



LAPORAN TUBES - VI231418

MUSCHAIN: INTEGRASI *BLOCKCHAIN* PADA SMART MONITORING BUDIDAYA JAMUR TIRAM UNTUK OTOMATISASI DAN RANTAI PASOK BERBASIS WEB3

RAFI PRIMANSYAH

NRP. 2042231014

BAMBANG PAMARTA PANGANYOMAN

NRP. 2042231016

NURLITA FARAH LAILA

NRP. 2042231056

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

DAFTAR ISI

HALAMAN JUDUL	i
DAFTAR ISI	1
BAB I PENDAHULUAN	3
1.1 Latar Belakang	3
1.2 Rumusan Permasalahan	4
1.3 Batasan Masalah	4
1.4 Tujuan	4
1.5 Manfaat	4
BAB II TINJAUAN PUSTAKA	5
2.1 Hasil Penelitian / Perancangan Terdahulu	5
2.2 Dasar Teori	6
2.2.1 Sistem Monitoring Budidaya Jamur Tiram	6
2.2.2 Protokol Komunikasi TCP/IP dan Modbus Client	7
2.2.3 Penyimpanan Data dengan InfluxDB dan Grafana	7
2.2.4 Pengembangan Aplikasi Dekstop dengan PyQt	8
2.2.5 Teknologi Blockchain dalam Sistem Industri Pertanian	8
2.2.6 Smart Contract dan Web3	9
BAB III METODOLOGI	11
3.1 Desain Sistem	11
3.2 Flowchart Sistem	11
3.2.1 Program Server	12
BAB IV HASIL DAN PEMBAHASAN	16
4.1 Pengambilan Data	16
4.2 Tampilan Data Explorer InfluxDB	16
4.2 Tampilan Dashboard Grafana	17
4.3 Tampilan Aplikasi QT	17
4.4 Tampilan Web3 Sensor Dashboard	18
4.5 Tampilan Eksekusi Perintah Rust TCP Server	19
4.6 Tampilan Eksekusi Perintah Modbus Client	19
4.7 Tampilan Smart Contract Blockchain	19
4.8 Tampilan Hasil Run Deploy	20
4.9 Tampilan Ketika Ingin Terhubung ke Web3	20
BAB V KESIMPULAN	21
5.1 Kesimpulan	21

5.2 Saran	21
DAFTAR PUSTAKA	22
LAMPIRAN	23
Lampiran 1. Pembuatan Program Client Sensor	23
Lampiran 2. Pembuatan GUI Dekstop	25
Lampiran 3. Pembuatan Web3 Sensor Dashboard	29
Lampiran 4. Codingan Tambahan	33

BAB I PENDAHULUAN

1.1 Latar Belakang

Jamur tiram (*Pleurotus ostreatus*) merupakan salah satu komoditas hortikultura yang memiliki nilai ekonomi tinggi dan menjadi sumber pangan alternatif yang kaya nutrisi, seperti protein, serat, vitamin B, dan mineral (Wahyuni dan Prasetyo, 2020). Di Indonesia, budidaya jamur tiram berkembang pesat seiring dengan meningkatnya permintaan pasar domestik maupun ekspor. Potensi ekonomi ini menjadikan jamur tiram sebagai salah satu pilihan strategis dalam pengembangan sektor agribisnis berbasis hortikultura.

Namun demikian, keberhasilan budidaya jamur tiram sangat dipengaruhi oleh stabilitas kondisi lingkungan mikro di dalam kumbung atau rumah jamur, terutama pada parameter suhu dan kelembaban relatif. Menurut penelitian, rentang optimal suhu untuk pertumbuhan miselium jamur tiram berkisar antara 24°C–30°C, sedangkan kelembaban ideal berada pada kisaran 80%–95% (Nugroho dan Sari, 2019). Fluktuasi suhu dan kelembaban yang signifikan di luar rentang tersebut dapat menyebabkan stres fisiologis pada jamur, menurunkan produktivitas, bahkan memicu kontaminasi mikroorganisme patogen yang berujung pada kegagalan panen (Santoso *et al.*, 2021).

Kondisi ini diperburuk oleh fakta bahwa mayoritas petani jamur tiram di Indonesia masih mengandalkan metode konvensional untuk memantau dan mengontrol lingkungan budidaya. Pemantauan dilakukan secara manual menggunakan alat ukur sederhana atau bahkan hanya berdasarkan persepsi subjektif petani. Metode ini bersifat tidak presisi, tidak konsisten, dan memerlukan intervensi manusia secara terus-menerus, sehingga rentan terhadap kelalaian dan kesalahan pencatatan.

Di sisi lain, sistem rantai pasok (supply chain) produk pertanian, termasuk jamur tiram, sering kali menghadapi tantangan transparansi dan ketertelusuran (traceability). Konsumen dan distributor kesulitan untuk memverifikasi apakah produk yang mereka konsumsi dibudidayakan dalam kondisi yang sesuai standar (Purwanto, 2020).

Untuk menjawab permasalahan tersebut, perkembangan teknologi secara *real-time* membuka peluang penerapan sistem monitoring lingkungan yang otomatis, presisi, dan berkelanjutan. Melalui penggunaan sensor digital seperti SHT20, parameter suhu dan kelembaban dapat dipantau secara *real-time* dan dikirim ke sistem berbasis cloud atau database untuk analisis lebih lanjut. Namun, penyimpanan data pada sistem sentralisasi (server terpusat) masih menyisakan celah terhadap risiko manipulasi dan kehilangan data, terutama jika sistem mengalami gangguan atau serangan siber.

Sebagai solusi, teknologi blockchain hadir dengan keunggulan utama berupa sifat desentralisasi, transparansi, dan imutabilitas (data tidak dapat diubah atau dihapus setelah tercatat). Dengan mengintegrasikan blockchain dalam sistem monitoring berbasis IoT, data yang direkam oleh sensor dapat langsung dicatat ke dalam ledger digital yang aman dan diverifikasi oleh seluruh pihak dalam jaringan. Hal ini memungkinkan terbentuknya sistem dokumentasi budidaya yang tidak hanya akurat, tetapi juga tahan terhadap manipulasi.

Lebih lanjut, implementasi smart contract yaitu program digital yang berjalan secara otomatis pada jaringan blockchain memungkinkan proses validasi kondisi budidaya dilakukan secara real-time tanpa intervensi manusia. Smart contract dapat diprogram untuk mengeksekusi logika tertentu, misalnya memastikan bahwa suhu dan kelembaban tetap berada dalam batas

optimal sebelum mencatat data tersebut ke dalam blockchain.

Dengan demikian, sistem ini tidak hanya mengotomatisasi proses monitoring lingkungan, tetapi juga membangun landasan digital untuk mewujudkan rantai pasok pertanian yang transparan, akuntabel, dan terpercaya (Putra dan Widodo, 2023).

1.2 Rumusan Permasalahan

Berdasarkan latar belakang yang telah diuraikan, rumusan masalah dalam penelitian ini adalah sebagai berikut:

- a. Bagaimana membangun sistem monitoring suhu dan kelembaban untuk budidaya jamur tiram secara real-time menggunakan sensor SHT20 dan basis data InfluxDB?
- b. Bagaimana mengintegrasikan data hasil monitoring dari sensor ke dalam jaringan blockchain Ethereum untuk menjamin keamanan dan imutabilitas data?
- c. Bagaimana membuat smart contract yang dapat mengotomatisasi dan pencatatan data kondisi jamur tiram ke dalam rantai pasok?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini didefinisikan sebagai berikut.

- a. Sensor yang digunakan untuk monitoring suhu dan kelembaban adalah sensor SHT20.
- b. Protokol komunikasi antara client sensor dan server menggunakan TCP/IP.
- c. Penyimpanan data deret waktu (time-series) menggunakan InfluxDB dan visualisasi data menggunakan Grafana.
- d. Aplikasi antarmuka monitoring pada sisi server dikembangkan sebagai aplikasi desktop menggunakan framework PyQt.
- e. Platform blockchain yang digunakan adalah Ethereum, yang dijalankan pada lingkungan pengujian local Hardhat.

1.4 Tujuan

Tujuan dilakukannya penelitian ini didefinisikan sebagai berikut

- a. Membangun prototipe sistem monitoring suhu dan kelembaban pada kumbung jamur tiram menggunakan sensor SHT20.
- b. Mengimplementasikan teknologi blockchain Ethereum untuk menyimpan catatan data penting dari hasil monitoring secara aman dan transparan.
- c. Mengembangkan dan menerapkan smart contract untuk otomatisasi validasi data kondisi lingkungan sesuai standar budidaya jamur tiram.

1.5 Manfaat

Penelitian yang diusulkan ini diharapkan dapat memberi manfaat sebagai berikut.

- a. Bagi Petani: Menyediakan sistem monitoring otomatis yang akurat untuk membantu menjaga kondisi lingkungan optimal, sehingga dapat meningkatkan kuantitas dan kualitas hasil panen.
- b. Bagi Pelaku Rantai Pasok: Meningkatkan kepercayaan terhadap kualitas produk melalui data budidaya yang transparan dan tidak dapat dimanipulasi.
- c. Bagi Konsumen: Memberikan jaminan bahwa produk jamur tiram yang dikonsumsi berasal dari budidaya yang terkontrol dan berkualitas.

BAB II TINJAUAN PUSTAKA

2.1 Hasil Penelitian / Perancangan Terdahulu

Tabel 2. 1 State of The Art

Judul, Penulis, dan Tahun	Metode	Hasil	Keterbaruan
"The rise of blockchain technology in agriculture and food supply chains" (Kamilaris, A., Fonts, A., & Prenafeta-Boldú, F. X, 2019)	Melakukan survei dan analisis kritis terhadap inisiatif, proyek, dan studi kasus yang ada di sektor pertanian dan rantai pasok pangan melalui pencarian literatur dan web.	Menunjukkan teknologi blockchain menjanjikan transparansi rantai pasok makanan. Sebagian besar proyek yang diidentifikasi berada pada tahap implementasi atau proof-of-concept. Mengidentifikasi berbagai tantangan seperti teknis, edukasi, kebijakan, dan kerangka regulasi.	bahwa adalah teknologi yang yang menjanjikan untuk transparansi rantai pasok makanan. Sebagian besar proyek yang diidentifikasi berada pada tahap implementasi atau proof-of-concept. Mengidentifikasi berbagai tantangan seperti teknis, edukasi, kebijakan, dan kerangka regulasi.
"A systematic literature review of blockchain-based applications: Current status, classification and open issues" (Casino, F., Dasaklis, T. K., & Patsakis, C., 2018)	Menggunakan metode tinjauan literatur sistematis dan analisis konten tematik untuk mengklasifikasi kan aplikasi berbasis blockchain, terkait aplikasi mengidentifikasi tren, serta menyoroti area penelitian di masa depan.	Memberikan klasifikasi komprehensif dari aplikasi berbasis blockchain di berbagai sektor seperti rantai pasok, bisnis, kesehatan, dan IoT. Menunjukkan adanya peningkatan pesat dalam jumlah publikasi dalam jumlah publikasi blockchain, terkait aplikasi blockchain pada tahun 2017.	Memberikan tinjauan yang sistematis dan konkret mengenai aplikasi yang dimungkinkan oleh blockchain, bukan hanya teknologinya itu sendiri. Memberikan klasifikasi yang berorientasi pada aplikasi dan peta jalan untuk penelitian di masa depan.
"The Effect of Blockchain Technology in Enhancing Ethical Sourcing and Supply Chain Transparency: Evidence from The	Menggunakan analisis regresi terhadap data dari 153 perusahaan di sektor kakao dan pertanian Ghana	Hasil menunjukkan bahwa adopsi teknologi blockchain berpengaruh signifikan dan positif terhadap transparansi rantai pasok dan praktik ethical sourcing.	Memberikan bukti empiris dan kuantitatif mengenai dampak adopsi blockchain pada ethical sourcing dan transparansi.

