RANCANG BANGUN ALAT PENGONTROLAN SUHU DAN KELEMBAPAN PADA PENGERINGAN LADA BERBASIS *INTERNET OF THINGS* (IOT)

PROYEK AKHIR

Laporan akhir ini dibuat dan diajukan untuk memenuhi salah satu syarat kelulusan Sarjana Terapan/Diploma IV Politeknik Manufaktur Negeri Bangka Belitung

Disusun Oleh:

Ejy Gustiansyah NIM: 1052207

Fira Sagita NIM: 1052209

POLITEKNIK MANUFAKTUR NEGERI BANGKA BELITUNG

TAHUN 2025

LEMBAR PENGESAHAN

RANCANG BANGUN ALAT PENGONTROLAN SUHU DAN KELEMBAPAN PADA PENGERINGAN LADA BERBASIS *INTERNET OF THINGS* (IOT)

Oleh:

Ejy Gustiansyah/1052207

Fira Sagita/1052209

Laporan akhir ini telah disetujui dan disahkan sebagai salah satu syarat kelulusan Program Diploma IV Politeknik Manufaktur Negeri Bangka Belitung

Menyetujui,

Pembimbing 1	Pembimbing 2
Yudhi, M.T.	Yuke Mareta Ariesta Sandra, S.P, M.Si.
Penguji 1	Penguji 2
Indra Dwisaputra, M.T.	Dewi Tumatul Ainin, S.Pd., M.Si.

PERNYATAAN BUKAN PLAGIAT

Yang bertanda tangan di bawah ini:

Nama Mahasiswa 1 : Ejy Gustiansyah NIM: 1052207

Nama Mahasiswa 2 : Fira Sagita NIM: 1052209

Dengan Judul : Rancang Bangun Alat Pengontrolan Suhu dan Kelembapan pada Pengeringan Lada Berbasis *Internet of Things* (IoT)

Menyatakan bahwa laporan akhir ini adalah hasil kerja kami sendiri dan bukan merupakan plagiat. Pernyataan ini kami buat dengan sebenarnya dan bila ternyata dikemudian hari ternyata melanggar pernyataan ini, kami bersedia menerima sanksi yang berlaku.

Sungailiat, 15 Juli 2025

Nama Mahasiswa	Tanda Tangan
1. Ejy Gustiansyah	.....
2. Fira Sagita	.....

ABSTRAK

Lada termasuk komoditas penting dalam sektor pertanian Indonesia. Provinsi Kepulauan Bangka Belitung menjadi salah satu wilayah penghasil terbesar. Proses pengeringan lada di daerah tersebut sebagian besar masih dilakukan secara tradisional, yang kurang efisien dan sangat bergantung pada kondisi cuaca. Penelitian ini bertujuan merancang alat pengering berbasis Internet of Things (IoT) untuk memantau serta mengontrol suhu, kelembapan, intensitas cahaya, dan curah hujan dari jarak jauh. Sistem dirancang menggunakan metode Research and Development (R&D) serta eksperimen. Alat terdiri dari rak pengering yang dilengkapi mikrokontroler ESP32, sensor DHT22, sensor cahaya BH1750, raindrop sensor, LCD, motor DC JGA25-370 sebagai penggerak atap, heater fan, dan lampu halogen sebagai pemanas. Aplikasi Android berbasis Flutter digunakan sebagai antarmuka untuk menampilkan data monitoring dan memberikan kontrol terhadap sistem. Hasil pengujian menunjukkan sistem mampu menampilkan pembacaan sensor secara real-time melalui LCD dan aplikasi, dengan persentase error suhu ±0,5% dan kelembapan ±5%. Sistem kontrol otomatis dan manual berjalan stabil sesuai perintah. Pengeringan menggunakan alat ini mempersingkat waktu menjadi dua hari, lebih cepat dibandingkan metode tradisional. Rancang bangun ini memberikan alternatif modern bagi petani dalam meningkatkan kualitas serta efisiensi proses pengeringan lada.

Kata kunci : DHT22, ESP32, Internet of Things(IoT), Lada, Pengering

ABSTRACT

Pepper is a vital commodity in Indonesia's agricultural sector, with the Bangka Belitung Islands Province being one of the major producing regions. In this area, the drying process is still largely carried out using traditional methods, which are inefficient and highly dependent on weather conditions. This study aims to design an Internet of Things (IoT)-based drying system to remotely monitor and control temperature, humidity, light intensity, and rainfall. The system is developed using the Research and Development (R&D) method combined with experimental approaches. The device consists of a drying rack equipped with an ESP32 microcontroller, DHT22 sensors, BH1750 light sensor, raindrop sensor, LCD, a JGA25-370 DC motor for roof control, a heater fan, and halogen lamps as heating elements. An Android application built with Flutter serves as the interface for displaying monitoring data and providing system control. Test results show that the system can display real-time sensor readings through both the LCD and the application, with a temperature error margin of ±0.5% and humidity ±5%. The automatic and manual control systems operate reliably according to commands. Drying using this tool shortens the process to two days, significantly faster than traditional methods. This development offers a modern alternative for farmers to improve the quality and efficiency of pepper drying.

Keywords: DHT22, Dryer, ESP32, Internet of Things (IoT), Pepper

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh.

Segala puji dan syukur penulis panjatkan kehadirat Allah SWT atas rahmat, taufik, dan hidayah-Nya, sehingga penulis dapat menyelesaikan Proyek Akhir ini dengan judul "RANCANG BANGUN ALAT PENGONTROLAN SUHU DAN KELEMBAPAN PADA PENGERINGAN LADA BERBASIS *INTERNET OF THINGS* (IOT)" sebagai syarat untuk menyelesaikan Program Studi D-IV Teknik Elektronika Jurusan Rekayasa Elektro dan Industri Pertanian Politeknik Manufaktur Negeri Bangka Belitung. Shalawat serta salam semoga tercurah kepada Nabi Muhammad SAW, keluarga, sahabat, dan seluruh umatnya yang senantiasa mengamalkan ajaran-ajarannya.

Penyusunan laporan ini bertujuan untuk merancang dan melakukan eksperimen penerapan sistem *smart roof* dalam alat pengeringan lada dengan sistem monitoring dan kontrol suhu serta kelembapan berbasis *Internet of Things* (IoT) guna meningkatkan efektivitas proses pascapanen lada yaitu proses pengeringan. Penulis menyadari bahwa proses pengerjaan Proyek Akhir ini tidak terlepas dari dukungan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Kedua orang tua tercinta dan seluruh keluarga yang telah memberikan dukungan dan do'a dengan ikhlas yang tak ternilai harganya.
2. Bapak I Made Andik Setiawan, M.Eng., Ph.D. selaku Direktur Politeknik Manufaktur Negeri Bangka Belitung.
3. Bapak Zanu Saputra, M.Tr.T. selaku Kepala Jurusan Rekayasa Elektro dan Industri Pertanian Politeknik Manufaktur Negeri Bangka Belitung.
4. Bapak Aan Febriansyah, M.T. selaku Kepala Program Studi D-IV Teknik Elektronika.
5. Bapak Yudhi, M.T. dan Ibu Yuke Mareta Ariesta Sandra S.P., M.Si. selaku dosen pembimbing yang telah membimbing, mengarahkan dan memberi saran dalam pembuatan alat dan penyusunan laporan Proyek Akhir ini.

6. Dosen dan Staf Pengajar di Politeknik Manufaktur Negeri Bangka Belitung yang telah mendidik, membina dan mengantarkan penulis untuk menempuh kematangan dalam berfikir dan berperilaku.
7. Sahabat yang selalu memberikan support, serta teman-teman yang lebih berkompeten yang telah meluangkan waktu untuk berdiskusi dan membantu dalam proses pembuatan alat dan penyusunan laporan Proyek Akhir ini.
8. Semua pihak yang tidak dapat disebutkan satu-persatu yang turut berkontribusi secara langsung maupun tidak langsung.

Penulis menyadari bahwa laporan Proyek Akhir ini masih banyak kekurangan, baik dalam segi bahasa maupun sistematika penulisan. Oleh karena itu, diharapkan kritik dan saran yang membangun dari pembaca agar dapat menunjang pengembangan dan perbaikan penulisan di kemudian hari. Semoga proyek akhir ini dapat berguna untuk menambah wawasan bagi penulis dan juga pembaca. Atas perhatiannya penulis mengucapkan terimakasih.

Wassalamua'laikum Warahmatullahi Wabarakatuh.

Sungailiat, 15 Juli 2025

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Tabel Halaman

DAFTAR GAMBAR

Gambar Halaman

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

Lada merupakan komoditas pertanian yang memiliki peranan penting dalam mendukung penerimaan devisa negara, meningkatkan pendapatan masyarakat tani, serta mendorong terciptanya lapangan kerja di sektor agrikultur. Berdasarkan data dari [1] selama sepuluh tahun terakhir, produksi lada di Indonesia terus meningkat, meskipun dengan pertumbuhan rata-rata yang relatif kecil yaitu 0,28% per tahun. Provinsi Kepulauan Bangka Belitung tercatat sebagai wilayah sentra utama dengan kontribusi sebesar 37,48% terhadap total produksi lada nasional. Berbagai tahap penanganan pascapanen yang perlu diperhatikan oleh petani dalam produksi lada. Tahap pengeringan merupakan salah satu bagian penting dalam penanganan pascapanen lada putih di Bangka, yang bertujuan menurunkan kadar air agar produk tidak mudah rusak dan memiliki daya simpan yang lebih lama [2].

