekanisme otentikasi pada pengiriman data sensor maupun transaksi *blockchain*. Penambahan fitur keamanan seperti enkripsi data, otentikasi berbasis token, dan pembatasan akses API sangat disarankan untuk mencegah manipulasi atau penyusupan data.

2. Implementasi *Smart Contract* yang Lebih Kompleks

Fungsi *smart contract* masih bersifat satu arah (mencatat data). Pada pengembangan selanjutnya, *smart contract* dapat dikembangkan untuk mendukung otomatisasi pengambilan keputusan seperti peringatan dini, *trigger* aksi, atau pencatatan audit log berdasarkan kondisi tertentu.

3. Penambahan Sensor dan Skala Implementasi

Sistem dapat diperluas dengan menambahkan lebih banyak sensor dan lokasi *monitoring* untuk skala industri yang lebih besar. Hal ini memerlukan pengelolaan identitas sensor yang lebih sistematis serta manajemen data yang lebih terstruktur.

4. Pengembangan Integrasi dengan Sistem Manajemen Logistik atau ERP

Data sensor yang telah terverifikasi dan transparan sangat bermanfaat bila diintegrasikan dengan sistem manajemen logistik atau ERP (*Enterprise Resource Planning*) untuk mendukung proses bisnis secara menyeluruh.

DAFTAR PUSTAKA

- Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4), 2347–2376. <https://doi.org/10.1109/COMST.2015.2444095>
- Haque, B., Hasan, R., & Zihad, O. M. (2021). SmartOil: Blockchain and smart contract-based oil supply chain management. *IET Blockchain*, 1(2–4), 95–104. <https://doi.org/10.1049/blc2.12005>
- Lu, H., Huang, K., Azimi, M., & Guo, L. (2019). Blockchain technology in the oil and gas industry: A review of applications, opportunities, challenges, and risks. *IEEE Access*, 7, 41426–41444. <https://doi.org/10.1109/ACCESS.2019.2907695>
- Lu, Y., Huang, H., & Liu, C. (2019). Overview of Oil and Gas Industry Chain and its Digital Transformation. *Journal of Petroleum Exploration and Production Technology*, 9, 2433–2447. <https://doi.org/10.1007/s13202-019-0712-6>
- Wang, X., Zha, X., Ni, W., Liu, R. P., Guo, Y. J., Niu, X., & Zheng, K. (2019). Survey on blockchain for Internet of Things. *Computer Communications*, 136, 10–29. <https://doi.org/10.1016/j.comcom.2019.01.006>

LAMPIRAN

- *Source code* tampilan InfluxDB

```
from(bucket: "tugas3")
|> range(start: v.timeRangeStart, stop: v.timeRangeStop)
|> filter(fn: (r) => r["_measurement"] == "monitoring")
|> filter(fn: (r) => r["_field"] == "humidity" or r["_field"] == "temperature")
|> filter(fn: (r) => r["location"] == "Mobil Tanki 1")
|> filter(fn: (r) => r["sensor_id"] == "SHT20-MobilTanki-001")
|> filter(fn: (r) => r["stage"] == "Distilasi")
|> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
|> yield(name: "mean")
```