Cocoa and Agricultural Sectors in Ghana" (Ibrahim, M., et al., 2024)	yang menggunakan teknologi blockchain. Kerangka teori yang digunakan adalah Teori Difusi Inovasi.	Perusahaan yang menggunakan blockchain cenderung 47.2% lebih mungkin terlibat dalam ethical sourcing.	Penelitian berfokus pada konteks spesifik negara berkembang, yaitu sektor kakao dan pertanian di Ghana, yang mengisi celah dari penelitian sebelumnya yang seringkali bersifat konseptual.
"An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends" (Zheng, Z., et al., 2017)	Merupakan sebuah tinjauan komprehensif (overview paper) yang mensintesis informasi dari berbagai sumber.	Memberikan tinjauan arsitektur blockchain , taksonomi jenis blockchain (publik, privat, konsorsium) , perbandingan berbagai algoritma konsensus (PoW, PoS, PBFT) , serta tantangan teknis seperti skalabilitas dan privasi	Memberikan perbandingan teknis yang sistematis antar algoritma konsensus dalam satu tabel. Menyajikan tantangan teknis dan solusi-solusi terkini yang ada pada saat itu, menjadikannya panduan teknis yang fundamental.

2.2 Dasar Teori

Berikut merupakan beberapa teori yang digunakan untuk menunjang analisa Integrasi *Smart Contract Ethereum Pada Monitoring Budidaya Jamur Tiram Untuk Otomatisasi Dan Keamanan Data Rantai Pasok Berbasis Blockchain* sebagai berikut.

2.2.1 Sistem Monitoring Budidaya Jamur Tiram

Budidaya jamur tiram (*Pleurotus ostreatus*) merupakan kegiatan pertanian berbasis mikroklimat yang sangat dipengaruhi oleh kondisi lingkungan di sekitar media tanam, khususnya suhu dan kelembaban udara. Keberhasilan dalam setiap tahapan budidaya baik pada fase pertumbuhan miselium maupun fase pembentukan tubuh buah sangat bergantung pada kestabilan parameter-parameter tersebut. Ketidaksesuaian suhu dan kelembaban dapat menyebabkan pertumbuhan jamur menjadi lambat, menghasilkan tubuh buah yang cacat, atau bahkan gagal panen secara keseluruhan (Suryaningsih, 2018).

Pada fase inkubasi atau pertumbuhan miselium, jamur tiram membutuhkan suhu lingkungan yang relatif hangat, umumnya berada pada kisaran 22°C hingga 28°C. Dalam fase ini, kelembaban udara optimal berkisar antara 60% hingga 70%, yang berfungsi untuk menjaga kelembaban media tanam tanpa menciptakan kondisi terlalu lembab yang bisa menyebabkan pertumbuhan mikroorganisme kontaminan (Rachman dan Lestari, 2021). Setelah miselium tumbuh sempurna, proses budidaya memasuki fase generatif, yaitu pembentukan tubuh buah. Pada fase ini, kondisi lingkungan perlu disesuaikan . Suhu optimal diturunkan ke kisaran 18°C hingga 25°C, sementara kelembaban udara dinaikkan secara signifikan ke rentang 80% hingga

95%.

Kondisi ini bertujuan untuk meniru lingkungan alami jamur yang tumbuh di hutan tropis dengan kelembaban tinggi dan suhu yang relatif sejuk (Sensirion AG, 2019). Pencahayaan difus dan sirkulasi udara yang baik juga diperlukan agar jamur tumbuh dengan bentuk dan warna yang optimal.

Untuk memastikan parameter-parameter lingkungan tersebut dapat dipantau secara akurat dan berkelanjutan, penggunaan sensor digital menjadi krusial dalam sistem pemantauan otomatis. Salah satu sensor yang banyak digunakan adalah SHT20, yaitu sensor suhu dan kelembaban digital yang diproduksi oleh Sensirion AG. Sensor ini menggunakan teknologi CMOSens® yang memberikan stabilitas jangka panjang, konsumsi daya rendah, serta akurasi tinggi dengan toleransi $\pm 0.3^{\circ}\text{C}$ untuk suhu dan $\pm 3\%$ RH untuk kelembaban relatif (Gunawan dan Kurniawan, 2020).

2.2.2 Protokol Komunikasi TCP/IP dan Modbus Client

Transmission Control Protocol (TCP) dan Internet Protocol (IP) merupakan protokol yang digunakan secara bersamaan dan digunakan sebagai protokol transport untuk internet. Ketika informasi modbus dikirim menggunakan protokol ini, data yang akan diteruskan ke TCP mana informasi tambahan terpasang dan diberikan kepada IP. IP kemudian menempatkan data dalam paket (atau datagram) dan kemudian dikirim. TCP harus membuat sambungan sebelum mentransfer data, karena merupakan protokol berbasis koneksi.

Master (atau Client di Modbus TCP) menetapkan koneksi dengan Slave (atau Server). Server menunggu untuk koneksi masuk dari klien. Setelah sambungan dibuat, Server kemudian merespon permintaan dari klien sampai klien menutup koneksi. Modbus TCP/IP lebih cepat dalam melakukan transfer data dibanding dengan Modbus RTU. Pada aplikasi sistem SCADA atau pun Automation yang kompleks dimana digunakan perangkat IED dalam jumlah yang banyak dan beraneka ragam atau dimana tingkat traffic transfer data yang padat, lebih disarankan menggunakan Modbus TCP/IP untuk mencapai tingkat real-time yang lebih tinggi. Namun tentu saja perangkat IED dengan Port TCP/IP itu sendiri harganya relatif lebih mahal dibanding dengan Modbus RTU yang hanya menggunakan komunikasi Port RS-485 (Ariwibisono et al., 2023). Perbedaan mendasar Modbus TCP dengan modbus RTU salah satunya adalah pada frame data yang dikirimkan.

2.2.3 Penyimpanan Data dengan InfluxDB dan Grafana

InfluxDB merupakan sebuah database time series open-source yang dikembangkan oleh InfluxData. InfluxDB didukung oleh Bahasa Go (Titin Nurfadila Sudirman, 2019). InfluxDB digunakan sebagai penyimpanan data untuk setiap kasus yang melibatkan sejumlah besar data time-stamped, termasuk pemantauan DevOps, data log, metrik 8 aplikasi, data sensor IoT, dan analisis real-time. Konfigurasi InfluxDB dapat menghemat ruang untuk menyimpan data dalam jangka waktu tertentu, secara otomatis berakhir dan menghapus semua data yang tidak diperlukan dari sistem.

Grafana merupakan perangkat open source untuk analisis dan visualisasi metrik. Grafana paling sering digunakan untuk memvisualisasikan data deret waktu untuk infrastruktur dan analitik aplikasi dan juga banyak digunakan di domain lain termasuk sensor industri, otomatisasi rumah, cuaca, dan kontrol proses. Grafana mendukung banyak storage backends

yang berbeda untuk data time series (Source Data).

2.2.4 Pengembangan Aplikasi Dekstop dengan PyQt

Pengembangan aplikasi desktop dengan PyQt merujuk pada proses pembuatan perangkat lunak yang memiliki antarmuka grafis (Graphical User Interface) dan berjalan di sistem operasi desktop seperti Windows, macOS, atau Linux, dengan menggunakan PyQt sebagai framework utamanya. PyQt sendiri adalah binding dari toolkit Qt untuk bahasa pemrograman Python, yang memungkinkan pengembang membuat aplikasi GUI yang interaktif, dan kaya fitur tanpa harus menggunakan bahasa C++ yang menjadi bahasa asli Qt.

Dalam proses pengembangannya, PyQt menyediakan berbagai komponen seperti tombol, menu, jendela, form input, hingga pengelolaan layout, yang semuanya dapat dikombinasikan untuk membentuk antarmuka pengguna yang kompleks. Selain itu, PyQt juga mendukung fitur lanjutan seperti multithreading, akses database, integrasi jaringan, dan dukungan multimedia, yang menjadikannya pilihan yang sangat fleksibel untuk membangun aplikasi desktop yang profesional dan responsif. Dengan memanfaatkan kemudahan sintaksis Python dan kekuatan Qt, pengembangan aplikasi desktop dengan PyQt menjadi solusi yang efisien dan produktif, baik untuk proyek berskala kecil maupun besar.

2.2.5 Teknologi Blockchain dalam Sistem Industri Pertanian

Blockchain merupakan teknologi *Distributed Ledger Technology* (DLT) yang memungkinkan pencatatan transaksi digital secara aman, transparan, dan terdesentralisasi. Dalam arsitektur blockchain, data dicatat dalam bentuk unit-unit yang disebut blok, yang saling terhubung secara kronologis melalui mekanisme hash kriptografis. Setiap blok memuat sekumpulan data transaksi, dan sekali data dicatat dalam blockchain, maka data tersebut tidak dapat diubah tanpa persetujuan mayoritas node dalam jaringan, menjadikan blockchain sebagai sistem yang immutable dan tahan manipulasi (Zheng *et al.*, 2018).

Konsep blockchain pertama kali diperkenalkan secara luas oleh Satoshi Nakamoto dalam publikasi "Bitcoin: A Peer-to-Peer Electronic Cash System" pada tahun 2008, yang mendemonstrasikan penerapan blockchain sebagai sistem pencatatan transaksi keuangan tanpa memerlukan otoritas pusat. Seiring perkembangan teknologi, blockchain tidak hanya terbatas pada sektor keuangan, tetapi telah merambah ke berbagai bidang, termasuk logistik, kesehatan, pendidikan, dan pertanian.

Dalam konteks pertanian, blockchain menawarkan solusi inovatif terhadap permasalahan klasik dalam sistem rantai pasok, seperti rendahnya transparansi, kesulitan pelacakan asal-usul produk, dan risiko pemalsuan data. Teknologi ini memungkinkan setiap entitas dalam rantai pasok mulai dari petani, pengepul, distributor, hingga pengecer untuk mencatat setiap aktivitas dan peristiwa penting ke dalam sistem yang dapat diaudit oleh semua pihak terkait (Kamilaris, 2019). Dengan demikian, blockchain mampu membangun trustless trust, yaitu kepercayaan yang tidak bergantung pada pihak ketiga, melainkan dijamin oleh algoritma konsensus dan verifikasi jaringan secara kolektif.