Penulis melakukan penelitian dan membuat "RANCANG BANGUN ALAT PENGONTROLAN SUHU DAN KELEMBAPAN PADA PENGERINGAN LADA BERBASIS *INTERNET OF THINGS* (IOT)", dengan penerapan sistem *smart roof* yang dilengkapi sensor DHT22 sebagai pendeteksi suhu dan kelembapan serta *monitoring* secara *real-time* melalui aplikasi Android Flutter sehingga dapat memudahkan pemantaua kondisi lingkungan pengeringan dari jarak jauh. Hal yang melatarbelakangi pemilihan topik atau pengembangan alat ini yaitu adanya tantangan dalam proses pengeringan lada yang dominan secara tradisional bergantung pada kondisi cuaca dan waktu pengeringan yang cukup lama, sehingga dibutuhkan pengembangan alat berbasis teknologi dan *Internet of Things* guna mempercepat dan memudahkan pemantauan proses pengeringan.

Penelitian ini menggunakan pendekatan *Research and Development* (R&D) serta metode eksperimen, di mana peneliti merancang dan membangun alat *monitoring* suhu dan kelembapan dalam proses pengeringan lada. Sistem ini menggunakan ESP32 sebagai mikrokontroler utama yang menerima data dari

sensor DHT22 (untuk suhu dan kelembapan), sensor cahaya BH1750, serta sensor hujan analog (*raindrop sensor*). Informasi dari sensor kemudian ditampilkan melalui layar LCD dan aplikasi Android sebagai media pemantauan secara real-time. Sistem ini bekerja secara otomatis dan juga memungkinkan kontrol melalui aplikasi Android.

Penelitian sebelumnya [3] yaitu merancang alat ukur kadar air dan pengering biji-bijian berbasis mikrokontroler menggunakan komponen pemanas/*heater* sebagai pemanas udara dalam ruang pengering dan beberapa komponen lain untuk mendeteksi suhu dan kelembapan. Hasil pengukuran dan perhitungan presentase alat masih dalam toleransi yaitu 5% dan sistem telah dirancang otomatis. Hasil penelitian [4] menunjukkan bahwa sistem monitoring pada alat pengering berbasis Arduino mampu melakukan pengukuran dan menampilkan data secara akurat pada display, dengan tingkat kesalahan pengukuran suhu sebesar ±2,2%, kecepatan aliran udara ±1,25%, dan kelembapan sebesar 0%. Kedua penelitian tersebut berhasil merancang dan membangun sistem pengering biji-bijian atau lada berbasis mikrokontroler dan Arduino.

Penelitian sebelumnya menunjukkan bahwa teknologi berbasis mikrokontroler seperti Arduino memiliki potensi untuk dikembangkan menjadi alat pengering komoditas pertanian yang dapat dikembangkan dalam sistem *monitoring* dan kontrol terhadap variabel-variabel yang memengaruhi proses pengeringan, sehingga proses berlangsung lebih konsisten dan efisien. Berdasarkan hal tersebut, penelitian ini merancang sistem pengontrolan suhu dan kelembapan untuk pengeringan lada menggunakan mikrokontroler dan teknologi *Internet of Things* (IoT), serta memanfaatkan aplikasi Android sebagai sarana *monitoring* dan kontrol. Pengujian dilakukan untuk mengetahui dampak kinerja sistem terhadap hasil pengeringan.

1.2. Perumusan Masalah

Permasalahan yang menjadi dasar dalam perancangan dan pengembangan proyek akhir ini dirumuskan sebagai berikut:

1) Apakah sensor-senor yang digunakan terintegrasi dan memilki respons baik terhadap perubahan cuaca sekitar?
2) Apakah sistem terintegrasi dengan baik antara ESP32 dengan interface pengguna seperti LCD dan Aplikasi Android?
3) Bagaimana pengaruh alat dalam proses pengeringan lada dibandingkan dengan pengeringan tradisional?

1.3. Tujuan Proyek Akhir

Tujuan dari perancangan sistem dan pembuatan proyek akhir ini yaitu:

1) Mengetahui respons sensor yang digunakan terhadap perubahan cuaca sekitar tempat pengeringan.
2) Merancang dan membuat sistem terintegrasi antara ESP32 dengan interface pengguna (LCD dan Aplikasi Android).
3) Dapat membantu proses pengeringan lada secara lebih optimal dan efisien, mengetahui pengaruh alat terhadap waktu yang dibutuhkan dalam proses pengeringan, serta membandingkan hasilnya dengan pengeringan tradisional.

BAB II
DASAR TEORI

2.1. Lada (*Piper nigrum* L)

Lada (*Piper nigrum* L) atau disebut juga merica atau sahang, adalah rempah yang memiliki rasa pedas dan hangat serta berbagai khasiat. Lada memiliki kandungan senyawa kimia dalam konsentrasi tinggi, seperti pati, minyak lemak, dan minyak atsiri. Keanekaragaman varietas lada di Indonesia meliputi lada hijau, merah, hitam, serta putih [5].

Gambar 2. 1 Lada *(Piper Nigrum* L*)*

(Sumber: www.perkebunan.bsip.pertanian.go.id)

Menurut [1] data ATAP (Analisis Toksik Perkebunan) dari Direktorat Jenderal Perkebunan, Provinsi Kepulauan Bangka Belitung merupakan produsen lada terbesar di Indonesia. Daerah ini terkenal dengan produksi lada putih (*Muntok White Pepper*) yang tersebar di beberapa kabupaten/kota. Kabupaten Bangka Selatan menempati posisi teratas sebagai penghasil lada terbesar di Provinsi Bangka Belitung, menyumbang 46,92% dari total produksi yang setara dengan 15.258 ton. Sementara itu, kabupaten-kabupaten lain yang turut berkontribusi adalah Belitung (19,35%), Bangka Barat (10,63%), Bangka Tengah (10,32%), Bangka (6,69%), Belitung Timur (6,08%). Dilihat dari data tersebut, Kepulauan Bangka Belitung tidak hanya menjadi pusat produksi lada nasional, tetapi juga memiliki sebaran produksi yang merata di berbagai wilayahnya.

Lada melalui serangkaian proses pengolahan sebelum menjadi produk siap pakai yang dipasarkan sebagai rempah. Proses ini meliputi pemanenan, perendaman, pengeringan, dan pengemasan. Proses pemanenan terbagi menjadi dua untuk menghasilkan jenis lada berbeda yaitu pemanenan lada merah untuk menghasilkan lada putih atau pemanenan lada hijau untuk menghasilkan lada hitam. Proses setelah dipanen yaitu lada direndam dalam air untuk melunakkan kulitnya, sehingga biji lada dapat terpisah dari kulitnya. Tahap berikutnya adalah pengeringan yang bertujuan untuk mengurangi kadar air dalam lada dan menjaga kualitasnya sebagai rempah penyedap rasa. Proses pemanenan, perendaman, dan pengeringan harus dilakukan secara optimal guna memperoleh kualitas lada terbaik.

2.1.1. Proses Pengeringan Lada Tradisional

Tujuan utama dari proses pengeringan adalah menurunkan kandungan air dalam lada guna mempertahankan kualitasnya. Menurut penelitian [6], pada tahap pascapanen lada, petani kerap menghadapi permasalahan berupa penurunan kualitas dan efisiensi yang diakibatkan oleh ketidaksesuaian kadar air dan minyak, serta adanya cemaran mikroorganisme dan bahan asing. Tahapan perendaman, pengupasan kulit, dan pengeringan menjadi fase yang rentan terhadap kontaminasi.

Proses pengeringan lada pada tingkat petani umumnya dilakukan dengan cara dijemur dan dihamparkan langsung di atas permukaan tanah. Teknik ini berpotensi menyebabkan kontaminasi dari debu maupun kotoran hewan peliharaan. Ketergantungan pada kondisi cuaca juga menjadi kendala, karena cuaca yang tidak mendukung dapat memperlambat proses pengeringan dan berdampak pada penurunan mutu hasil panen [5]. Parameter seperti suhu dan kelembaban memengaruhi mutu lada sehingga perlunya penerapan teknologi dalam membantu proses pengontrolan atau pemantauan suhu dan kelembapan pada pengeringan lada.

Gambar 2. 2 Penjemuran Lada Putih di Tingkat Petani

(Sumber: Natawidjaya et al., 2012)

2.2. *Internet of Things* (IoT)

Internet of Things (IoT) merupakan sebuah revolusi teknologi yang merepresentasikan masa depan komputer dan komunikasi, dimana dinamika setiap elemen terintegrasi dengan sistem kecerdasan untuk memanfaatkan kekuatan jaringan dengan pemrosesan informasi yang di perlukan. *Internet of Things* (IoT) merepresentasikan pengembangan teknologi berbasis jaringan yang menghubungkan berbagai benda secara otomatis, dengan tujuan mempermudah kegiatan harian serta meningkatkan efektivitas kerja manusia [7].

Internet of Things (IoT) beroperasi dengan mengimplementasikan logika pemrograman tertentu, di mana setiap perintah logis akan menciptakan interaksi otomatis antar perangkat. Konektivitas antara komponen difasilitasi oleh jaringan internet, memungkinkan komunikasi terjadi tanpa batasan jarak. Peran manusia terbatas pada fungsi pengawasan dan konfigurasi awal perangkat saja, sementara proses operasional berjalan secara mandiri tanpa memerlukan intervensi manusia secara langsung [8].

2.3. Sistem *Monitoring* dan Kontrol

Monitoring merupakan proses pengamatan secara sistematis terhadap suatu aktivitas guna memperoleh data yang sesuai dan menghindari penyimpangan yang melebihi standar yang telah ditentukan. Data yang terkumpul dapat dijadikan dasar dalam menentukan keputusan untuk aktivitas selanjutnya, upaya ini dilakukan untuk menjaga efektivitas dan efisiensi dalam suatu proses aktivitas [4].

Menurut penelitian [9], *monitoring* bertujuan mengumpulkan informasi yang relevan yang terkait dengan parameter atau kondisi tertentu. Tujuan ini dapat bervariasi, seperti memastikan keamanan dalam lingkungan industri, menilai kualitas air di lingkungan, atau memantau tingkat kelembapan dalam proses produksi, dan lain-lain.