- *Source code* script.js Web3

```
const contractAddress = "0x478103fadea92c329018adc7f0912af637af69ac";
const abiPath = "abi/SensorStorage.abi";
const rpcURL = "http://127.0.0.1:8545";

let chart;
let cachedEvents = [];

async function loadSensorData() {
    const abiRes = await fetch(abiPath);
    const abi = await abiRes.json();

    const provider = new ethers.JsonRpcProvider(rpcURL);
    const contract = new ethers.Contract(contractAddress, abi, provider);

    const filter = contract.filters.DataStored();
    const events = await contract.queryFilter(filter, 0, "latest");
    cachedEvents = events;

    const tableBody = document.querySelector("#sensorTable tbody");
    tableBody.innerHTML = "";

    const labels = [], temps = [], hums = [];

    events.forEach((e) => {
        const data = e.args;
        const timeStr = new Date(Number(data.timestamp) * 1000).toLocaleString();
        const temp = Number(data.temperature) / 100;
        const hum = Number(data.humidity) / 100;

        tableBody.innerHTML += `
            <tr>
                <td>${timeStr}</td>
                <td>${data.sensorId}</td>
                <td>${data.location}</td>
                <td>${data.stage}</td>
```

```

        <td>${temp.toFixed(2)}</td>
        <td>${hum.toFixed(2)}</td>
    </tr>
';

labels.push(timeStr);
temps.push(temp);
hums.push(hum);
});

renderChart(labels, temps, hums);
}

function simulatePurchase() {
const name = document.getElementById("buyerName").value.trim();
const date = document.getElementById("purchaseDate").value;
const amount = parseInt(document.getElementById("amount").value);
const desiredTemp = parseFloat(document.getElementById("desiredTemp").value);
const desiredHum = parseFloat(document.getElementById("desiredHum").value);

if (!name || !date || !amount || isNaN(desiredTemp) || isNaN(desiredHum)) {
    alert("Harap lengkapi semua field.");
    return;
}

const tempTolerance = 0.1;
const humTolerance = 0.1;

const matchingData = cachedEvents.filter((e) => {
    const temp = Number(e.args.temperature) / 100;
    const hum = Number(e.args.humidity) / 100;
    return (
        Math.abs(temp - desiredTemp) <= tempTolerance &&
        Math.abs(hum - desiredHum) <= humTolerance
    );
});

if (matchingData.length < amount) {
    alert(`Data yang cocok hanya ditemukan sebanyak ${matchingData.length}.`);
    return;
}

const totalEth = amount * 1;
document.getElementById("ethTotal").innerText = totalEth;
document.getElementById("qrContainer").classList.remove("hidden");
alert(`Simulasi pembelian ${amount} container oleh ${name} pada ${date} berhasil!`);
}

```

```

}

function renderChart(labels, temps, hums) {
  const ctx = document.getElementById("chart").getContext("2d");
  if (chart) chart.destroy();

  chart = new Chart(ctx, {
    type: "line",
    data: {
      labels,
      datasets: [
        {
          label: "Temperature (°C)",
          borderColor: "#e74c3c",
          data: temps,
          fill: false,
        },
        {
          label: "Humidity (%)",
          borderColor: "#3498db",
          data: hums,
          fill: false,
        },
      ],
    },
  });
}

```

- *Source code main.rs server*

```

use tokio::net::TcpListener;
use tokio::io::{AsyncBufReadExt, BufReader};
use serde::Deserialize;
use reqwest::Client;

use ethers::prelude::*;
use ethers::abi::Abi;
use std::{fs, sync::Arc};
use chrono::DateTime;

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

#[tokio::main]

```

```

async fn main() -> anyhow::Result<()> {
    // --- InfluxDB setup ---
    let influx_url = "http://localhost:8086/api/v2/write?org=greista&bucket=tugas3&precision=s";
    let influx_token = "LPUZWJRM5aJ48PRfqgK5VozzXMg2BOGMyf_HPbY0H0jVeXbl9ksw3N1DuPbKtUFsb1gNw5gSSCkLsquOaiZ_Hg==";
    let http_client = Client::new();

    // --- Ethereum setup ---
    let provider = Provider::try_from("http://127.0.0.1:8545")?;
    let wallet: LocalWallet = "0xdc7b400adf8cc0a0a02a1a5d6ceac7a67be9e6310baef6fea962d92e40a7c54d"
        .parse::<LocalWallet>()?;
    let client = Arc::new(SignerMiddleware::new(provider, wallet));

    // Baca dan parse ABI dan bytecode dengan benar
    let abi_str = fs::read_to_string("build/SensorStorage.abi")?;
    let bytecode_str = fs::read_to_string("build/SensorStorage.bin")?;

    let abi: Abi = serde_json::from_str(&abi_str)?;
    let bytecode = bytecode_str.trim().parse::<Bytes>()?;

    let factory = ContractFactory::new(abi, bytecode, client.clone());

    let contract = factory.deploy()?.send().await?;
    println!(" ✅ Smart contract deployed at: {:?}", contract.address());

    // --- TCP Server ---
    let listener = TcpListener::bind("0.0.0.0:9000").await?;
    println!(" 📡 TCP Server listening on port 9000...");

    loop {
        let (socket, addr) = listener.accept().await?;
        println!(" 🎁 New connection from {}", addr);

        let influx_url = influx_url.to_string();
        let influx_token = influx_token.to_string();
        let http_client = http_client.clone();
        let contract = contract.clone();

        tokio::spawn(async move {
            let reader = BufReader::new(socket);
            let mut lines = reader.lines();

            while let Ok(Some(line)) = lines.next_line().await {
                match serde_json::from_str::<SensorData>(&line) {
                    Ok(data) => {
                        println!(" 💡 Received sensor data: {:?}", data);
                    }
                }
            }
        });
    }
}