Selain itu, blockchain juga mendukung penerapan smart contract, yaitu program otomatis yang berjalan di atas jaringan blockchain dan dieksekusi berdasarkan logika kondisi tertentu. Dalam praktiknya, smart contract dapat digunakan untuk mengotomatiskan proses validasi data budidaya pertanian, penghitungan hasil panen, hingga pelepasan pembayaran secara otomatis setelah syarat tertentu terpenuhi. Pendekatan ini mengurangi kebutuhan akan

intervensi manual, mempercepat transaksi, dan mengurangi potensi konflik antar pihak (Casino *et al.*, 2019).

2.2.6 Smart Contract dan Web3

Smart contract merupakan salah satu komponen kunci dalam arsitektur blockchain modern yang memungkinkan otomatisasi proses transaksi atau eksekusi logika bisnis tanpa perlu keterlibatan pihak ketiga. Istilah smart contract pertama kali diperkenalkan oleh Nick Szabo pada tahun 1997 sebagai protokol komputer yang dapat secara otomatis menegakkan isi kontrak digital berdasarkan aturan yang telah ditentukan sebelumnya (Szabo, 1997). Kontrak cerdas ini dirancang untuk meniru logika dari perjanjian hukum konvensional, tetapi dengan pelaksanaan yang sepenuhnya otomatis dan terdesentralisasi.

Smart contract umumnya dijalankan pada platform blockchain yang mendukung Turing-complete programming, seperti Ethereum. Dalam ekosistem Ethereum, smart contract ditulis menggunakan bahasa pemrograman seperti Solidity, kemudian dikompilasi dan di-deploy ke jaringan blockchain melalui transaksi. Setelah berada di jaringan, kontrak tersebut akan dijalankan oleh Ethereum Virtual Machine (EVM) secara deterministik, sehingga hasil eksekusinya akan sama pada semua node dalam jaringan (Wood, 2014).

Keunggulan utama dari smart contract terletak pada sifatnya yang transparan, deterministik, dan tidak dapat diubah (immutable) setelah di-deploy. Hal ini memastikan bahwa semua pihak dalam sistem dapat memverifikasi logika kontrak, dan tidak ada satu entitas pun yang dapat memodifikasi perilaku kontrak secara sepihak. Dengan demikian, smart contract menjadi alat yang efektif untuk membangun kepercayaan dalam sistem yang bersifat terbuka dan terdistribusi.

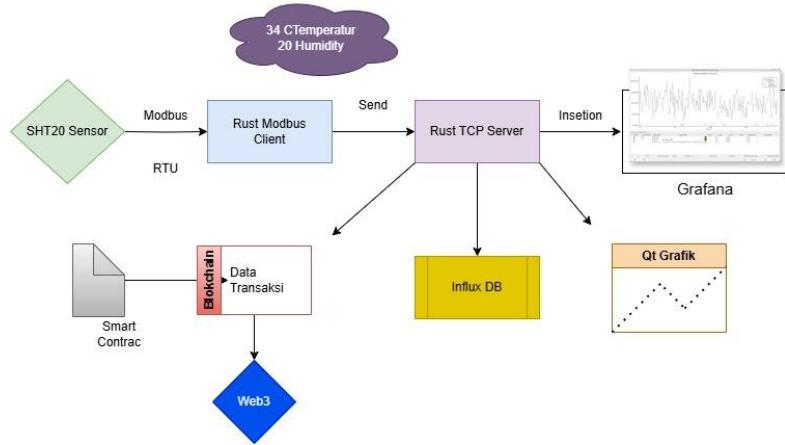
Dalam konteks pertanian cerdas (smart agriculture), smart contract dapat diintegrasikan dengan sistem IoT untuk menciptakan arsitektur otomasi yang efisien dan akuntabel. Misalnya, data lingkungan seperti suhu dan kelembaban yang dikumpulkan oleh sensor digital dapat dikirim secara berkala ke jaringan blockchain melalui antarmuka yang telah dikonfigurasi. Smart contract kemudian menjalankan logika kondisi berdasarkan parameter tersebut untuk melakukan validasi data.

Eksekusi logika ini memungkinkan sistem untuk secara otomatis mengklasifikasikan status lingkungan budidaya, seperti pada rumah jamur (kumbung), dan mencatatnya ke dalam blockchain secara permanen. Pencatatan ini tidak hanya berguna untuk pelacakan (traceability), tetapi juga dapat dijadikan dasar dalam pengambilan keputusan operasional, penilaian kualitas hasil panen, dan sebagai bukti validasi dalam rantai pasok (Zailani *et al.*, 2022).

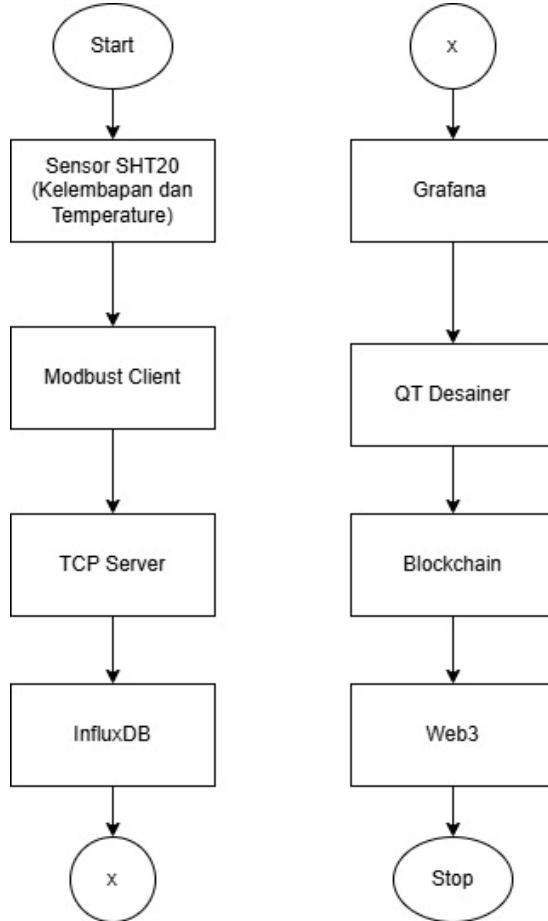
Agar dapat diakses oleh aplikasi eksternal seperti dashboard berbasis web atau perangkat mobile, smart contract berinteraksi dengan frontend melalui pustaka komunikasi blockchain seperti Web3.js (untuk JavaScript) atau Web3.py (untuk Python). Pustaka ini memungkinkan pengembang untuk mengirim transaksi, membaca data kontrak, serta menghubungkan antarmuka pengguna dengan data yang tersimpan di blockchain secara real-time (Web3 Foundation, 2021). Proses ini juga dapat diperkuat dengan middleware seperti The Graph untuk indexing data, serta IPFS untuk menyimpan data tambahan secara terdesentralisasi. Dengan menggabungkan teknologi smart contract dan blockchain, sistem pertanian dapat berevolusi menuju sistem digital yang sepenuhnya otomatis, aman, dan berbasis data. Implementasi ini sangat sesuai untuk mendukung transformasi digital sektor hortikultura, khususnya budidaya jamur tiram yang sensitif terhadap fluktuasi lingkungan.

BAB III METODOLOGI

3.1 Desain Sistem



3.2 Flowchart Sistem



Pada flowchart tersebut menunjukkan sebuah alur kerja dari sebuah sistem pemantauan suhu dan kelembapan secara otomatis dan canggih. Proses dimulai dari sensor SHT20 yang berfungsi untuk mengukur kelembapan dan suhu udara. Data dari sensor ini dibaca menggunakan program Modbus Client, lalu dikirim ke komputer melalui TCP Server. Setelah itu, data disimpan ke dalam InfluxDB, yaitu tempat penyimpanan khusus untuk data yang terus berubah seiring waktu, seperti data sensor. Data ini kemudian bisa ditampilkan dalam dua cara.

Cara pertama adalah dengan menggunakan Grafana, yaitu aplikasi yang menampilkan data dalam bentuk grafik agar mudah dipahami.

Cara kedua adalah lewat aplikasi buatan sendiri menggunakan Qt Designer, yang nantinya juga bisa dihubungkan ke teknologi blockchain untuk mencatat data secara aman dan transparan. Terakhir, sistem ini bisa terhubung ke Web3, yaitu teknologi internet masa kini yang lebih terbuka dan aman. Jadi, keseluruhan sistem ini memungkinkan kita memantau suhu dan kelembapan secara real-time, menyimpan data, menampilkannya secara visual, dan menjaganya tetap aman.

3.2.1 Program Server

Main.rs

```
// sistem_fermentasi/crates/tcp-server/src/main.rs

// BARU: Imports untuk ethers, dotenv, dan env use ethers::prelude::*;

use tokio::net::{TcpListener, TcpStream}; use tokio::io::{AsyncReadExt, BufReader}; use
serde::Deserialize; use influxdb2::Client as InfluxClient; use influxdb2::models::DataPoint;
use std::sync::Arc; use std::error::Error; use chrono::{DateTime, Utc};

// BARU: Generate Rust binding dari file ABI abigen!( SensorRegistry,
"./abi/Monitoring.json", event_derives(serde::Deserialize, serde::Serialize) );

// Struct data yang diterima (tidak berubah) #[derive(Deserialize, Debug, Clone)] struct
SensorData { timestamp: String, sensor_id: String, location: String, process_stage: String,
temperature_celsius: f64, humidity_percent: f64, }

#[tokio::main] async fn main() -> Result<(), Box> { // BARU: Muat variabel dari file .env
dotenv().ok();

    // --- Konfigurasi InfluxDB (Tidak berubah) ---
    let influx_url = "http://localhost:8086";
    let influx_org = "Jamur";
    let influx_bucket = "Tiram2";
    let influx_token =
"AMdJrRUmxq1PdZa4JA3CHg8QCEz3_JNxp_9dD0uFCnFrVtDOJnSBbSaHXrgeWMIW_
0QIbz12aisMXPZXWaEmGA==";
    let influx_client = Arc::new(InfluxClient::new(influx_url,
influx_org, influx_token));

    // --- BARU: Konfigurasi dan inisialisasi koneksi Blockchain ---
    let rpc_url = env::var("RPC_URL").expect("RPC_URL harus diset");
    let private_key = env::var("PRIVATE_KEY").expect("PRIVATE_KEY
harus diset");
    let contract_address =
env::var("CONTRACT_ADDRESS").expect("CONTRACT_ADDRESS harus
diset");

    let provider = Provider::try_from(rpc_url)?;
    let wallet: LocalWallet =
private_key.parse()? .with_chain_id(31337u64);
    let eth_client = SignerMiddleware::new(provider, wallet.clone());
}
```

```

let contract =
SensorRegistry::new(contract_address.parse::<Address>()?,,
Arc::new(eth_client));

// BARU: Bungkus kontrak dalam Arc agar bisa di-share antar thread
let contract_arc = Arc::new(contract);
println!("[Blockchain] Terhubung ke smart contract di alamat: {},
contract_address);

// --- Inisialisasi TCP Server (Tidak berubah) ---
let addr = "127.0.0.1:7878";
let listener = TcpListener::bind(&addr).await.unwrap();
println!("[Server] Berjalan pada {}, siap menerima data...",,
addr);

loop {
    let (socket, _) = listener.accept().await.unwrap();
    // UBAH: Clone Arc untuk InfluxDB dan Blockchain
    let influx_clone = Arc::clone(&influx_client);
    let contract_clone = Arc::clone(&contract_arc);
    let bucket_clone = influx_bucket.to_string();

    tokio::spawn(async move {
        // UBAH: Kirim kedua clone ke handle_connection
        if let Err(e) = handle_connection(socket, influx_clone,
contract_clone, &bucket_clone).await {
            eprintln!("🔴 [Server] Gagal memproses koneksi: {}",,
e);
        }
    });
}
}

// UBAH: Tambahkan parameter contract ke fungsi async fn handle_connection( stream:
TcpStream, influx_client: Arc, contract: Arc<SensorRegistry<SignerMiddleware<Provider,
Walletk256::ecdsa::SigningKey>>>, bucket: &str, ) -> Result<(), Box> { let mut buffer =
Vec::new(); let mut reader = BufReader::new(stream); reader.read_to_end(&mut
buffer).await?;

let data: SensorData = serde_json::from_slice(&buffer)?;
println!("👉 [Server] Data diterima: {:?}", data);

// --- Bagian 1: Simpan ke InfluxDB (Tidak berubah) ---
tokio::spawn({
    let data_clone = data.clone();
    let influx_client_clone = Arc::clone(&influx_client);
    let bucket_clone = bucket.to_string();
    async move {
        if let Err(e) = save_to_influxdb(&data_clone,
influx_client_clone, &bucket_clone).await {
            eprintln!("🔴 [InfluxDB] Gagal menyimpan data: {},
{}", e);
        }
    }
});
}