Berdasarkan artikel [10] menyatakan bahwa kemajuan teknologi elektronik di awal abad ke-20 membuka jalan bagi sistem kendali atau kontrol yang lebih kompleks. Pada tahun 1927, Harold S. Black menemukan *negative feedback amplifier*, yang menjadi dasar sistem kendali tertutup modern. Penemuan ini merevolusi industri komunikasi dan elektronik. Pada periode yang sama, ilmu kontrol formal mulai terbentuk dengan publikasi teori kendali klasik oleh Norbert Wiener. Konsep *Cybernetics* yang diperkenalkan Wiener pada tahun 1948 menjadi titik awal penggabungan antara kendali otomatis, biologi, dan komunikasi. Perkembangan terkini menunjukkan bahwa sistem kontrol dan monitoring telah terintegrasi dengan teknologi *Internet of Things* (IoT) dan *artificial intelligence* (AI). Konektivitas antarperangkat memungkinkan pertukaran data secara otomatis serta operasional yang mandiri tanpa campur tangan pengguna.

2.4. Mikrokontroler ESP32

ESP32 merupakan mikrokontroler canggih yang mengintegrasikan berbagai komponen dalam satu chip tunggal. Perangkat ini menyertakan fitur konektivitas nirkabel berupa Wi-Fi standar 802.11 b/g/n dan Bluetooth 4.2, dilengkapi dengan beragam antarmuka peripheral. Chip ini memiliki arsitektur lengkap yang mencakup unit pemrosesan, memori, serta port input/output serbaguna (GPIO) [11].

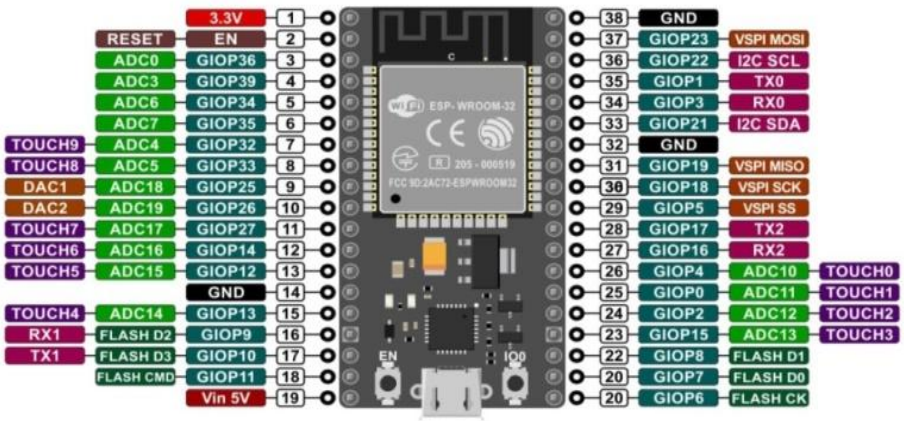
ESP32 adalah mikrokontroler keluaran Espressif Systems yang merupakan pengembangan dari seri sebelumnya, yaitu ESP8266. Mikrokontroler ini memiliki prosesor *dual-core* berbasis arsitektur Xtensa LX16 dengan kecepatan hingga 240 MHz, dilengkapi RAM sebesar 520 KB dan memori *flash* hingga 4 MB, serta beroperasi pada tegangan 3,3V dengan konsumsi daya yang rendah. Karakteristik

ini menjadikannya ideal untuk aplikasi *Internet of Things* (IoT) [12]. Pada proyek ini, ESP32 berperan sebagai unit pengolah utama yang bertugas mengumpulkan data dari berbagai sensor, memproses informasi tersebut, dan mengirimkannya ke server melalui koneksi internet.

Fitur ADC pada ESP32 digunakan untuk membaca sinyal analog dari sensor hujan dan mengubahnya menjadi data digital 12-bit. Data ini kemudian dikalibrasi dan ditampilkan secara real-time melalui aplikasi IoT berbasis Flutter, memungkinkan pemantauan suhu dan kelembapan berlangsung akurat dan efisien. Berikut adalah gambar ESP32 ditunjukan pada Gambar 2.3 dan spesifikasi dari ESP32 ditunjukan pada Tabel 2.1.

Gambar 2. 3 Mikrokontroler ESP32 38PIN

(Sumber:Maharani et al., 2023)



Gambar 2. 4 PIN ESP32

(Sumber: www.embeddednesia.com)

Tabel 2. 1 Spesifikasi Mikrokontroler ESP32 38PIN

No.	Pin	Keterangan
1.	*Operating voltage*	3.3V
2.	*Input voltage (Vin)*	5-12V (rekomendasi 7-9V)
3.	*Digital IO Pin (DIO)*	25
4.	*Analog Input Pin (ADC)*	15
5.	*Analog Output Pin (DAC)*	2
6.	UART	2
7.	SPI	1
8.	I2C	1
9.	*Program memory*	448 kB
10.	*Maximum clock*	120 MHz
11.	*Wi-Fi*	802.11 b/g/n/e/i (802.11n @ 2.4 GHz up to 150 Mbit/s)
12.	*Bluetooth*	v4.2 BR/EDR *and Bluetooth Low Energy* (BLE)
13.	*SDIO master/slave*	50 MHz
14.	Mode supported	AP, STA, dan AP+STA

2.5. DHT22 (Sensor Suhu dan Kelembapan)

DHT22 merupakan sensor digital yang berfungsi mengukur suhu dan kelembapan dengan tingkat akurasi tinggi. Sensor ini memanfaatkan teknologi kapasitif untuk kelembapan dan NTC thermistor untuk suhu, serta terintegrasi langsung dengan mikrokontroler. Keunggulannya mencakup respons yang cepat, kestabilan jangka panjang, dan ketahanan terhadap gangguan [11].

DHT22 memiliki presisi lebih baik dengan harga yang lebih terjangkau, sehingga sensor ini menjadi pilihan untuk berbagai pengaplikasian. Apabila dipasangkan dengan *board microcontroller*, sensor ini memungkinkan pengguna dengan mudah memantau dan menganalisis hubungan antara perubahan suhu dan kelembaban secara *real-time* [13].

Berikut adalah gambar DHT22 ditunjukan pada Gambar 2.5 dan spesifikasi dari DHT22 ditunjukan pada Tabel 2.2.

Gambar 2. 5 DHT22 *(Temperature and Humidity)*

(Sumber: Saputra et al., 2020)

Tabel 2. 2 Spesifikasi DHT22

Voltage Input	3.3 – 5V
Maximum Current	2.5mA
Temperature range	-40~80°C (±0.5°C error)
Humidity range	0-100% RH (±5% RH error)

2.6. Sensor BH1750 (Intensitas Cahaya)

BH1750 merupakan sensor digital yang digunakan untuk mengukur intensitas cahaya di sekitarnya. Sensor ini bekerja melalui antarmuka I2C yang memudahkan koneksi dengan berbagai perangkat mikrokontroler. Keunggulan utamanya terletak pada kemampuan membaca cahaya dengan sangat akurat (resolusi 16-bit), rentang pengukuran luas dari 1 lux (sangat redup) hingga 65,535 lux (sangat terang), serta pengukuran langsung dalam satuan lux (ukuran kecerahan cahaya) [14].

Gambar 2. 6 *Module Sensor BH1750*

(Sumber: www.nn-digital.com)

Menurut penelitian [15], modul BH1750 merupakan sensor cahaya digital yang bekerja berdasarkan prinsip fotometri, yaitu dengan mengubah intensitas cahaya menjadi data digital yang dapat langsung dibaca oleh mikrokontroler. Rentang pengukuran intensitas cahaya pada modul ini bergantung pada tipe atau model yang digunakan, sehingga dapat disesuaikan dengan kebutuhan sistem. Dibandingkan dengan sensor analog seperti LDR dan photodiode, BH1750 memiliki keunggulan dalam hal akurasi serta kemudahan integrasi, sehingga lebih praktis dan efisien untuk digunakan dalam sistem berbasis mikrokontroler. Berikut spesifikasi dari sensor BH1750 ditunjukan pada Tabel 2.3.

Tabel 2. 3 Spesifikasi *Module Sensor BH1750*

Catu daya	3-5V
Resolusi	1-65535 lux
Antarmuka	I2C
Ukuran papan	13,9 mm × 18,5 mm
Keluaran data	digital
Mode pengukuran	*Single atau Continuous*

2.7. *Raindrop Sensor*

Raindrop sensor merupakan alat pendeteksi curah hujan yang umum digunakan dalam berbagai aplikasi lingkungan dan otomatisasi. Cara kerja sensor ini yaitu ketika air hujan mengenai permukaan sensor, terjadi reaksi elektrolisis karena sifat elektrolit pada air hujan yang memungkinkan terjadinya aliran arus listrik. Modul ini dilengkapi dengan IC komparator yang menghasilkan sinyal digital berupa logika *high* atau *low* (on/off), serta menyediakan *output* tegangan sebagai sinyal analog. Kemampuan memberikan dua jenis keluaran ini memungkinkan sensor digunakan baik untuk pemantauan sederhana maupun sistem otomatis yang membutuhkan respon terhadap intensitas hujan secara *real-time* [16].

Gambar 2. 7 *Raindrop Sensor*

(sumber: Lestari & Abdulrahman, 2021)

Tabel 2. 4 Spesifikasi *Raindrop Sensor*

Operating Voltage	3.3 – 5V
Output	Digital dan analog
Board PCB size	3.2cm x 1.4cm
Interface	VCC positif 3-5V, GND *ground*, DO *Digital output* (0 dan 1), AO *Analog output*

2.8. Motor DC JGA25-370

Motor DC JGA25-370 12V Gearbox dengan encoder 60rpm adalah motor DC kecil dengan gearbox yang memungkinkan torsi lebih tinggi dan kecepatan yang lebih rendah (60 putaran per menit). Penulis menggunakan motor DC ini sebagai penggerak sistem otomasi atap, dimana atap ini juga dapat dikontrol melalui aplikasi Android sehingga memerlukan torsi yang cukup tinggi.