```

```

// --- InfluxDB Write ---
let timestamp = DateTime::parse_from_rfc3339(&data.timestamp)
    .unwrap()
    .timestamp();

let line_protocol = format!(
    "monitoring,sensor_id={},location={},stage={}
temperature={},humidity={} {}",
    data.sensor_id.replace(" ", "\\"),
    data.location.replace(" ", "\\"),
    data.process_stage.replace(" ", "\\"),
    data.temperature_celsius,
    data.humidity_percent,
    timestamp
);

match http_client
    .post(&influx_url)
    .header("Authorization", format!("Token {}", influx_token))
    .header("Content-Type", "text/plain")
    .body(line_protocol)
    .send()
    .await
{
    Ok(resp) if resp.status().is_success() => {
        println!("✅ InfluxDB: data written");
    }
    Ok(resp) => {
        println!("⚠️ InfluxDB error: {}", resp.status());
    }
    Err(e) => {
        println!("❌ InfluxDB HTTP error: {}", e);
    }
}

// --- Ethereum Contract Write ---
let method_call = contract
.method:<_, H256>("storeData", (
    timestamp as u64,
    data.sensor_id.clone(),
    data.location.clone(),
    data.process_stage.clone(),
    (data.temperature_celsius * 100.0) as i64,
    (data.humidity_percent * 100.0) as i64,
))
.unwrap();

let tx = method_call.send().await;

```

```

        match tx {
            Ok(pending_tx) => {
                println!("📡 Ethereum: tx sent: {:?}", pending_tx);
            }
            Err(e) => {
                println!("✖ Ethereum tx error: {:?}", e);
            }
        }
    }
    Err(e) => println!("✖ Invalid JSON received: {}", e),
}
);
}
}
}

```

- *Source code* main.rs sensor

```

use tokio_modbus::{client::rtu, prelude::*};
use tokio_serial::{SerialPortBuilderExt, Parity, StopBits, DataBits};
use tokio::net::TcpStream;
use tokio::io::AsyncWriteExt;
use serde::Serialize;
use chrono::Utc;
use std::error::Error;
use tokio::time::{sleep, Duration};

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

async fn read_sensor(slave: u8) -> Result<Vec<u16>, Box<dyn Error>> {
    let builder = tokio_serial::new("/dev/ttyUSB0", 9600)
        .parity(Parity::None)
        .stop_bits(StopBits::One)
        .data_bits(DataBits::Eight)
        .timeout(std::time::Duration::from_secs(1));

    let port = builder.open_native_async()?;
    let mut ctx = rtu::connect_slave(port, Slave(slave)).await?;
    let response = ctx.read_input_registers(1, 2).await?;

    Ok(response)
}

```

```

#[tokio::main]
async fn main() -> Result<(), Box<dyn Error>> {
    loop {
        match read_sensor(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} %", temp, rh);

                let data = SensorData {
                    timestamp: Utc::now().to_rfc3339(),
                    sensor_id: "SHT20-MobilTanki".into(),
                    location: "Mobil Tanki".into(),
                    process_stage: "Container".into(),
                    temperature_celsius: temp,
                    humidity_percent: rh,
                };
            }

            let json = serde_json::to_string(&data)?;

            match TcpStream::connect("0.0.0.0:9000").await {
                Ok(mut stream) => {
                    stream.write_all(json.as_bytes()).await?;
                    stream.write_all(b"\n").await?;
                    println!("✅ Data dikirim ke TCP server");
                },
                Err(e) => {
                    println!("❌ Gagal koneksi ke TCP server: {}", e);
                }
            },
            Ok(other) => {
                println!("⚠️ Data tidak lengkap: {:?}", other);
            },
            Err(e) => {
                println!("❌ Gagal baca sensor: {}", e);
            }
        }

        sleep(Duration::from_secs(5)).await;
    }
}