```

```

        }
    }

}) ;

// --- BARU: Bagian 2: Kirim Transaksi ke Blockchain ---
println!("✉️ [Blockchain] Mempersiapkan transaksi...");

// Konversi tipe data untuk smart contract
// <<< DIUBAH: Baris ini dihapus karena Smart Contract tidak
meminta timestamp.
// let timestamp_onchain =
U256::from_dec_str(&data.timestamp.parse::<DateTime<Utc>>() ? .
timestamp().to_string();;

let location_onchain = data.location.clone();
// Kita kalikan 10 untuk menyimpan 1 angka desimal sebagai
integer
let temperature_onchain =
I256::from((data.temperature_celsius * 10.0) as i64);
let humidity_onchain = U256::from((data.humidity_percent *
10.0) as u64);

// <<< DIUBAH: Pemanggilan fungsi disesuaikan agar hanya
mengirim 3 argumen
// sesuai urutan di file Monitoring.sol (location,
temperature, humidity).
let tx = contract.add_reading(
    location_onchain,
    temperature_onchain,
    humidity_onchain
).send().await?.await?;

println!("☑️ [Blockchain] Data berhasil dicatat! Hash: {:?}", ,
tx.unwrap().transaction_hash);

Ok(())
}

// BARU: Fungsi terpisah untuk menyimpan ke InfluxDB agar lebih rapi async fn
save_to_influxdb(data: &SensorData, client: Arc, bucket: &str) -> Result<(), Box> {
    let timestamp = data.timestamp.parse::<DateTime>()?;

    let point = DataPoint::builder("environment_monitoring")
        .tag("sensor_id", data.sensor_id.clone())
        .tag("location", data.location.clone())
        .tag("process_stage", data.process_stage.clone())
        .field("temperature_celsius", data.temperature_celsius)
        .field("humidity_percent", data.humidity_percent)
}

```

```
.timestamp(timestamp.timestamp_nanos_opt().unwrap_or_default())
    .build()?;

client.write(bucket,
futures::stream::iter(vec![point])).await?;
println!("💾 [InfluxDB] Data berhasil disimpan.");
Ok(())

}
```

```
Cargo.toml
[package]
name = "tcp-server"
version = "0.1.0"
edition = "2021"

[dependencies]
tokio = { version = "1", features = ["full"] }
serde = { version = "1.0", features = ["derive"] }
serde_json = "1.0"
influxdb2 = "0.3"
futures = "0.3"
chrono = "0.4"
ethers = { version = "2.0", features = ["legacy"] } # <--  
BARU
dotenv = "0.15"
```

BAB IV HASIL DAN PEMBAHASAN

4.1 Pengambilan Data

Alur pengambilan data dalam sistem monitoring ini dimulai dari sensor lingkungan SHT20 yang mengukur parameter suhu dan kelembaban di area produksi jamur tiram secara real-time. Sensor ini terhubung ke perangkat mikrokontroler atau gateway yang menjalankan program Modbus Client berbasis bahasa pemrograman Rust, yang secara berkala membaca data sensor menggunakan protokol Modbus RTU. Setelah data diperoleh, informasi seperti suhu, kelembaban, waktu pencatatan, ID sensor, lokasi, dan tahap proses dikemas dalam format JSON.

Kemudian dikirim melalui koneksi TCP ke Rust TCP Server yang berjalan pada alamat 127.0.0.1:7878. Server ini menerima, memproses, dan menyimpan data tersebut ke dalam database time-series InfluxDB untuk keperluan historis dan analisis, serta secara paralel mencatat data penting ke dalam smart contract di jaringan blockchain lokal melalui endpoint Hardhat (<http://127.0.0.1:8545>). Data yang sudah tersimpan kemudian dapat divisualisasikan menggunakan Grafana dan aplikasi Web3 (DApp) untuk menampilkan data dari blockchain secara transparan, serta ditampilkan juga di aplikasi desktop Qt untuk monitoring lokal. Seluruh alur ini memastikan bahwa data sensor yang dikumpulkan bukan hanya tersimpan dengan baik dan divisualisasikan secara real-time, tetapi juga dicatat secara permanen dan terverifikasi di blockchain sebagai bukti keaslian dan keamanan data.

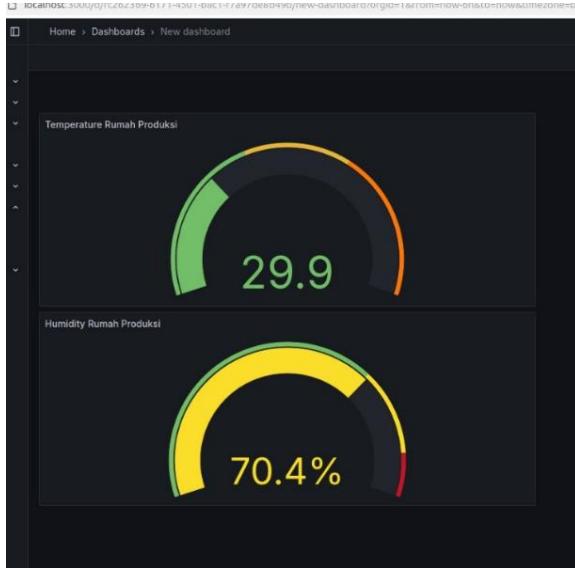
4.2 Tampilan Data Explorer InfluxDB



Gambar tersebut menunjukkan antarmuka Data Explorer dari InfluxDB yang sedang menampilkan grafik hasil pemantauan parameter lingkungan menggunakan sensor SHT20 dengan ID “SHT20-PascaPanen-001” dalam proses produksi jamur tiram pada lokasi “Produksi Jamur”. Data yang diambil berasal dari bucket bernama “Tiram2” dan difilter berdasarkan pengukuran “environment_monitoring”, dengan dua parameter utama yaitu kelembaban udara (humidity_percent) dan suhu (temperature_celsius).

Grafik memperlihatkan nilai kelembaban yang stabil berada pada kisaran sekitar 70%, sementara suhu relatif konstan pada kisaran 30°C dengan sedikit fluktuasi tajam sesaat, yang dapat diindikasikan sebagai gangguan sensor atau perubahan suhu mendadak. Data diambil dalam rentang waktu, dengan interval waktu (window period) 5 detik, dan menggunakan fungsi agregat “mean” untuk merata-ratakan nilai dalam setiap jendela waktu. Hal ini menunjukkan sistem monitoring berjalan baik dalam mengamati kondisi lingkungan pada tahap pasca panen jamur tiram, memungkinkan pengguna untuk mengambil keputusan cepat terhadap perubahan lingkungan.

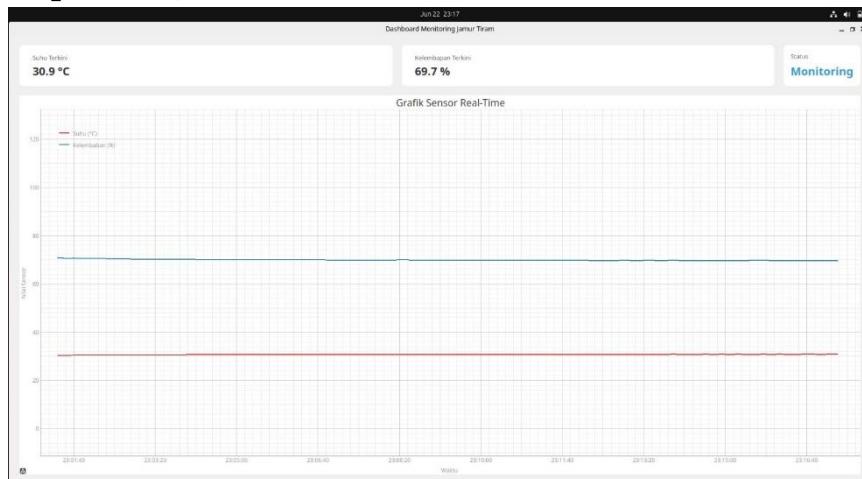
4.2 Tampilan Dashboard Grafana



Gambar yang ditampilkan merupakan dashboard visualisasi data dari Grafana yang digunakan untuk memantau kondisi lingkungan di dalam rumah produksi jamur tiram. Terdapat dua panel utama yang menyajikan data suhu dan kelembaban secara real-time dalam bentuk gauge atau pengukur analog digital yang intuitif. Panel pertama menunjukkan nilai suhu udara sebesar 29.9°C yang berada dalam zona hijau, mengindikasikan bahwa kondisi suhu di dalam ruangan berada dalam rentang optimal untuk pertumbuhan jamur.

Sementara itu, panel kedua menunjukkan nilai kelembaban udara sebesar 70.4%, yang juga berada di zona kuning-hijau, menunjukkan kelembaban cukup tinggi namun masih dalam batas toleransi untuk produksi jamur. Visualisasi ini memungkinkan operator atau pengguna untuk secara cepat menilai kondisi aktual lingkungan dan melakukan tindakan korektif bila parameter mulai mendekati batas kritis. Dengan pemantauan semacam ini, sistem membantu menjaga kestabilan mikroklimat yang penting dalam proses budidaya jamur agar tetap produktif dan efisien.