Motor gearbox merupakan motor yang dilengkapi sistem roda gigi untuk mengubah kecepatan tinggi dan torsi rendah menjadi kecepatan rendah dengan torsi tinggi. Keunggulannya terletak pada kemampuannya menghasilkan gaya dorong lebih besar, sehingga lebih sulit dihentikan dibanding motor biasa. Umumnya digunakan pada aplikasi yang membutuhkan torsi besar, seperti elevator. Meskipun terjadi sedikit kehilangan energi saat reduksi kecepatan, motor ini tetap efektif dalam membawa beban berat [17].

Gambar 2. 8 Motor DC JGA25-370

(sumber: www.narincomicro.com)

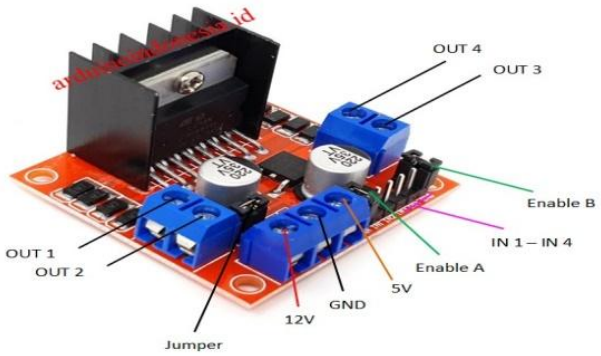
Tabel 2. 5 Spesifikasi Motor DC JGA25-370

Operating Voltage	12V
Gear	Metal
Shaft	Bentuk D
Dimensi	P 73mm x L 24.4mm x T 24.4mm

2.9. *Driver Motor L298N*

Driver L298N merupakan modul pengendali motor DC yang berfungsi untuk mengatur kecepatan dan arah putaran motor. Modul ini umum digunakan bersama mikrokontroler seperti Arduino. Komponen utamanya adalah IC L298N bertipe H-bridge, yang dirancang untuk menangani beban induktif tinggi pada perangkat seperti motor DC, motor stepper, relay, dan solenoid. IC ini juga dilengkapi rangkaian logika TTL dan gerbang NAND untuk mengontrol arah putaran motor secara efisien [18].

Keunggulan dari *module driver motor L298N* ini adalah dari segi presisi dalam pengontrolan motor, sehingga motor lebih mudah dikendalikan. Penulis menggunakan *driver motor L298N* untuk mengontrol pergerakan motor DC karena memiliki kelebihan dalam hal kepresisian, sehingga nantinya motor DC lebih mudah untuk dikontrol sesuai kondisi yang sudah diatur pada program Arduino IDE.



Gambar 2. 9 *Driver Motor L298N*

(Sumber: www.arduinoindonesia.id)

2.10. *Heater Fan* (Kipas Pemanas)

Heater fan adalah jenis pemanas yang menggunakan kipas angin untuk mendistribusikan panas ke seluruh ruang tertutup. Heater fan pada alat pengering lada berperan untuk menghasilkan udara panas yang digunakan dalam proses pengeringan. Udara panas ini membantu menurunkan kadar air pada lada secara merata dan efektif, sehingga kualitas lada tetap terjaga. Heater fan bekerja dengan cara memanaskan udara dan menyebarkannya ke seluruh bagian lada yang sedang dikeringkan, mempercepat penguapan air dan mencegah tumbuhnya jamur atau bakteri.

Penggunaan pemanas jenis *heater* listrik atau *heater fan* memiliki keuntungan yaitu membantu mempercepat waktu pengeringan, serta mengurangi emisi karbon dibandingkan dengan penggunaan bahan bakar fosil. Selain itu, teknologi ini dapat digunakan dalam berbagai kondisi cuaca, memberikan fleksibilitas yang lebih besar bagi pengguna [19].

Gambar 2. 10 *Heater fan*

2.11. LCD *(Liquid Crystal Display)*

LCD *(Liquid Crystal Display)* adalah teknologi layar yang memanfaatkan kristal cair untuk mengontrol cahaya dan menampilkan suatu data, baik karakter, huruf ataupun grafik [3]. Strukturnya terdiri dari dua lapisan transparan yang diisi dengan kristal cair, diapit oleh dua filter polarisasi yang saling tegak lurus. LCD tidak menghasilkan cahaya sendiri, sehingga memerlukan pencahayaan latar belakang, biasanya menggunakan lampu LED atau neon putih, untuk menerangi layar. Setiap piksel dikendalikan oleh transistor yang mengatur orientasi kristal cair melalui medan listrik, memungkinkan atau menghalangi cahaya untuk melewati filter polarisasi dan menciptakan tampilan gambar.

Untuk menghasilkan warna, setiap piksel dibagi menjadi tiga sub-piksel dengan filter warna merah, hijau, dan biru (RGB). Dengan mengatur intensitas cahaya yang melewati masing-masing sub-piksel, berbagai kombinasi warna dapat ditampilkan. Penulis menggunakan jenis LCD 16x2 seperti pada Gambar 2.11. LCD 16x2 dapat menampilkan sebanyak 32 karakter yang terdiri dari 2 baris dan tiap baris dapat menampilkan 16 karakter [20].

Gambar 2. 11 LCD *(Liquid Crystal Display)*

(Sumber: www.altanelektronik.net)

2.12. Lampu Halogen

Penelitian sebelumnya yang dilakukan pada sistem kabinet pengering jagung [21] memanfaatkan lampu halogen sebagai sumber panas utama. Pengujian dilakukan dengan memvariasikan posisi rak pengering (rak 1 hingga rak 3) serta mengatur kecepatan blower pada tiga tingkat berbeda, yaitu 15.000

rpm, 10.000 rpm, dan 5.000 rpm, dalam durasi waktu pengeringan yang sama selama 30 menit.

Hasil yang diperoleh menunjukkan bahwa penggunaan lampu halogen memberikan proses pengeringan yang lebih merata pada setiap rak dibandingkan dengan penggunaan lampu bohlam konvensional. Hal ini disebabkan oleh daya pancar lampu halogen yang lebih luas serta kemampuannya dalam menghasilkan suhu yang lebih stabil. Selain itu, sirkulasi udara di dalam kabinet menjadi lebih optimal karena distribusi panas yang merata dari lampu halogen. Keunggulan ini menjadikan lampu halogen lebih efektif sebagai elemen pemanas dalam sistem pengering, terutama pada aplikasi yang membutuhkan kestabilan suhu dan distribusi panas yang seragam, seperti pada pengeringan biji-bijian.

Gambar 2. 12 Lampu Halogen

2.13. Arduino IDE (*Integrated Developmen Environment*)

Arduino IDE adalah software yang dirancang khusus untuk memprogram berbagai board mikrokontroler. Aplikasi ini menyediakan lingkungan kerja lengkap mulai dari menulis kode, mengkompilasi, hingga mengupload program ke board. Bahasa pemrograman yang digunakan berbasis C/C++ dengan berbagai fungsi bawaan yang sudah disederhanakan. Keunggulan utama software ini adalah sifatnya yang open source, memungkinkan siapa saja bisa memanfaatkannya untuk mengembangkan proyek elektronik tanpa kendala [11].

Antarmuka *software* ini sederhana namun lengkap, terdiri dari beberapa bagian utama yang memudahkan proses pembuatan dan pengunggahan kode program. Editor teks menjadi area utama tempat pengguna menulis *sketch* (*sketch*: istilah yang digunakan Arduino untuk menyebut kode program). Di bagian atas antarmuka, terdapat *toolbar* dengan berbagai tombol penting. Tombol-tombol ini

berfungsi untuk memverifikasi kode, mengunggah program ke *board* Arduino, membuat *sketch* baru, membuka file *existing*, serta menyimpan pekerjaan. Setiap *sketch* yang dibuat akan disimpan dengan ekstensi file.ino, format khusus untuk program Arduino [16].

Gambar 2. 13 Logo *Software* Arduino IDE
(Sumber: www.commons.wikimedia.org)

2.14. Flutter

Flutter merupakan *framework* pengembangan *aplikasi mobile* berbasis *open source* yang dikembangkan oleh Google. *Platform* ini mendukung pembuatan aplikasi lintas sistem operasi seperti Android, iOS, hingga Fuchsia [22]. Bahasa pemrograman yang digunakan adalah Dart, yang juga dikembangkan oleh Google dan dikenal sebagai bahasa pemrograman serbaguna (*general-purpose*). Sintaks Dart mirip dengan Java dan JavaScript, sehingga mudah dipahami oleh pengembang yang telah familiar dengan kedua bahasa tersebut. Dart mendukung pengembangan berbagai jenis aplikasi, mulai dari aplikasi Android, web, IoT, backend berbasis CLI, hingga game. Bahasa ini bersifat dinamis dan mendukung kompilasi awal, sehingga memberikan performa tinggi [23].

Gambar 2. 14 Flutter
(Sumber: flutter-logo.svg)

Menurut penelitian [22], Flutter dibangun di atas engine yang ditulis dalam C++ dan memanfaatkan library grafis Skia untuk menghasilkan tampilan

antarmuka dengan kecepatan hingga 120 frame per detik. Struktur antarmuka pengguna dalam Flutter dibentuk dari widget, yang mencakup berbagai elemen visual seperti teks, bentuk, dan animasi. Salah satu fitur andalan Flutter adalah *hot reload*, yang memungkinkan pengembang melihat perubahan kode secara instan tanpa perlu menjalankan ulang aplikasi, sehingga mempercepat proses pengembangan dan uji coba [24].