```

- *Source code web3 index.html*

```
index.HTML
```

```
<!DOCTYPE html>
```

```

<html lang="en">
<head>
<meta charset="UTF-8">
<title> 📦 Blockchain Container Purchase</title>
<script src="https://cdn.jsdelivr.net/npm/ethers/dist/ethers.umd.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<link rel="stylesheet" href="style.css">
</head>
<body>
<h1> 📦 Blockchain Container Purchase Dashboard</h1>

<div class="form-container">
<h2> 📈 Simulasi Pemesanan</h2>
<input type="text" id="buyerName" placeholder="Nama Pembeli">
<input type="date" id="purchaseDate">
<input type="number" id="amount" placeholder="Jumlah Container" min="1">
<input type="number" step="0.01" id="desiredTemp" placeholder="Suhu yang Diinginkan (°C)">
<input type="number" step="0.01" id="desiredHum" placeholder="Kelembapan yang Diinginkan (%)">
<button onclick="simulatePurchase()">➡ Filter & Simulate Purchase</button>
</div>

<div id="qrContainer" class="hidden">
<h3> 💰 QR Code Simulasi Pembayaran</h3>

<p>Total: <span id="ethTotal">0</span> ETH (simulasi)</p>
</div>

<div class="button-load">
<button onclick="loadSensorData()">🔄 Load Sensor Data</button>
</div>

<table id="sensorTable">
<thead>
<tr>
<th>Timestamp</th>
<th>Sensor ID</th>
<th>Location</th>
<th>Stage</th>
<th>Temperature (°C)</th>
<th>Humidity (%)</th>
</tr>
</thead>
<tbody></tbody>
</table>

<canvas id="chart" height="100"></canvas>
<script src="script.js"></script>

```

```
</body>
</html>
```

- SensorStorage.Sol

```
Contracts > SensorStorage.Sol

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SensorStorage {
    event DataStored(
        uint256 timestamp,
        string sensorId,
        string location,
        string stage,
        int256 temperature,
        int256 humidity
    );

    function storeData(
        uint256 timestamp,
        string memory sensorId,
        string memory location,
        string memory stage,
        int256 temperature,
        int256 humidity
    ) public {
        emit DataStored(timestamp, sensorId, location, stage, temperature,
humidity);
    }
}
```

- *Source code style.css Web3*

```
body {
    font-family: Arial, sans-serif;
    background-color: #f5f7fa;
    text-align: center;
    margin: 0;
    padding: 0;
}

h1 {
    margin-top: 20px;
    color: #2c3e50;
}

.form-container {
    margin: 20px auto;
```

```
background: white;
padding: 20px;
border-radius: 10px;
box-shadow: 0 0 10px rgba(0,0,0,0.1);
width: 300px;
}

.form-container input,
.form-container button {
    display: block;
    width: 100%;
    margin: 10px 0;
    padding: 10px;
    font-size: 14px;
    border-radius: 5px;
    border: 1px solid #ccc;
}

.form-container button {
    background-color: #2ecc71;
    color: white;
    border: none;
    cursor: pointer;
}

#qrContainer {
    margin-top: 20px;
}

#qrContainer img {
    width: 300px;
    height: auto;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0,0,0,0.2);
}

.button-load button {
    margin: 20px auto;
    padding: 10px 20px;
    font-size: 16px;
    background-color: #3498db;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

.hidden {
    display: none;
}
```

```

#sensorTable {
    width: 90%;
    margin: 20px auto;
    border-collapse: collapse;
}

#sensorTable th,
#sensorTable td {
    border: 1px solid #ccc;
    padding: 8px;
}

#sensorTable thead {
    background-color: #007bff;
    color: white;
}

```

- *Source code* gui.py

```

import tkinter as tk
from tkinter import ttk
import requests
import threading
import time
import csv
from io import StringIO
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
from collections import deque

# Konfigurasi InfluxDB
INFLUX_QUERY_URL = "http://localhost:8086/api/v2/query"
ORG = "greista"
BUCKET = "Tugas4"
TOKEN = "5V6JaLA3gG0ZbdBI-h-Crvmz_qgIaIaGaDKTCtcEGwSnL5UvwsR_w0Jiv3YrvrGFwRaM806c0yA-I9bNsq1D2_g=="

# Riwayat data
history_length = 50
temp_history = deque(maxlen=history_length)
rh_history = deque(maxlen=history_length)
time_history = deque(maxlen=history_length)

def get_latest_data():
    flux_query = f"""
        from(bucket: "{BUCKET}")
        |> range(start: -1m)
        |> filter(fn: (r) => r._measurement == "monitoring")
        |> filter(fn: (r) => r. field == "temperature" or r. field == "humidity")"""