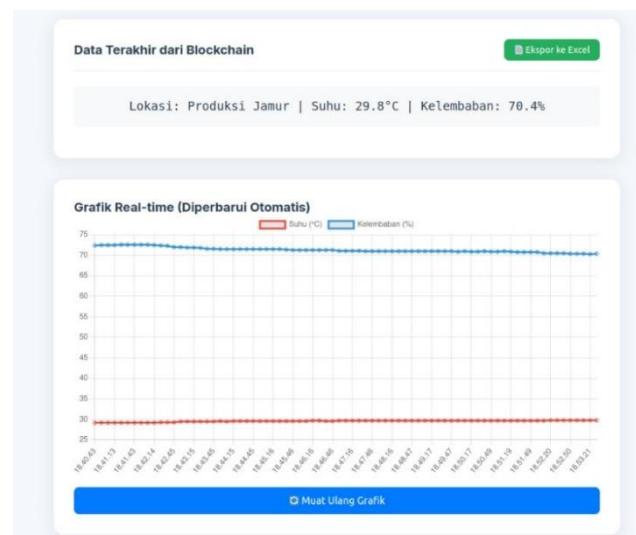
4.3 Tampilan Aplikasi QT



Gambar tersebut merupakan tampilan aplikasi desktop yang dibangun menggunakan Qt dengan fungsi utama sebagai dashboard monitoring real-time untuk budidaya jamur tiram.

Antarmuka ini menyajikan dua parameter penting, yaitu suhu dan kelembapan lingkungan, yang ditampilkan secara langsung dari pembacaan sensor. Nilai suhu terkini tercatat sebesar 30.9°C dan kelembapan sebesar 69.7%, ditampilkan dalam bentuk angka besar di bagian atas untuk memudahkan pembacaan cepat. Di bawahnya, terdapat grafik waktu nyata (real-time) yang menunjukkan tren nilai sensor seiring waktu, di mana garis merah mewakili suhu dan garis biru mewakili kelembapan. Kedua grafik menunjukkan kestabilan lingkungan dengan sedikit fluktuasi, menandakan bahwa sistem monitoring bekerja secara konsisten dalam menjaga dan memantau kondisi optimal untuk pertumbuhan jamur. Status “Monitoring” di sisi kanan atas mengindikasikan bahwa sistem sedang aktif melakukan pemantauan. Aplikasi ini sangat bermanfaat untuk pengguna karena memberikan umpan balik langsung dari sensor ke antarmuka pengguna, memungkinkan pemantauan lingkungan produksi secara terus-menerus dan efisien.

4.4 Tampilan Web3 Sensor Dashboard



Gambar tersebut menampilkan antarmuka web berbasis Web3 yang secara real-time mengambil dan menampilkan data lingkungan dari blockchain, menunjukkan hasil integrasi teknologi monitoring sensor dengan sistem desentralisasi. Di bagian atas, terdapat ringkasan data terbaru yang ditarik langsung dari blockchain, mencakup lokasi pemantauan yaitu “Produksi Jamur” dengan nilai suhu sebesar 29.8°C dan kelembaban udara 70.4%, yang menunjukkan kondisi lingkungan yang stabil dan sesuai untuk budidaya jamur.

Di bawahnya, terdapat grafik real-time yang diperbarui secara otomatis, dengan garis merah mewakili suhu dan garis biru mewakili kelembaban. Kedua parameter tersebut ditampilkan dengan konsistensi dan kestabilan, memperlihatkan bahwa sensor bekerja secara akurat dan integrasi datanya ke dalam blockchain berhasil dilakukan dengan baik. Tombol “Ekspor ke Excel” juga disediakan untuk memungkinkan pengguna mengunduh data historis untuk keperluan analisis lanjutan. Sistem ini mencerminkan penerapan teknologi Web3 dalam bidang pertanian cerdas, di mana data sensor disimpan di blockchain untuk menjamin transparansi, keaslian data, dan mencegah manipulasi oleh pihak yang tidak berwenang.

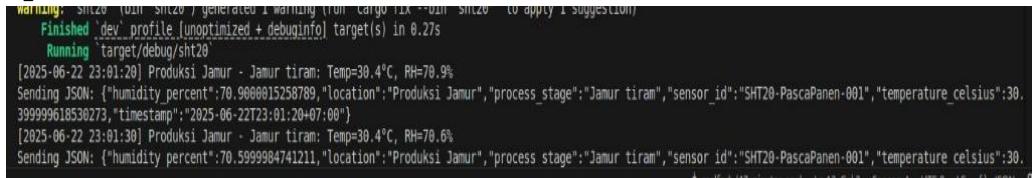
4.5 Tampilan Eksekusi Perintah Rust TCP Server



```
nyoman16@nyoman16-VirtualBox:~/bismillah_ready/rust_tcp_blockchain$ cargo run
   Finished dev profile [unoptimized + debuginfo] target(s) in 1.29s
     Running `target/debug/tcp-server`
[Blockchain] Terhubung ke smart contract di alamat: 0x5f0882315678fech367f032893f642f64108aa3
[Server] Berjalan pada 127.0.0.1:7878, siap menerima data...
[Server] Data diterima: SensorData { timestamp: "2025-06-22T23:13:16+07:00", sensor_id: "SHT20-PascaPanen-001", location: "Produksi Jamur", process_stage: "Jamur tiram", temperature_celsius: 30.799999237069547, humidity_percent: 69.6999964824219 }
[Blockchain] Mempersiapkan transaksi...
[InfluxDB] Data berhasil disimpan.
[Blockchain] Data berhasil dicatat! Hash: 0x5d4cc43c5802599e9d22e3940ef802b113f219c2dad7714f99f51edec3eebf77
```

Gambar tersebut menunjukkan hasil eksekusi Rust TCP Server yang berhasil berjalan pada alamat 127.0.0.1:7878 dan menerima data sensor lingkungan secara real-time. Data yang diterima mencakup ID sensor “SHT20-PascaPanen-001”, lokasi “Produksi Jamur”, suhu sebesar 30.79°C, dan kelembapan sebesar 69.7%. Data ini kemudian dicatat ke dalam InfluxDB dan secara otomatis dikirim ke smart contract blockchain, yang ditandai dengan keberhasilan pencatatan data dengan hash transaksi. Hal ini membuktikan bahwa sistem Rust TCP Server berfungsi sebagai penghubung yang efisien antara perangkat sensor dan pencatatan terverifikasi di blockchain.

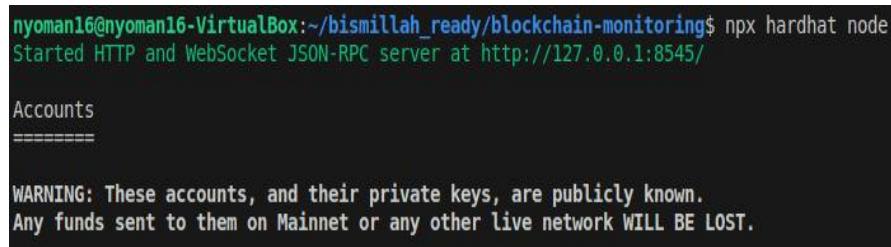
4.6 Tampilan Eksekusi Perintah Modbus Client



```
Warning: sht20 (use sht20 / generated 1 warning) (on cargo fix --fix sht20 to apply 1 suggestion)
  Finished dev profile [unoptimized + debuginfo] target(s) in 0.27s
    Running `target/debug/sht20`
[2025-06-22 23:01:20] Produksi Jamur - Jamur tiram: Temp=30.4°C, RH=70.9%
Sending JSON: {"humidity_percent": 70.9, "location": "Produksi Jamur", "process_stage": "Jamur tiram", "sensor_id": "SHT20-PascaPanen-001", "temperature_celsius": 30.399999618530273, "timestamp": "2025-06-22T23:01:20+07:00"}
[2025-06-22 23:01:30] Produksi Jamur - Jamur tiram: Temp=30.4°C, RH=70.6%
Sending JSON: {"humidity_percent": 70.5999984741211, "location": "Produksi Jamur", "process stage": "Jamur tiram", "sensor id": "SHT20-PascaPanen-001", "temperature celsius": 30.399999618530273, "timestamp": "2025-06-22T23:01:30+07:00"}
```

Gambar tersebut menunjukkan hasil eksekusi program modbus_client berbasis Rust yang berhasil membaca data dari sensor SHT20 melalui protokol Modbus dan mengirimkannya dalam format JSON. Terlihat bahwa suhu terukur sebesar 30.4°C dan kelembapan 70.9% serta 70.6% pada dua waktu berbeda, dikirim dengan lengkap mencakup informasi lokasi, tahap proses, ID sensor, dan timestamp. Data JSON ini kemudian siap untuk dikirim ke server atau dicatat ke sistem lain seperti InfluxDB atau blockchain, menandakan bahwa koneksi sensor via Modbus telah berjalan dengan baik.

4.7 Tampilan Smart Contract Blockchain



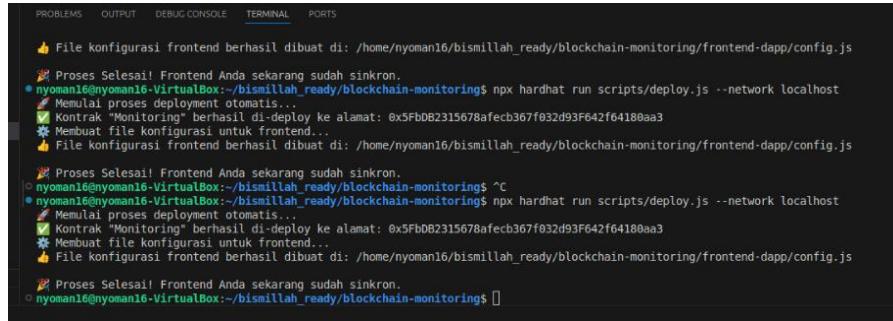
```
nyoman16@nyoman16-VirtualBox:~/bismillah_ready/blockchain-monitoring$ npx hardhat node
Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/

Accounts
=====

WARNING: These accounts, and their private keys, are publicly known.
Any funds sent to them on Mainnet or any other live network WILL BE LOST.
```

Gambar tersebut menunjukkan hasil eksekusi perintah npx hardhat node, yang berfungsi untuk menjalankan jaringan blockchain lokal menggunakan Hardhat, sebuah framework pengembangan Ethereum yang populer. Saat perintah ini dijalankan, sistem akan membuat jaringan simulasi yang berjalan secara lokal pada alamat http://127.0.0.1:8545, lengkap dengan beberapa akun Ethereum beserta private key-nya untuk keperluan pengujian dan pengembangan smart contract. Dengan node lokal ini, pengembang dapat menguji logika smart contract secara aman dan cepat sebelum melakukan deploy ke jaringan testnet atau mainnet, sehingga menjadi fondasi penting dalam pembangunan sistem berbasis blockchain seperti yang digunakan dalam proyek monitoring lingkungan jamur tiram.