2.15. Firebase

Firebase adalah layanan berbasis *cloud* milik Google yang menyediakan berbagai fitur untuk mendukung komunikasi antar perangkat secara daring. Layanan ini mencakup kemampuan seperti analisis aplikasi, penyimpanan data, pengiriman notifikasi, serta pelaporan kesalahan atau bug dalam aplikasi. *Firebase* menjadi solusi yang sangat membantu para pengembang dalam membangun dan mengelola aplikasi yang terhubung dengan banyak perangkat secara efisien [20].

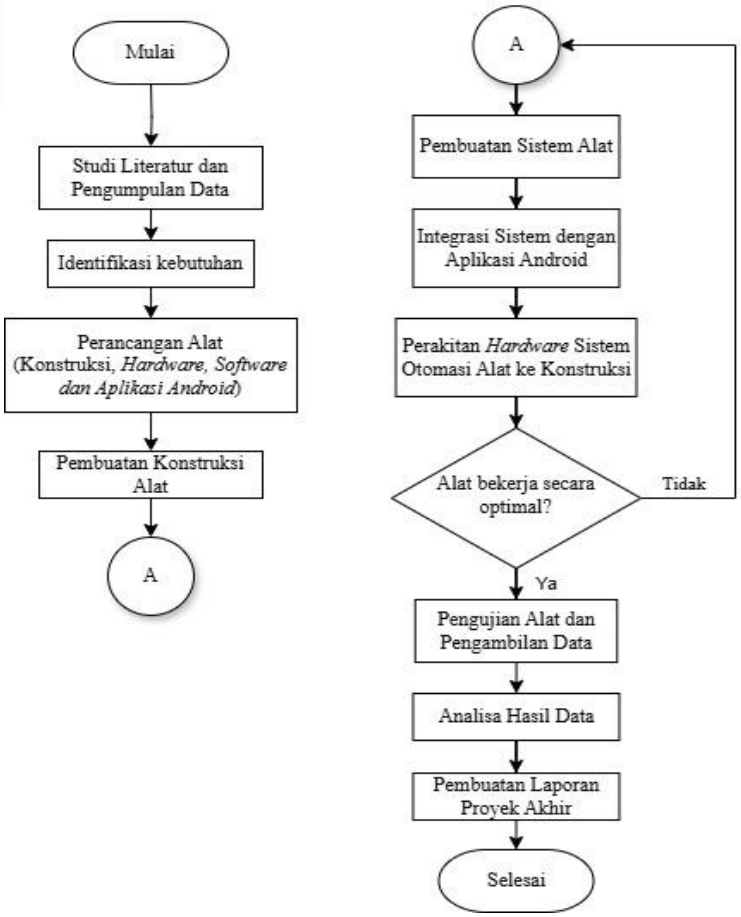
Gambar 2. 15 Firebase

(sumber: www.*firebase.google.com)*

Dalam penelitian ini, fitur *Firebase* yang dimanfaatkan adalah *Realtime Database*. Menurut penelitian [23], *Firebase Realtime Database* adalah database NoSQL yang memungkinkan akses data secara langsung dari sisi client. Melalui fitur ini, data disimpan dalam format JSON dan secara otomatis disinkronkan secara *real-time* ke seluruh pengguna yang terhubung. Data yang tersimpan pada *Realtime Database* kemudian ditampilkan dalam aplikasi Android, sehingga memudahkan proses pemantauan dan pengendalian sistem dari jarak jauh.

BAB III
METODE PELAKSANAAN

Proyek Akhir ini mengimplementasikan tahapan-tahapan sistematis dalam perancangan alat pengontrolan suhu dan kelembapan pada pengeringan lada berbasis *Internet of Things* (IoT). Penelitian menggunakan metode *Research and Development* (R&D) serta eksperimen. Peneliti merancang alat dengan sistem *monitoring* dan kontrol otomatis yang terintegrasi aplikasi Android. Pengujian dilakukan terhadap performa alat menggunakan sampel lada, kemudian hasilnya dibandingkan dengan metode pengeringan tradisional (konvensional). Diagram alir pada Gambar 3.1 menggambarkan tahapan pelaksanaan proyek secara keseluruhan.



Gambar 3. 1 Diagram Alir Metode Pelaksanaan

3.1. Identifikasi Kebutuhan

Berdasarkan studi literatur yang dilakukan untuk memahami parameter terkait pengeringan lada, seperti suhu dan kelembapan optimal, kondisi lingkungan pengeringan seperti intensitas cahaya dan hujan, serta analisis data penelitian sebelumnya guna mengamati masalah dalam pengeringan tradisional. Berdasarkan hasil identifikasi, ditentukan komponen utama alat, seperti mikrokontroler ESP32, sensor DHT22, *raindrop sensor*, sensor BH1750, lampu halogen dan motor DC JGA25-370, *heater fan* dan kipas DC, serta kebutuhan IoT seperti protokol komunikasi dan *platform cloud* Firebase.

3.2. Perancangan Sistem Alat

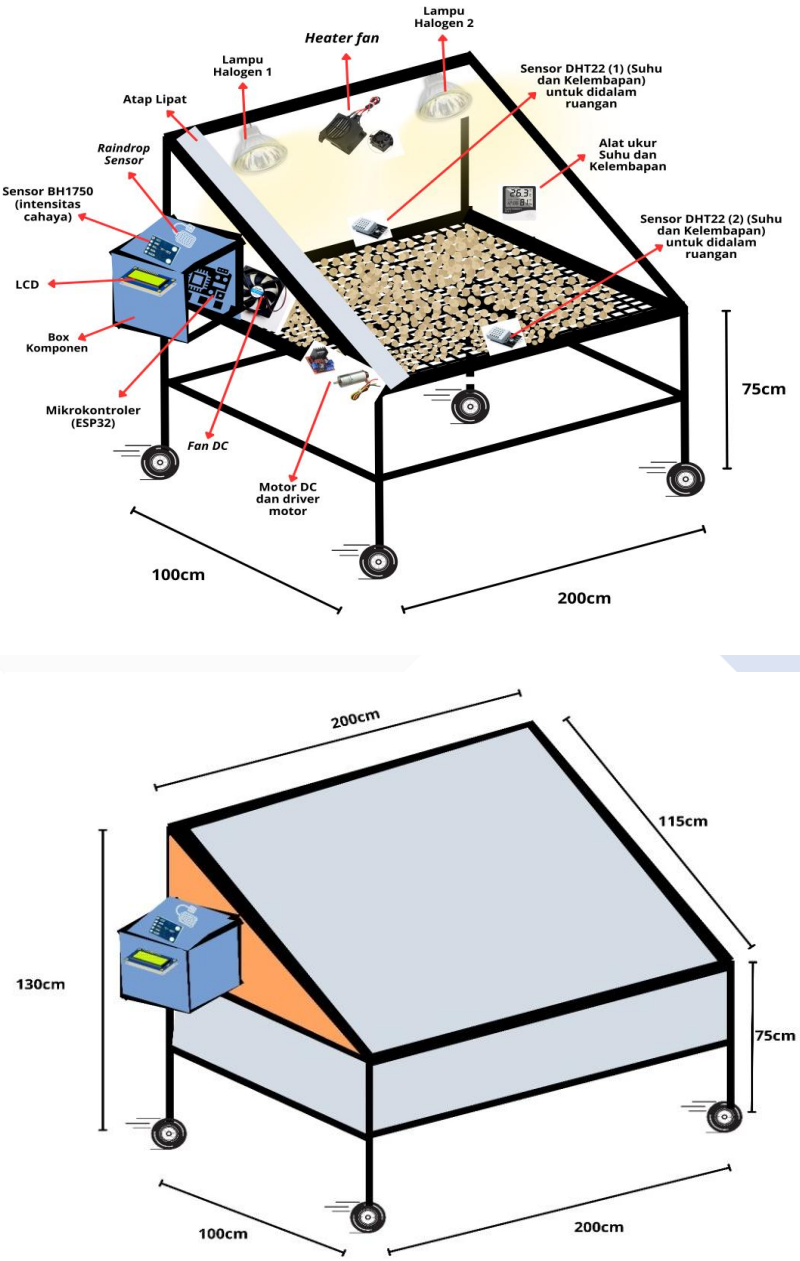
Tahap perancangan sistem meliputi beberapa aspek penting yang saling berkaitan, yaitu perancangan konstruksi fisik alat, perangkat keras (*hardware*), perangkat lunak (*software*), serta aplikasi Android sebagai antarmuka pengguna. Pada bagian konstruksi, dilakukan perancangan struktur dan tata letak komponen agar sistem dapat berfungsi secara optimal dan aman digunakan. Perancangan *hardware* mencakup pemilihan dan integrasi komponen seperti mikrokontroler, sensor, aktuator, serta modul komunikasi.

Perancangan *software* berfokus pada pembuatan program yang tertanam pada mikrokontroler untuk mengelola *input* dan *output* dari sistem. Aplikasi Android dirancang sebagai media *monitoring* dan kontrol jarak jauh, yang terhubung dengan sistem secara *real-time* melalui Firebase, sehingga pengguna dapat menerima data secara langsung.

3.2.1. Perancangan Konstruksi Alat

Perancangan konstruksi sangat memperhatikan bentuk alat pengering, kapasitas muatan pengeringan, bahan yang digunakan untuk membuat konstruksi, serta posisi atau tata letak komponen-komponen yang digunakan pada konstruksi. Konstruksi alat pengering lada ini dibuat dari rangka baja ringan berukuran 2 x 1 meter dengan atap berbahan plastik UV tahan panas dan air, serta alas

pengeringan menggunakan jaring bahan alumunium. Ilustrasi rancangan konstruksi dapat dilihat pada Gambar 3.2.