```

```

|> last()
"""

headers = {
    "Authorization": f"Token {TOKEN}",
    "Content-Type": "application/vnd.flux",
    "Accept": "application/csv"
}

try:
    response = requests.post(
        INFLUX_QUERY_URL,
        params={"org": ORG},
        headers=headers,
        data=flux_query
    )

    reader = csv.DictReader(StringIO(response.text))
    data = {}
    for row in reader:
        try:
            field = row["_field"]
            value = float(row["_value"])
            data[field] = value
        except:
            continue

    if "temperature" in data and "humidity" in data:
        return data["temperature"], data["humidity"]
    return None
except Exception as e:
    print("✖ Exception query Influx:", e)
    return None

def get_data_range(start_time, end_time):
    flux_query = f"""
from(bucket: "{BUCKET}")
|> range(start: {start_time}, stop: {end_time})
|> filter(fn: (r) => r._measurement == "monitoring")
|> filter(fn: (r) => r._field == "temperature" or r._field == "humidity")
"""

    headers = {
        "Authorization": f"Token {TOKEN}",
        "Content-Type": "application/vnd.flux",
        "Accept": "application/csv"
    }

    try:
        response = requests.post(

```

```

INFLUX_QUERY_URL,
params={"org": ORG},
headers=headers,
data=flux_query
)

reader = csv.DictReader(StringIO(response.text))
temp_map = {}
rh_map = {}

for row in reader:
    try:
        t = row["_time"]
        field = row["_field"]
        value = float(row["_value"])
        if field == "temperature":
            temp_map[t] = value
        elif field == "humidity":
            rh_map[t] = value
    except:
        continue

sorted_keys = sorted(set(temp_map.keys()) & set(rh_map.keys()))
temps = [temp_map[t] for t in sorted_keys]
rhs = [rh_map[t] for t in sorted_keys]
times = [t[11:19] for t in sorted_keys] # jam:menit:detik

return temps, rhs, times
except Exception as e:
    print("✖ Exception query Influx:", e)
    return [], [], []

```

```

def update_data():
    while True:
        result = get_latest_data()
        current_time = time.strftime("%H:%M:%S")

        if result:
            temp, rh = result
            label_temp.config(text=f"Suhu: {temp:.1f} °C")
            label_rh.config(text=f"Kelembaban: {rh:.1f} %")
            status_label.config(text="✓ Data dari Influx")

            temp_history.append(temp)
            rh_history.append(rh)
            time_history.append(current_time)

            plot_graph()
        else:
            label_temp.config(text="Suhu: ---")

```

```

label_rh.config(text="Kelembaban: ---")
status_label.config(text="✖ Gagal ambil data")

time.sleep(2)

def plot_graph():
    ax1.clear()
    ax2.clear()

    # Set background ke hitam
    fig.patch.set_facecolor('black')
    ax1.set_facecolor('black')
    ax2.set_facecolor('black')

    x = list(range(len(time_history)))
    times = list(time_history)

    # Plot data dengan garis dan warna yang kontras
    ax1.plot(x, list(temp_history), label='Suhu (°C)', color='red', marker='o',
              linestyle='--')
    ax2.plot(x, list(rh_history), label='Kelembaban (%)', color='cyan',
              marker='x', linestyle='--')

    ax1.set_title("Grafik Suhu", color='white')
    ax2.set_title("Grafik Kelembaban", color='white')
    ax1.set_ylabel("°C", color='white')
    ax2.set_ylabel("%", color='white')

    # Tampilkan hanya label waktu setiap 5 data
    interval = 5
    tick_positions = x[::interval]
    tick_labels = times[::interval]

    ax1.set_xticks(tick_positions)
    ax2.set_xticks(tick_positions)
    ax1.set_xticklabels(tick_labels, rotation=45, ha="right", color='white')
    ax2.set_xticklabels(tick_labels, rotation=45, ha="right", color='white')

    for ax in [ax1, ax2]:
        ax.tick_params(axis='y', colors='white')
        ax.tick_params(axis='x', colors='white')
        ax.grid(True, linestyle='--', alpha=0.3, color='gray')
        for spine in ax.spines.values():
            spine.set_color('white')

    fig.tight_layout()
    canvas.draw()

def show_history():
    start = entry_start.get()