4.8 Tampilan Hasil Run Deploy



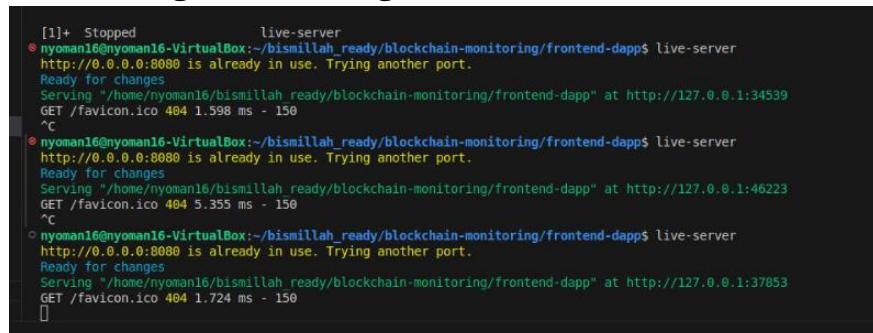
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

👉 File konfigurasi frontend berhasil dibuat di: /home/nyoman16/bismillah_ready/blockchain-monitoring/frontend-dapp/config.js
👉 Proses Selesai! Frontend Anda sekarang sudah sinkron.
nyoman16@nyoman16-VirtualBox:~/bismillah_ready/blockchain-monitoring$ npx hardhat run scripts/deploy.js --network localhost
Memulai proses deployment otomatis...
Kontrak "Monitoring" berhasil di-deploy ke alamat: 0x5FbDB2315678afebcb367f032d93F642f64180aa3
Membuat file konfigurasi untuk frontend...
👉 File konfigurasi frontend berhasil dibuat di: /home/nyoman16/bismillah_ready/blockchain-monitoring/frontend-dapp/config.js
👉 Proses Selesai! Frontend Anda sekarang sudah sinkron.
nyoman16@nyoman16-VirtualBox:~/bismillah_ready/blockchain-monitoring$ npx hardhat run scripts/deploy.js --network localhost
Memulai proses deployment otomatis...
Kontrak "Monitoring" berhasil di-deploy ke alamat: 0x5FbDB2315678afebcb367f032d93F642f64180aa3
Membuat file konfigurasi untuk frontend...
👉 File konfigurasi frontend berhasil dibuat di: /home/nyoman16/bismillah_ready/blockchain-monitoring/frontend-dapp/config.js
👉 Proses Selesai! Frontend Anda sekarang sudah sinkron.
nyoman16@nyoman16-VirtualBox:~/bismillah_ready/blockchain-monitoring$
```

Gambar tersebut menunjukkan hasil eksekusi skrip deploy.js menggunakan Hardhat, yang digunakan untuk mendesain dan menyebarkan (deploy) smart contract ke jaringan lokal blockchain, sebagai bagian dari sistem monitoring terintegrasi Web3. Pada terminal terlihat bahwa proses deployment berhasil dilakukan, dengan kontrak pintar bernama “Monitoring” berhasil di-deploy pada alamat tertentu di jaringan lokal (--network localhost).

Informasi ini akan digunakan oleh aplikasi frontend DApp agar bisa mengambil data sensor lingkungan (suhu dan kelembaban) dari database InfluxDB dan mencatatnya ke dalam blockchain untuk menjamin keaslian dan transparansi data. Proses ini mencerminkan integrasi menyeluruh antara backend sensor, penyimpanan database time-series (InfluxDB), dan teknologi Web3 melalui kontrak pintar untuk menciptakan sistem monitoring yang transparan dan tidak dapat dimanipulasi.

4.9 Tampilan Ketika Ingin Terhubung ke Web3



```
[1]+ Stopped                  live-server
nyoman16@nyoman16-VirtualBox:~/bismillah_ready/blockchain-monitoring/frontend-dapp$ live-server
http://0.0.0.0:8080 is already in use. Trying another port.
Ready for changes
Serving "/home/nyoman16/bismillah_ready/blockchain-monitoring/frontend-dapp" at http://127.0.0.1:34539
GET /favicon.ico 404 1.598 ms - 150
^C
nyoman16@nyoman16-VirtualBox:~/bismillah_ready/blockchain-monitoring/frontend-dapp$ live-server
http://0.0.0.0:8080 is already in use. Trying another port.
Ready for changes
Serving "/home/nyoman16/bismillah_ready/blockchain-monitoring/frontend-dapp" at http://127.0.0.1:46223
GET /favicon.ico 404 5.355 ms - 150
^C
nyoman16@nyoman16-VirtualBox:~/bismillah_ready/blockchain-monitoring/frontend-dapp$ live-server
http://0.0.0.0:8080 is already in use. Trying another port.
Ready for changes
Serving "/home/nyoman16/bismillah_ready/blockchain-monitoring/frontend-dapp" at http://127.0.0.1:37853
GET /favicon.ico 404 1.724 ms - 150
```

Gambar tersebut memperlihatkan proses menjalankan live server untuk menampilkan antarmuka Web3 berbasis frontend DApp (Decentralized Application) yang berada dalam folder frontend-dapp. File konfigurasi config.json yang ditampilkan di editor memuat parameter penting seperti alamat TCP server, batas suhu dan kelembaban minimum serta maksimum, serta kredensial untuk mengakses database InfluxDB yang digunakan sebagai sumber data sensor lingkungan.

Di terminal, perintah live-server dijalankan untuk menyajikan aplikasi web secara lokal, namun karena port 8080 sudah digunakan, sistem mencoba port lain secara otomatis hingga berhasil menyajikan aplikasi pada alamat alternatif seperti http://127.0.0.1:34539, 46223, dan akhirnya 37853. Keberhasilan server menampilkan web aplikasi ini memungkinkan pengguna untuk berinteraksi dengan data sensor yang telah dicatat ke dalam blockchain, melihat grafik dan nilai parameter lingkungan secara real-time, serta menjamin keaslian dan transparansi data melalui teknologi desentralisasi.

BAB V KESIMPULAN

5.1 Kesimpulan

Berdasarkan serangkaian hasil pengujian dan implementasi sistem, dapat disimpulkan bahwa proyek monitoring lingkungan untuk budidaya jamur tiram berhasil mengintegrasikan berbagai komponen teknologi secara efektif. Data dari sensor suhu dan kelembaban yang terpasang di lingkungan produksi berhasil diakuisisi melalui komunikasi Modbus menggunakan bahasa pemrograman Rust. Data tersebut kemudian dikirim ke server melalui protokol TCP, diproses, dan dicatat ke dalam sistem penyimpanan berbasis time-series database (InfluxDB) serta ke dalam blockchain melalui smart contract. Hal ini membuktikan bahwa pipeline data dari perangkat keras hingga pencatatan digital dapat berjalan dengan baik dan terstruktur.

Selain keberhasilan pada sisi backend, sistem juga menunjukkan performa baik dalam hal visualisasi dan antarmuka pengguna. Data yang terekam di InfluxDB dapat divisualisasikan dengan baik melalui Grafana dalam bentuk gauge dan grafik time-series, memberikan informasi yang jelas dan mudah dipahami terkait kondisi suhu dan kelembaban di rumah produksi. Antarmuka Web3 yang dikembangkan dengan DApp mampu menampilkan data dari blockchain secara real-time, termasuk fitur ekspor data dan pembaruan otomatis. Tidak hanya itu, aplikasi desktop berbasis Qt juga memberikan tampilan yang ringkas namun informatif untuk pengguna lokal, yang mendukung operasional di lapangan tanpa koneksi internet.

Keseluruhan proses ini juga telah membuktikan bahwa pencatatan data ke blockchain memberikan manfaat signifikan dalam aspek keaslian data dan transparansi. Dengan menyimpan data lingkungan ke dalam smart contract, setiap perubahan kondisi terekam secara permanen dan tidak dapat dimanipulasi, menjadikan sistem ini sangat ideal untuk bidang pertanian presisi, audit produksi, dan sertifikasi mutu. Penggunaan Rust dalam pengembangan backend memberikan performa tinggi dan keandalan, sementara frontend yang fleksibel memungkinkan berbagai pihak baik petani, teknisi, maupun pemangku kebijakan untuk mengakses informasi dengan mudah dan akurat.

5.2 Saran

Adapun saran yang dapat dilakukan untuk penelitian selanjutnya adalah sebagai berikut.

- 1) Tambahkan sistem peringatan otomatis seperti SMS, email, atau notifikasi aplikasi ketika suhu atau kelembaban melampaui ambang batas, agar operator dapat segera melakukan tindakan korektif.
- 2) Terapkan logika pemrosesan awal di sisi perangkat (misalnya menggunakan Raspberry Pi) untuk mengurangi beban server dan mempercepat respon terhadap kondisi kritis.
- 3) Kembangkan fitur multi-lokasi monitoring yang memungkinkan sistem digunakan untuk banyak rumah produksi sekaligus, dengan identifikasi data berdasarkan ID lokasi secara otomatis.
- 4) Buat aplikasi mobile berbasis Android/iOS agar monitoring dapat dilakukan secara fleksibel kapan pun dan di mana pun.

DAFTAR PUSTAKA

- Gunawan, D. & Kurniawan, B., 2020. Implementasi Sensor SHT20 untuk Sistem Monitoring Kelembaban Berbasis IoT. Prosiding Seminar Nasional Teknologi Informasi dan Aplikasinya, 7(1), pp. 89–95.
- Nugroho, A.P. & Sari, N.F., 2019. Analisis Pengaruh Suhu dan Kelembaban terhadap Produksi Jamur Tiram. Agritech, 39(3), pp. 175–182.
- Purwanto, H., 2020. Manajemen Rantai Pasok Produk Hortikultura Berbasis Teknologi Informasi. Jurnal Agribisnis Terapan, 7(4), pp. 288–296.
- Putra, R.D. & Widodo, A.P., 2023. Smart Contract untuk Validasi Data IoT pada Blockchain Ethereum. Jurnal Sistem Informasi, 19(2), pp. 89–100.
- Rachman, A. & Lestari, W., 2021. Optimasi Lingkungan Mikro pada Fase Fruktifikasi Jamur Tiram. Jurnal Hortikultura Tropika, 10(1), pp. 22–29.
- Santoso, R., et al., 2021. Pengaruh Variasi Lingkungan Mikro terhadap Kualitas Jamur Konsumsi. Jurnal Agroteknologi Tropika, 10(1), pp. 20–28.
- Sensirion AG, 2019. Humidity and Temperature Sensor SHT20 Datasheet. Available at: <https://www.sensirion.com/file/datasheet-sht20> [Accessed 18 June 2025].
- Suryaningsih, E., 2018. Pengaruh Suhu dan Kelembaban terhadap Pertumbuhan Jamur Tiram. Jurnal Agribisnis dan Teknologi Pertanian, 5(2), pp. 101–108.
- Wahyuni, L. & Prasetyo, E., 2020. Potensi Jamur Tiram sebagai Sumber Pangan Fungsional. Jurnal Hortikultura Indonesia, 13(2), pp. 134–142.

LAMPIRAN

Lampiran 1. Pembuatan Program Client Sensor

```
Main.rs
```

```
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, Box> { 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, // Ubah dari Utc ke Local ) ->
Result<(), Box> { 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> { let sensor_id
= "SHT20-PascaPanen-001"; let location = "Produksi Jamur"; let
process_stage = "Jamur tiram";