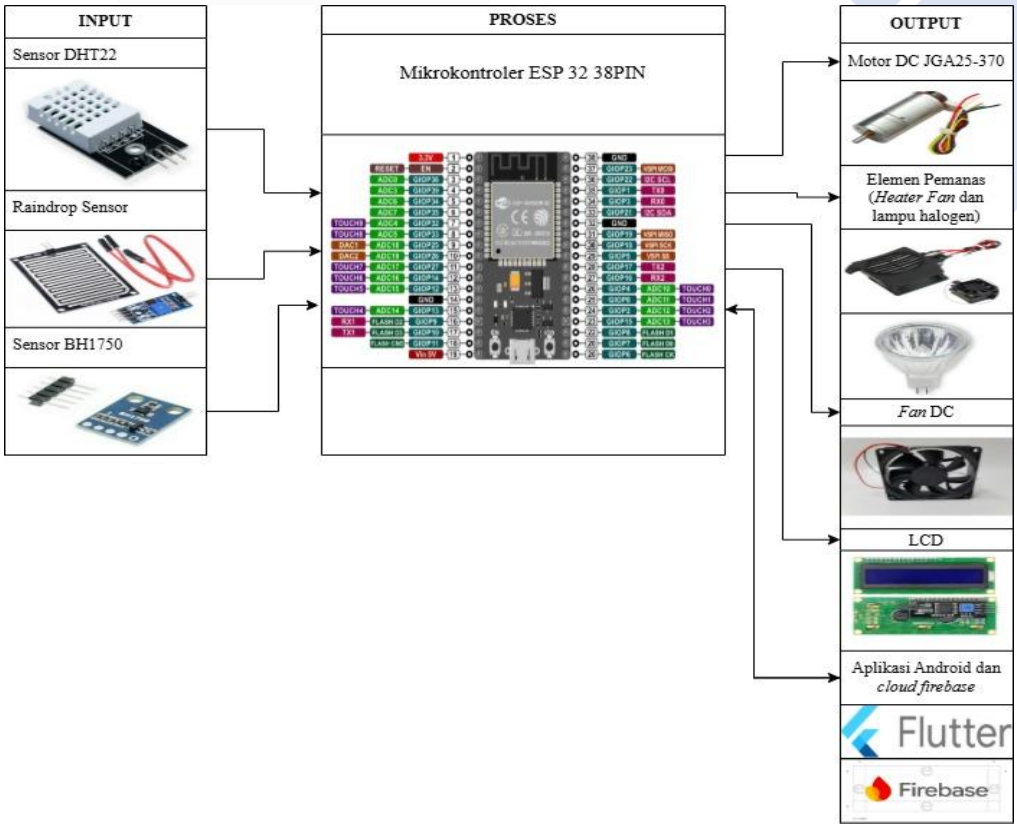
Gambar 3. 2 Ilustrasi Rancangan Konstruksi

3.2.2 Rancangan Perangkat Keras (*Hardware*)

Penulis merancang hubungan antar komponen *input*, proses dan *outputnya*, pada bagian *hardware*. *Hardware* ini biasanya direpresentasikan dalam diagram

blok yang dapat dilihat pada Gambar 3.3. Diagram blok sistem menggambarkan alur kerja alat pengontrol suhu dan kelembapan berbasis mikrokontroler ESP32. Sistem terdiri atas tiga bagian utama, yaitu *input*, proses, dan *output*. Bagian *input* mencakup tiga sensor: DHT22 untuk mengukur suhu dan kelembapan, *raindrop sensor* untuk mendeteksi hujan, serta BH1750 untuk mengukur intensitas cahaya. Data dari sensor ini dikirim ke mikrokontroler ESP32, yang berfungsi sebagai unit pemroses utama.

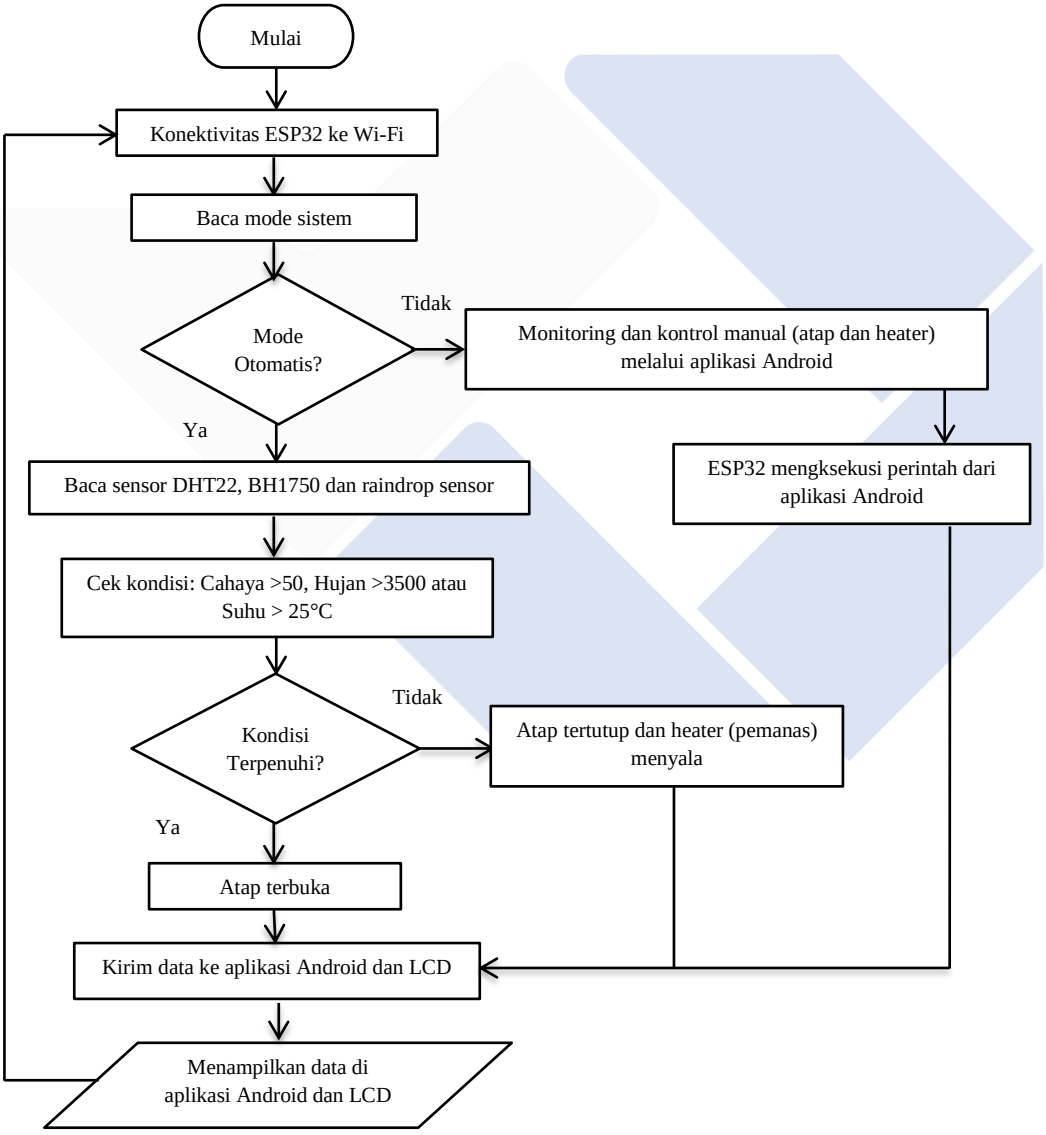
Mikrokontroler memproses data secara *real-time* dan mengendalikan beberapa komponen output, yaitu motor DC JGA25-370 untuk membuka atau menutup atap, *heater fan* dan lampu halogen sebagai pemanas, serta kipas DC untuk menjaga sirkulasi udara. Sistem juga terhubung dengan aplikasi Android berbasis Flutter, yang menampilkan data sensor dan memungkinkan pengguna melakukan kontrol jarak jauh melalui *Firebase Realtime Database*. Integrasi ini memungkinkan sistem bekerja secara otomatis, responsif, dan dapat dipantau dari mana saja.



Gambar 3. 3 Diagram Blok Sistem

3.2.3. Rancangan Perangkat Lunak (*software*)

Perancangan perangkat lunak pada sistem ini dilakukan dengan memprogram mikrokontroler menggunakan Arduino IDE, yang berfungsi untuk membaca data dari sensor serta mengendalikan seluruh komponen sistem. Sementara itu, aplikasi Android dikembangkan menggunakan Flutter, dengan fitur utama berupa pemantauan data secara *real-time* dan menampilkan histori data ke pengguna. Sistem ini juga terintegrasi dengan layanan *cloud* Firebase, yang digunakan untuk menyimpan dan mengirimkan data secara online.



Gambar 3. 4 *Flowchart System*

Flowchart system diatas menggambarkan alur kerja sistem yang dimulai dari pembacaan mode otomatis atau manual. Apabila berada di mode otomatis maka dilakukan pembacaan sensor hujan, cahaya, suhu, dan kelembapan. Jika kondisi lingkungan normal, atap terbuka dan data ditampilkan pada aplikasi Android serta LCD. Jika terdeteksi hujan, mendung, atau suhu rendah, atap ditutup dan pemanas aktif, kemudian data dikirim ke aplikasi dan LCD. Apabila berada di mode manual, maka sistem otomatis akan terputus dan pengguna dapat melakukan aksi buka/tutup atap atau on/off *heater* melalui *button* pada aplikasi Android.

3.3. Pembuatan Konstruksi Alat

Tahap pembuatan konstruksi alat ini dilakukan perakitan konstruksi berdasarkan rancangan yang telah dibuat seperti pada Gambar 3.2. Beberapa komponen penting dalam perakitan konstruksi yaitu kerangka pengeringan, wadah pengeringan, dan atap alat pengeringan. Alat ini juga didesain menggunakan roda agar dapat dipindahkan dan lebih fleksibel dalam penggunaannya.

3.4. Pembuatan Sistem Alat

Tahap pembuatan sistem alat ini dilakukan perakitan komponen elektronik dan mekanik dalam *enclosure* yang aman. Sensor-sensor dan aktuator diuji untuk memastikan fungsionalitasnya sebelum diintegrasikan. Proses ini juga mencakup pemrograman *hardware* menggunakan Arduino IDE dan membuat desain tampilan aplikasi Android menggunakan Flutter.