```

```

end = entry_end.get()
tempss, rhs, times = get_data_range(start, end)

if tempss and rhs:
    temp_history.clear()
    rh_history.clear()
    time_history.clear()

    temp_history.extend(tempss)
    rh_history.extend(rhs)
    time_history.extend(times)

    label_temp.config(text="(Hist) Suhu: -- °C")
    label_rh.config(text="(Hist) RH: -- %")
    status_label.config(text=" ✅ Menampilkan data historis")
    plot_graph()
else:
    status_label.config(text=" ❌ Tidak ada data historis")

# GUI Setup
root = tk.Tk()
root.title("Monitor SHT20 dari InfluxDB")
root.geometry("800x650")

label_temp = tk.Label(root, text="Suhu: -- °C", font=("Helvetica", 16))
label_temp.pack(pady=5)

label_rh = tk.Label(root, text="Kelembaban: -- %", font=("Helvetica", 16))
label_rh.pack(pady=5)

status_label = tk.Label(root, text="Status: ---", fg="blue")
status_label.pack(pady=5)

frame_input = tk.Frame(root)
frame_input.pack(pady=5)

tk.Label(frame_input, text="Start (RFC3339):").grid(row=0, column=0,
padx=5)
entry_start = tk.Entry(frame_input, width=30)
entry_start.grid(row=0, column=1)

tk.Label(frame_input, text="End (RFC3339):").grid(row=1, column=0,
padx=5)
entry_end = tk.Entry(frame_input, width=30)
entry_end.grid(row=1, column=1)

btn_show = tk.Button(frame_input, text="Tampilkan Riwayat",
command=show_history)
btn_show.grid(row=0, column=2, rowspan=2, padx=10)

```

```
fig = Figure(figsize=(6, 4), dpi=100)
ax1 = fig.add_subplot(211)
ax2 = fig.add_subplot(212)

canvas = FigureCanvasTkAgg(fig, master=root)
canvas.get_tk_widget().pack(pady=10)

# Mulai update realtime
threading.Thread(target=update_data, daemon=True).start()

root.mainloop()
```

BIODATA PENULIS



Muhammad Hadid Qushairi lahir di Prabumulih, Sumatera Selatan tanggal 04 Desember 2005. Ia menyelesaikan pendidikan dasar di SDIT Ishlahul Ummah Prabumulih (tahun 2010-2017) dan di SMP Negeri 01 Prabumulih (tahun 2017-2020). Pendidikan menengahnya ditempuh di SMK Negeri 01 Prabumulih dengan jurusan Teknik Instalasi Tenaga Listrik (tahun 2020-2023, Kemudian, melanjutkan kuliah di Institut Teknologi Sepuluh Nopember dengan program studi D4 Teknologi Rekayasa Instrumentasi dari 2023 – sekarang dengan beasiswa full dari PT. Tanjung Enim Lestari. Penulis memperoleh pengalaman selama magang di PT. Tanjung Enim Lestari sebagai helper di bengkel listrik di masa SMK. Penulis telah memegang peran kepemimpinan dan hubungan masyarakat untuk acara-acara seperti UKM EXPO ITS dan VOITSFEST ITS.



Greista Tear Rizki Saputra lahir di kota Madiun, Jawa Timur tanggal 20 Juli 2004. Ia menyelesaikan pendidikan dasar di SD Kartoharjo 1 (tahun 2010-2017) dan di SMP Negeri 04 Madiun (tahun 2017-2020). Pendidikan menengahnya ditempuh di SMA Negeri 06 Madiun dengan jurusan MIPA (tahun 2020-2023), Kemudian, melanjutkan kuliah di Institut Teknologi Sepuluh Nopember dengan program studi D4 Teknologi Rekayasa Instrumentasi dari 2023 – sekarang. Penulis telah memegang peran kepemimpinan dan hubungan masyarakat untuk acara-acara seperti GERIGI ITS dan Suaka Mahasiswa Madiun ITS(SUMMITS).



Penulis dilahirkan di Sidoarjo, 31 Oktober 2004, merupakan anak tunggal. Penulis telah menempuh pendidikan formal yaitu di TK DHARMA WANITA PERSATUAN GELAM, SD NEGERI SUMORAME, SMP NEGERI 5 SIDOARJO, dan SMK NEGERI 3 BUDURAN SIDOARJO. Setelah lulus dari SMK NEGERI 3 BUDURAN SIDOARJO Tahun 2023. Penulis mengikuti UTBK-SNBT dan diterima di Departemen Teknik Instrumentasi FV – ITS pada tahun 2023 dan terdaftar dengan NRP 2042231035.

Di Departemen Teknik Instrumentasi, Penulis aktif di beberapa kegiatan kepanitiaan yang diselenggarakan oleh departemen maupun fakultas, mengikuti kegiatan organisasi Himpunan Mahasiswa Teknik Instrumentasi dan BEM Fakultas Vokasi.