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;
}
}

```

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"

```

Lampiran 2. Pembuatan GUI Dekstop

Main.py

```

import sys from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout,
QHBoxLayout, QPushButton, QLabel, QGridLayout, QFrame from PyQt5.QtGui import
 QFont, QPixmap from PyQt5.QtCore import Qt, QTimer import pyqtgraph as pg from
influxdb_client import InfluxDBClient from datetime import datetime import pandas as pd

```

--- BARU: Kelas khusus untuk menampilkan waktu di sumbu X ---

```

class TimeAxisItem(pg.AxisItem): def init(self, *args, **kwargs): super().init(*args,
**kwargs) self.setLabel(text='Waktu', units=None) self.enableAutoSIPrefix(False)

def tickStrings(self, values, scale, spacing):
    # Mengonversi timestamp (float) menjadi format string HH:mm:ss
    return [datetime.fromtimestamp(value).strftime('%H:%M:%S') for
value in values]

class InfluxMonitor(QWidget): def init(self): super().init() # Inisialisasi data terlebih dahulu
self.last_temp = "N/A" self.last_humidity = "N/A" self.status_text = "Idle" self.status_color
= "grey"

    self.initUI()
    self.connectInfluxDB()

    # BARU: Timer untuk refresh otomatis setiap 5 detik
    self.auto_refresh_timer = QTimer(self)
    self.auto_refresh_timer.timeout.connect(self.loadData)
    self.auto_refresh_timer.start(5000) # 5000 ms = 5 detik

    self.loadData() # Muat data pertama kali saat aplikasi dimulai

def initUI(self):
    self.setWindowTitle("Dashboard Monitoring Jamur Tiram")
    self.setGeometry(100, 100, 900, 700) # Ubah ukuran window jika perlu

    # Atur warna latar belakang utama
    self.setStyleSheet("background-color: #f0f0f0;")

    layout = QVBoxLayout()
    layout.setContentsMargins(20, 20, 20, 20)
    layout.setSpacing(20)

```

```

# --- BARU: Membuat Header Panel Atas yang Menarik ---
header_frame = QFrame()
header_frame.setLayout(QHBoxLayout())
header_frame.layout().setContentsMargins(0,0,0,0)
header_frame.layout().setSpacing(20)

# Panel Suhu
self.temp_panel = self.create_info_panel(
    "Suhu Terkini",
    self.last_temp,
    "temperature-icon.png"
)
# Panel Kelembapan
self.humidity_panel = self.create_info_panel(
    "Kelembapan Terkini",
    self.last_humidity,
    "humidity-icon.png"
)
# Panel Status
self.status_panel = self.create_status_panel("Status",
self.status_text)

header_frame.layout().addWidget(self.temp_panel)
header_frame.layout().addWidget(self.humidity_panel)
header_frame.layout().addWidget(self.status_panel)

layout.addWidget(header_frame)

# --- Grafik ---
# UBAH: Gunakan TimeAxisItem untuk sumbu 'bottom' (X)
axis_items = {'bottom': TimeAxisItem(orientation='bottom')}
self.graphWidget = pg.PlotWidget(axisItems=axis_items)
self.graphWidget.setBackground('w')
self.graphWidget.addLegend()
self.graphWidget.showGrid(x=True, y=True)
self.graphWidget.getPlotItem().getAxis('left').setLabel('Nilai Sensor')
self.graphWidget.setTitle("Grafik Sensor Real-Time",
color='#444', size='16pt')

self.temp_plot = self.graphWidget.plot(name="Suhu (°C)",
pen=pg.mkPen('#d62728', width=2)) # Warna merah
self.humidity_plot = self.graphWidget.plot(name="Kelembaban (%)",
pen=pg.mkPen('#1f77b4', width=2)) # Warna biru

layout.addWidget(self.graphWidget)
self.setLayout(layout)

# --- BARU: Fungsi untuk membuat panel info yang stylish ---
def create_info_panel(self, title_text, value_text, icon_path):
    panel = QFrame()
    panel.setStyleSheet("""
        QFrame {
            background-color: white;
            border-radius: 8px;
    """)

```

```

        }
    """
)
panel.setFrameShape(QFrame.StyledPanel)
panel_layout = QBoxLayout(panel)
panel_layout.setContentsMargins(15, 15, 15, 15)

icon_label = QLabel()
pixmap = QPixmap(icon_path)
if not pixmap.isNull():
    icon_label.setPixmap(pixmap.scaled(48, 48,
Qt.KeepAspectRatio, Qt.SmoothTransformation))
panel_layout.addWidget(icon_label)

text_layout = QVBoxLayout()
text_layout.setSpacing(0)

title_label = QLabel(title_text)
title_label.setFont(QFont('Segoe UI', 10))
title_label.setStyleSheet("color: #666;")

value_label = QLabel(value_text)
value_label.setFont(QFont('Segoe UI', 18, QFont.Bold))
value_label.setStyleSheet("color: #333;")

text_layout.addWidget(title_label)
text_layout.addWidget(value_label)

panel_layout.addLayout(text_layout)
panel_layout.addStretch()

# Simpan referensi ke value_label agar bisa diupdate
if "Suhu" in title_text:
    self.temp_value_label = value_label
elif "Kelembapan" in title_text:
    self.humidity_value_label = value_label

return panel

# --- BARU: Fungsi untuk membuat panel status ---
def create_status_panel(self, title_text, status_text):
    panel = QFrame()
    panel.setStyleSheet("""
        QFrame {
            background-color: white;
            border-radius: 8px;
        }
    """)
    panel.setFrameShape(QFrame.StyledPanel)
    panel_layout = QVBoxLayout(panel)
    panel_layout.setContentsMargins(15, 15, 15, 15)

    title_label = QLabel(title_text)
    title_label.setFont(QFont('Segoe UI', 10))
    title_label.setStyleSheet("color: #666;")

    self.status_value_label = QLabel(status_text)

```

```

        self.status_value_label.setFont(QFont('Segoe UI', 18,
QFont.Bold))
        self.update_status_ui() # Atur warna awal

    panel_layout.addWidget(title_label)
    panel_layout.addWidget(self.status_value_label)

    return panel

def connectInfluxDB(self):
    # Konfigurasi tidak berubah
    self.token =
"AMdJrRUMxq1PdZa4JA3CHg8QCEz3_JNxp_9dD0uFCnFrVtDOJnSBbSaHXrgeWMIW_
0QIbzl2aisMXPZXWaEmGA=="
    self.org = "Jamur"
    self.bucket = "Tiram2"
    self.url = "http://localhost:8086"
    try:
        self.client = InfluxDBClient(url=self.url,
token=self.token, org=self.org)
        self.query_api = self.client.query_api()
        self.status_text = "Terhubung"
        self.status_color = "#2ecc71" # Warna hijau
    except Exception as e:
        self.status_text = "Gagal"
        self.status_color = "#e74c3c" # Warna merah
        print(f"Gagal terhubung ke InfluxDB: {e}")
    self.update_status_ui()

def loadData(self):
    query = f'''
    from(bucket: "{self.bucket}")
        |> range(start: -1h)
        |> filter(fn: (r) => r._measurement ==
"environment_monitoring")
        |> filter(fn: (r) => r._field == "temperature_celsius" or
r._field == "humidity_percent")
        |> filter(fn: (r) => r["location"] == "Produksi Jamur")
        |> filter(fn: (r) => r["process_stage"] == "Jamur tiram")
        |> filter(fn: (r) => r["sensor_id"] == "SHT20-PascaPanen-
001")
    '''
    try:
        result_df = self.query_api.query_data_frame(query)

        if result_df.empty:
            self.status_text = "Menunggu Data"
            self.status_color = "#f39c12" # Warna oranye
            self.update_status_ui()
            return

        self.status_text = "Monitoring"
        self.status_color = "#3498db" # Warna biru

        # Proses data Suhu
    
```

```

        temp_df = result_df[result_df['_field'] ==
'temperature_celsius']
        if not temp_df.empty:
            temp_times =
pd.to_datetime(temp_df['_time']).astype(int) / 10**9
            temp_values = temp_df['_value']
            self.temp_plot.setData(x=temp_times.to_numpy(),
y=temp_values.to_numpy())
            self.last_temp = f"{temp_values.iloc[-1]:.1f} °C"
        else:
            self.temp_plot.setData([], [])

        # Proses data Kelembapan
        hum_df = result_df[result_df['_field'] ==
'humidity_percent']
        if not hum_df.empty:
            hum_times =
pd.to_datetime(hum_df['_time']).astype(int) / 10**9
            hum_values = hum_df['_value']
            self.humidity_plot.setData(x=hum_times.to_numpy(),
y=hum_values.to_numpy())
            self.last_humidity = f"{hum_values.iloc[-1]:.1f} %"
        else:
            self.humidity_plot.setData([], [])

        self.update_ui_values()

    except Exception as e:
        self.status_text = "Error Query"
        self.status_color = "#e74c3c" # Warna merah
        print(f"Gagal mengambil data: {e}")

    self.update_status_ui()

# --- BARU: Fungsi untuk update teks di panel atas ---
def update_ui_values(self):
    self.temp_value_label.setText(self.last_temp)
    self.humidity_value_label.setText(self.last_humidity)

# --- BARU: Fungsi untuk update teks dan warna status ---
def update_status_ui(self):
    if hasattr(self, 'status_value_label'):
        self.status_value_label.setText(self.status_text)
        self.status_value_label.setStyleSheet(f"color:
{self.status_color}; font-size: 18pt; font-weight: bold;")

if name == 'main': app = QApplication(sys.argv) window =
InfluxMonitor() window.show() sys.exit(app.exec_())

```

Lampiran 3. Pembuatan Web3 Sensor Dashboard

App.js

// app.js untuk Dashboard Monitoring Jamur (Polling Otomatis & Ekspor)

document.addEventListener('DOMContentLoaded', () => { // Variabel Global let web3,

```

contract, userAccount, sensorChart; let allReadings = [] // Variabel untuk menyimpan
semua data yang sudah diambil

// Elemen DOM
const connectButton = document.getElementById('connectButton');
const statusEl = document.getElementById('status');
const mainContentEl = document.getElementById('main-content');
const refreshAllButton =
document.getElementById('refreshAllButton');
const exportButton = document.getElementById('exportButton');
const latestDataEl =
document.getElementById('latestFromBlockchain');
const contractAddressInfoEl =
document.getElementById('contractAddressInfo');

function init() {
    if (typeof contractAddress === 'undefined' || typeof
contractABI === 'undefined') {
        alert("Error Kritis: File 'config.js' tidak
ditemukan.\n\nPastikan Anda sudah menjalankan:\nnpx hardhat run
scripts/deploy.js --network localhost");
        connectButton.disabled = true;
        connectButton.innerText = "Konfigurasi Error";
        return;
    }

    contractAddressInfoEl.innerText = Menggunakan Kontrak:
${contractAddress};
    initChart();
    connectButton.addEventListener('click', connectWallet);
    refreshAllButton.addEventListener('click', refreshAllData);
    exportButton.addEventListener('click', exportToExcel);
}

async function connectWallet() {
    if (typeof window.ethereum !== 'undefined') {
        try {
            statusEl.innerText = "Menunggu izin dari MetaMask...";
            const accounts = await ethereum.request({ method:
'eth_requestAccounts' });
            userAccount = accounts[0];
            statusEl.innerText = Terhubung sebagai:
${userAccount.substring(0,
6)}...${userAccount.substring(userAccount.length - 4)};
            mainContentEl.classList.remove('hidden');
            connectButton.parentElement.classList.add('hidden');
// Sembunyikan div connection-status

            web3 = new Web3(window.ethereum);
            contract = new web3.eth.Contract(contractABI,
contractAddress);

            await refreshAllData();
            setInterval(refreshAllData, 10000);
        }
    }
}