3.5. Integrasi Sistem Dengan Aplikasi Android

Tahap integrasi sistem dengan aplikasi Android merupakan proses di mana mikrokontroler dikoneksikan ke *Wi-Fi* untuk mengirim data ke *cloud* Firebase, lalu aplikasi Android mengambil data tersebut untuk ditampilkan dalam bentuk data monitoring yang sudah di desain sebelumnya. Pemrograman yang dilakukan pada *software* Arduino IDE harus terintegrasi dengan *code* di Flutter sehingga

dapat saling berkomunikasi dan menghasilkan *output* sesuai dengan yang telah ditentukan.

3.6. Perakitan *Hardware* Sistem Otomasi Alat ke Konstruksi

Tahap perakitan *hardware* sistem otomasi alat ke konstruksi ini merupakan perakitan *hardware* yang telah dibuat dan komponennya telah diuji untuk memastikan fungsionalitasnya ke konstruksi alat yang sudah dibuat dengan peletakan komponen sesuai rancangan, kemudian secara keseluruhan *hardware* dan *software* bekerja sesuai fungsi dan kegunaan alat.

3.7. Pengujian Alat dan Pengambilan Data

Alat diuji dalam berbagai kondisi, termasuk respon sensor, akurasi suhu dan kelembapan, serta uji kinerja pengeringan lada. Akurasi sensor suhu dan kelembapan akan dikomper dengan alat ukur hygrometer digital (HTC-1) yang terdapat pada alat pengeringan. Efektivaitas alat dalam mengeringkan lada, diketahui dengan pengujian proses pengeringan menggunakan alat yang dibuat dan penjemuran secara tradisional yaitu dengan menjemur lada dibawah sinar matahari dengan menggunakan alas karung atau terpal sejenisnya. Berdasarkan data yang dihasilkan akan dibandingkan seberapa efektif alat pengeringan dalam segi lama waktu pengeringan dan hasil akhir lada yang dikeringkan.

3.8. Analisis Data

Data yang diperoleh dari hasil pengujian sistem kemudian dianalisis secara menyeluruh untuk mengevaluasi tingkat konsistensi kinerja sistem serta efisiensi proses pengeringan yang dihasilkan. Analisis ini mencakup pengamatan terhadap respons sistem terhadap perubahan kondisi lingkungan, serta keandalan pengiriman data ke aplikasi Android dan LCD. Apabila ditemukan ketidaksesuaian antara hasil aktual dan target yang diharapkan, maka dilakukan langkah optimasi, seperti penyesuaian parameter kontrol, perbaikan desain perangkat keras, atau pembaruan logika sistem agar performa sistem tetap optimal dan sesuai tujuan perancangan.

BAB IV
HASIL DAN PEMBAHASAN

4.1. Alat Pengontrolan Suhu Dan Kelembapan Pada Pengeringan Lada IoT

Sistem pengontrolan suhu dan kelembapan untuk pengeringan lada ini menggunakan pendekatan berbasis *Internet of Things* (IoT). Mikrokontroler ESP32 berperan sebagai pusat pemrosesan data dari berbagai sensor sekaligus pengendali utama antara perangkat keras dan antarmuka pengguna. Data hasil pemantauan dapat dilihat secara *real-time* melalui LCD *display* maupun aplikasi Android. Sensor DHT22 digunakan untuk membaca suhu dan kelembapan, BH1750 untuk mengukur intensitas cahaya, dan sensor hujan (*raindrop*) untuk mendeteksi kondisi hujan.

Smartroof atau sistem atap otomatis digerakkan oleh motor DC JGA-25 370 yang dikendalikan melalui *driver motor L298N*. Sistem ini bekerja mengikuti logika yang ditanamkan ke dalam mikrokontroler berdasarkan kondisi lingkungan. Pemanas terdiri atas *heater fan* dan lampu halogen, dengan tambahan kipas DC untuk menjaga sirkulasi udara agar tetap merata saat atap dalam kondisi tertutup.

Aplikasi Android yang dibangun dengan Flutter berfungsi sebagai media pemantauan dan pengontrolan. Pengguna dapat melihat data sensor, mengetahui status atap dan pemanas, serta menjalankan kontrol manual melalui aplikasi. Seluruh data termasuk histori pemantauan disimpan secara otomatis di *Firebase Realtime Database* sebagai solusi penyimpanan berbasis *cloud*.

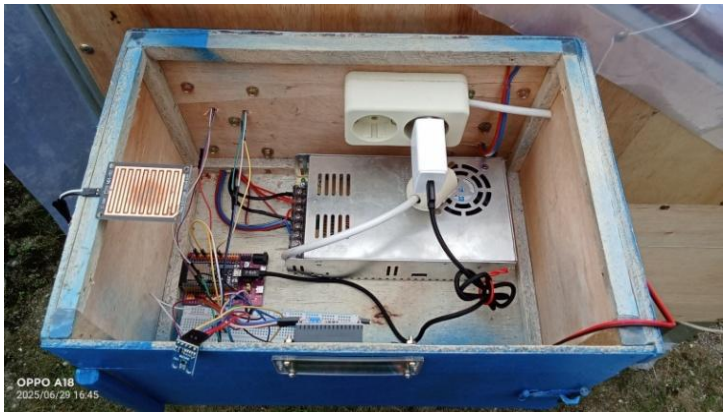
Tahap pembuatan alat ini dimulai dengan merakit konstruksi alat yaitu rangka pengering, wadah penjemuran, mekanisme atap, hingga penempatan komponen pada box kontrol. Proses perakitan dilakukan berdasarkan rancangan teknis yang telah dibuat sebelumnya. Gambar 4.1 menampilkan hasil akhir dari perangkat yang telah dirakit.

Gambar 4. 1 Konstruksi Alat dan Box Penyimpanan Komponen

Tahap selanjutnya dilakukan perakitan hardware yang bertujuan mengintegrasikan seluruh komponen elektronik ke dalam konstruksi fisik alat pengering lada. Sistem terdiri dari mikrokontroler ESP32, dua sensor DHT22 untuk membaca suhu dan kelembapan, sensor cahaya BH1750, serta raindrop sensor. Sensor-sensor ini dipasang di bagian dalam dan luar ruang pengering untuk memperoleh data lingkungan secara akurat, dengan dua sensor DHT22 ditempatkan di titik berbeda guna menghitung nilai rata-rata.

Motor DC JGA-25 370 digunakan sebagai aktuator atap, dikendalikan driver motor untuk membuka dan menutup atap secara otomatis selama 10 detik per aksi. Sistem pemanas yang terdiri dari heater fan, lampu halogen, dan kipas DC dikendalikan secara bersamaan melalui satu pin digital dengan mosfet switching saat atap tertutup.

Seluruh komponen seperti ESP32, driver motor, dan power supply dirakit dalam box. Sebagai antarmuka lokal, LCD 16x2 I2C menampilkan informasi suhu, kelembapan, intensitas cahaya, status mode, dan hujan, sehingga pengguna dapat memantau sistem langsung dari lokasi alat tanpa membuka aplikasi Android.



Gambar 4. 2 Perakitan *Hardware* pada Konstruksi

Tahap akhir yaitu pengujian alat yang dilakukan setelah semua sistem *monitoring* dan kontrol selesai dirakit pada konstruksi alat pengontrolan suhu dan kelembapan pada pengeringan lada berbasis IoT. Pengujian ini bertujuan untuk memastikan bahwa seluruh komponen sistem berfungsi dengan baik dan saling terintegrasi sesuai rancangan.

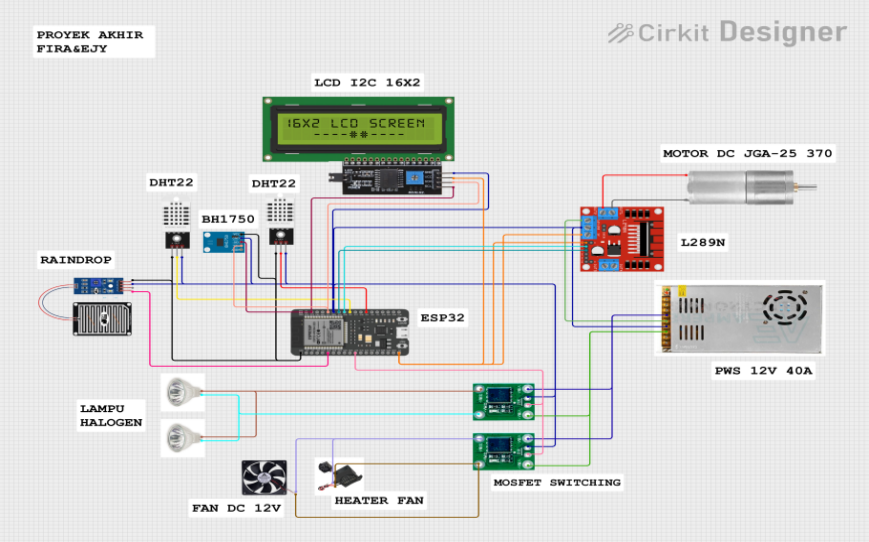
Kegiatan pengujian melibatkan serangkaian tahapan yang meliputi pemeriksaan fungsi sensor, pengujian respons sistem terhadap perubahan lingkungan, evaluasi kinerja aplikasi Android sebagai media *monitoring* dan kontrol, serta penilaian efektivitas alat dalam mempercepat proses pengeringan lada. Melalui tahapan ini, dapat diketahui sejauh mana alat yang dirancang mampu memenuhi tujuan utama, yaitu menghasilkan sistem pengering yang efisien, akurat, dan dapat dioperasikan secara otomatis maupun manual.