```

```

        } catch (error) {
            statusEl.innerText = ● Gagal terhubung:
            ${error.message};
        }
    } else {
        alert("Metamask tidak ditemukan.");
    }
}

async function refreshAllData() {
    if (!contract) return;
    console.log([${new Date().toLocaleTimeString()}] Memperbarui
data...);

    try {
        const total = await
contract.methods.getReadingsCount().call();
        if (total == 0) {
            latestDataEl.innerText = "Kontrak ini kosong (tidak
ada data).";
            allReadings = [];
            // Reset grafik jika kosong
            sensorChart.data.labels = [];
            sensorChart.data.datasets[0].data = [];
            sensorChart.data.datasets[1].data = [];
            sensorChart.update();
            return;
        }

        const readingsPromises = [];
        for (let i = 0; i < total; i++) {

readingsPromises.push(contract.methods.readings(i).call());
        }
        allReadings = await Promise.all(readingsPromises);

        const latestReading = allReadings[allReadings.length - 1];
        const temp = Number(latestReading.temperature) / 10.0;
        const humid = Number(latestReading.humidity) / 10.0;
        latestDataEl.innerText = Lokasi: ${latestReading.location}
| Suhu: ${temp.toFixed(1)}°C | Kelembaban: ${humid.toFixed(1)}%;

        sensorChart.data.labels = allReadings.map(r => new
Date(Number(r.timestamp) * 1000).toLocaleTimeString('id-ID'));
        sensorChart.data.datasets[0].data = allReadings.map(r =>
Number(r.temperature) / 10.0);
        sensorChart.data.datasets[1].data = allReadings.map(r =>
Number(r.humidity) / 10.0);
        sensorChart.update();

    } catch (err) {
        console.error("Gagal refresh data:", err);
        latestDataEl.innerText = "☒ Gagal memuat data dari
blockchain!";
    }
}

```

```

}

function exportToExcel() {
    if (allReadings.length === 0) {
        alert("Tidak ada data untuk diekspor!");
        return;
    }

    const headers = "ID,Waktu,Lokasi,Suhu (°C),Kelembaban (%)";

    const rows = allReadings.map((reading, index) => {
        const timestamp = new Date(Number(reading.timestamp) *
1000).toLocaleString('id-ID');
        const temp = (Number(reading.temperature) /
10.0).toFixed(1);
        const humid = (Number(reading.humidity) /
10.0).toFixed(1);
        return
`${index}`,"${timestamp}",`${reading.location}`,${temp},${humid};
    });

    const csvContent = [headers, ...rows].join("\n");

    const blob = new Blob([csvContent], { type:
'text/csv;charset=utf-8;' });
    const link = document.createElement("a");
    if (link.download !== undefined) {
        const url = URL.createObjectURL(blob);
        link.setAttribute("href", url);
        link.setAttribute("download", "data_sensor_jamur.csv");
        link.style.visibility = 'hidden';
        document.body.appendChild(link);
        link.click();
        document.body.removeChild(link);
    }
}

function initChart() {
    const ctx =
document.getElementById('sensorChart').getContext('2d');
    sensorChart = new Chart(ctx, {
        type: 'line',
        data: { labels: [], datasets: [ { label: 'Suhu (°C)', data: [], borderColor: '#e74c3c', fill: false }, { label: 'Kelembaban (%)', data: [], borderColor: '#3498db', fill: false } ] },
        options: { responsive: true, maintainAspectRatio: false }
    });
}

init();

});

```

Lampiran 4. Codingan Tambahan

Config.js

```
// FILE INI DIBUAT SECARA OTOMATIS OLEH deploy.js  
// JANGAN DIEDIT SECARA MANUAL
```

```
const contractAddress =  
"0x5FbDB2315678afecb367f032d93F642f64180aa3";  
const contractABI = [  
  {  
    "inputs": [],  
    "stateMutability": "nonpayable",  
    "type": "constructor"  
  },  
  {  
    "anonymous": false,  
    "inputs": [  
      {  
        "indexed": true,  
        "internalType": "uint256",  
        "name": "readingId",  
        "type": "uint256"  
      },  
      {  
        "indexed": false,  
        "internalType": "uint256",  
        "name": "timestamp",  
        "type": "uint256"  
      },  
      {  
        "indexed": false,  
        "internalType": "string",  
        "name": "location",  
        "type": "string"  
      }  
    ],  
    "name": "NewReading",  
    "type": "event"  
  },  
  {  
    "inputs": [  
      {  
        "internalType": "string",  
        "name": "_location",  
        "type": "string"  
      },  
      {  
        "internalType": "int256",  
        "name": "_temperature",  
        "type": "int256"  
      },  
      {  
        "internalType": "uint256",  
        "name": "_humidity",  
        "type": "uint256"  
      }  
    ]  
  }]
```

```

],
  "name": "addReading",
  "outputs": [],
  "stateMutability": "nonpayable",
  "type": "function"
},
{
  "inputs": [],
  "name": "getReadingsCount",
  "outputs": [
    {
      "internalType": "uint256",
      "name": "",
      "type": "uint256"
    }
  ],
  "stateMutability": "view",
  "type": "function"
},
{
  "inputs": [],
  "name": "owner",
  "outputs": [
    {
      "internalType": "address",
      "name": "",
      "type": "address"
    }
  ],
  "stateMutability": "view",
  "type": "function"
},
{
  "inputs": [
    {
      "internalType": "uint256",
      "name": "",
      "type": "uint256"
    }
  ],
  "name": "readings",
  "outputs": [
    {
      "internalType": "uint256",
      "name": "timestamp",
      "type": "uint256"
    },
    {
      "internalType": "string",
      "name": "location",
      "type": "string"
    },
    {
      "internalType": "int256",
      "name": "temperature",
      "type": "int256"
    }
  ]
}

```

```

        },
        {
            "internalType": "uint256",
            "name": "humidity",
            "type": "uint256"
        }
    ],
    "stateMutability": "view",
    "type": "function"
}
];

```

```

Index.html
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0"/>
    <title>Dashboard Monitoring Jamur</title>
    <link rel="stylesheet" href="style.css"/>
    <!-- Library untuk Grafik dan Web3 -->
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/web3@1.10.0/dist/web3.min.js"></
script>
</head>
<body>
    <div class="container">
        <header>
            <h1>🌿 Dashboard Monitoring Jamur Tiram</h1>
            <p>Mencatat dan menampilkan data sensor langsung dari
Blockchain.</p>
        </header>

        <!-- Elemen ini akan disembunyikan oleh JavaScript setelah
terhubung -->
        <div class="connection-status card">
            <button id="connectButton">Hubungkan Wallet
MetaMask</button>
            <p id="status">Status: Tidak Terhubung</p>
            <p id="contractAddressInfo" class="contract-info"></p>
        </div>

        <!-- Bagian utama ini akan muncul setelah wallet terhubung
-->
        <main id="main-content" class="hidden">
            <div class="card">
                <div class="card-header">
                    <h2>Data Terakhir dari Blockchain</h2>
                    <button id="exportButton" class="export-
btn">📝 Ekspor ke Excel</button>
                </div>
                <p id="latestFromBlockchain">Menunggu data...</p>
            </div>
        </main>
    </div>

```

```

        <div class="card chart-card">
            <h2>Grafik Real-time (Diperbarui Otomatis)</h2>
            <div class="chart-container">
                <canvas id="sensorChart"></canvas>
            </div>
            <button id="refreshAllButton">  Muat Ulang Grafik</button>
        </div>
    </main>
</div>

<!-- Skrip Aplikasi -->
<script src="config.js"></script>
<script src="app.js"></script>
</body>
</html>

```

Style.css

```

/* style.css (Desain Minimalis & Profesional) */ @import
url('https://fonts.googleapis.com/css2?family=Inter:wght@400;500;700&display=swap');

:root { --bg-color: #f8f9fa; /* Latar belakang abu-abu sangat terang / --card-bg: #ffffff; --
primary-color: #4CAF50; / Hijau Jamur / --secondary-color: #007bff; / Biru untuk
link/tombol sekunder */ --text-dark: #343a40; --text-light: #6c757d; --border-color:
#e9ecef; --shadow-color: rgba(0, 0, 0, 0.05); }

body { font-family: 'Inter', sans-serif; margin: 0; padding: 2.5rem; background-color: var(--bg-color); color: var(--text-dark); }

.container { max-width: 960px; margin: auto; }

header { text-align: center; margin-bottom: 2.5rem; }

header h1 { font-size: 2.2rem; font-weight: 700; margin-bottom: 0.5rem; display: flex;
justify-content: center; align-items: center; gap: 0.75rem; }

header p { color: var(--text-light); font-size: 1.1rem; }

.card { background: var(--card-bg); padding: 1.5rem 2rem; border-radius: 12px; box-
shadow: 0 4px 15px var(--shadow-color); margin-bottom: 2rem; border: 1px solid var(--border-color); }

.card-header { display: flex; justify-content: space-between; align-items: center; border-
bottom: 1px solid var(--border-color); padding-bottom: 1rem; margin-bottom: 1.5rem; }

.card h2 { margin: 0; font-size: 1.2rem; font-weight: 500; }

button { background-color: var(--secondary-color); color: white; padding: 10px 20px;
border: none; border-radius: 8px; cursor: pointer; font-size: 0.95rem; font-weight: 500;
transition: all 0.2s ease-in-out; }

```

```
button:hover { opacity: 0.85; transform: translateY(-2px); }

.export-btn { background-color: var(--primary-color); font-size: 0.9rem; padding: 8px 16px; }

#refreshAllButton { width: 100%; margin-top: 1rem; background-color: var(--text-light); }

#latestFromBlockchain { font-family: monospace; font-size: 1.2rem; color: var(--text-dark); text-align: center; background-color: #f8f9fa; padding: 1.5rem; border-radius: 8px; }

.chart-card { height: 500px; display: flex; flex-direction: column; }

.chart-container { flex-grow: 1; position: relative; }

.hidden { display: none !important; }

/* Penyesuaian khusus untuk elemen yang tidak digunakan lagi // .connection-status tidak lagi disembunyikan di sini, tapi dikontrol oleh app.js untuk pengalaman yang lebih baik */
.contract-info { font-family: monospace; font-size: 0.8rem; color: var(--text-light); margin-top: 1rem; }
```