4.2. Skematik Rangkaian *Hardware*

Skematik rangkaian hardware untuk sistem pengontrol suhu dan kelembapan pada alat pengering lada berbasis *Internet of Things* (IoT) dirancang menggunakan *platform* Cirkit Designer IDE. *Platform* ini dipilih karena mendukung perancangan sirkuit elektronik secara visual, memudahkan dalam penempatan dan pengaturan koneksi antar komponen, serta terdapat fitur *custom parts* yaitu fitur untuk membuat komponen sesuai keinginan atau kebutuhan yang tidak ada dalam fitur *parts* bawaan *platform*. Setiap komponen dirangkai dengan

cermat berdasarkan konfigurasi pin yang sesuai dengan spesifikasi teknis masing-masing modul, seperti ESP32, sensor DHT22, sensor cahaya BH1750, *raindrop sensor*, driver motor L298N, serta perangkat pemanas dan aktuator lainnya.

Perancangan skematik ini menjadi tahap penting dalam memastikan integrasi sistem berjalan dengan baik sebelum dilakukan perakitan fisik. Koneksi antara pin *input/output* ESP32 dengan komponen lain juga diperhatikan untuk menghindari kesalahan wiring saat implementasi. Skematik rangkaian yang telah selesai dirancang kemudian dijadikan acuan utama dalam proses perakitan *hardware*. Gambar 4.3 menunjukkan secara detail rancangan koneksi antar komponen pada sistem alat ini yang menjadi dasar dari realisasi fisik perangkat.



Gambar 4. 3 Skematik Rangkaian *Hardware* Alat

4.3. Tampilan Aplikasi Android Flutter

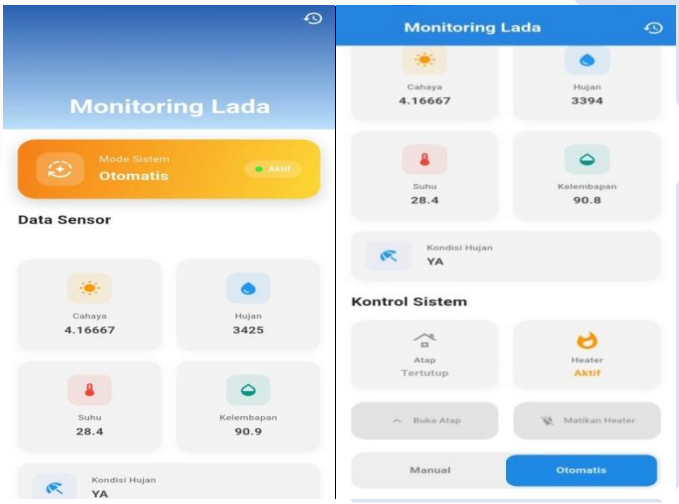
Antarmuka aplikasi Android dikembangkan menggunakan *framework* Flutter karena mendukung proses pembuatan *aplikasi mobile* yang cepat, ringan, dan kompatibel lintas *platform*. Flutter menggunakan bahasa pemrograman Dart serta menawarkan berbagai *widget* yang mempermudah dalam membangun tampilan yang interaktif dan responsif. Aplikasi ini berfungsi sebagai penghubung antara pengguna dan alat pengering, memungkinkan pemantauan serta

pengendalian sistem secara jarak jauh melalui integrasi dengan *Firebase Realtime Database.*

Struktur tampilan aplikasi dibagi menjadi dua halaman utama (*screen*) yaitu sebagai berikut:

- *Screen monitoring* dan kontrol

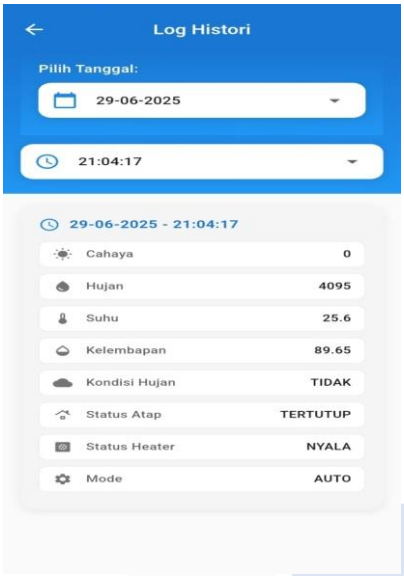
Layar pertama berfungsi sebagai panel *monitoring* dan kontrol, menampilkan data sensor secara *real-time* seperti suhu rata-rata, kelembapan rata-rata, intensitas cahaya, dan kondisi hujan. Informasi ini diperoleh dari Firebase dan ditampilkan dalam format teks yang diperbarui secara berkala. Selain itu, status sistem seperti status atap, status *heater*/pemanas, dan mode kontrol (otomatis atau manual) juga ditampilkan. Jika mode manual aktif, tombol kontrol akan muncul untuk membuka atau menutup atap serta menyalakan atau mematikan *heater*/pemanas secara langsung.



Gambar 4. 4 Tampilan *Monitoring* pada Smartphone

- *Screen History*

Layar kedua difokuskan pada tampilan histori data yang ditampilkan berdasarkan tanggal. Data histori diambil dari Firebase dengan struktur *path* yang telah ditentukan, kemudian diproses dan ditampilkan menggunakan elemen seperti *ListView* dan *Card*. Data yang ditampilkan mencakup waktu pencatatan, nilai sensor, dan status sistem saat itu, sehingga pengguna dapat melihat riwayat pengoperasian alat.



Gambar 4. 5 Tampilan Histori pada Smartphone

4.4. Pengujian komunikasi ESP32 ke LCD dan Aplikasi Android

Pengujian komunikasi antara ESP32 dengan LCD dan aplikasi Android dilakukan untuk memastikan bahwa data yang dihasilkan dari sistem sensor dapat ditampilkan dan disinkronkan secara *real-time* ke dua media *output* utama yaitu LCD 16x2 I2C sebagai tampilan lokal dan aplikasi Android Flutter sebagai tampilan jarak jauh.

a.) Pengujian Komunikasi ESP32 ke LCD

LCD 16x2 I2C digunakan untuk menampilkan informasi penting seperti suhu, kelembapan, intensitas cahaya, kondisi hujan, dan mode sistem. Pengujian dilakukan dengan langkah berikut:

- ESP32 membaca data dari dua sensor DHT22 dan menghitung rata-rata suhu serta kelembapan.
- Data suhu, kelembapan, intensitas cahaya, dan kondisi hujan dari *raindrop sensor* ditampilkan ke LCD.
- Mode sistem juga ditampilkan sebagai huruf “A” (Auto/Otomatis) atau “M” (Manual)
- Hasil tampilan diverifikasi dengan mencocokkan nilai yang tampil di LCD dengan hasil pembacaan sensor yang ditampilkan di Serial Monitor.

Tabel 4. 1 Pengujian Komunikasi ESP32 ke LCD

Percobaan	Tampilan LCD	Serial Monitor	Waktu Respon
1		Output Serial Monitor Message (Enter to send message to 'ESP32 ====== DATA SENSOR ===== Suhu Rata-rata: 28.20 °C Kelembapan Rata-rata: 90.30 % Cahaya: 4.17 lux Hujan: 4095 (ADC) Kondisi Hujan: TIDAK === Status Sistem === Status Atap: TERTUTUP Status Heater: NYALA Mode: AUTO	1 – 2 detik
2		====== DATA SENSOR ====== Suhu Rata-rata: 28.30 °C Kelembapan Rata-rata: 90.15 % Cahaya: 4.17 lux Hujan: 2327 (ADC) Kondisi Hujan: YA === Status Sistem === Status Atap: TERTUTUP Status Heater: NYALA Mode: AUTO	1 – 2 detik
3		tput Serial Monitor × essage (Enter to send message to 'ESF ===== DATA SENSOR ===== ıhu Rata-rata: 28.40 °C elembapan Rata-rata: 91.00 % ahaya: 577.50 lux ıjan: 4095 (ADC) ondisi Hujan: TIDAK = Status Sistem === tatus Atap: TERBUKA tatus Heater: MATI ode: AUTO	2 detik
4		Output Serial Monitor × Message (Enter to send message to 'ESP32 ====== DATA SENSOR ====== Suhu Rata-rata: 28.40 °C Kelembapan Rata-rata: 91.00 % Cahaya: 577.50 lux Hujan: 3103 (ADC) Kondisi Hujan: YA === Status Sistem === Status Atap: TERTUTUP Status Heater: NYALA Mode: AUTO	2 detik
5		====== DATA SENSOR ===== Suhu Rata-rata: 28.40 °C Kelembapan Rata-rata: 90.90 % Cahaya: 4.17 lux Hujan: 3389 (ADC) Kondisi Hujan: YA === Status Sistem === Status Atap: TERTUTUP Status Heater: NYALA Mode: AUTO	1 – 2 detik

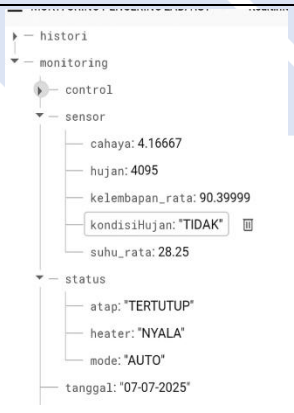
Berdasarkan hasil data pengujian pada Tabel 4.1, LCD mampu menampilkan data secara dinamis setiap 1 hingga 2 detik dan sinkronisasi data pada LCD dengan serial monitor berjalan konsisten dengan tingkat akurasi ±99,8%. Format tampilan mudah dibaca dan responsif terhadap perubahan kondisi lingkungan.

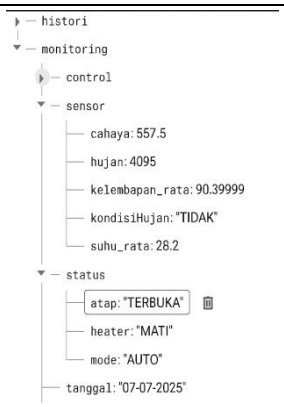
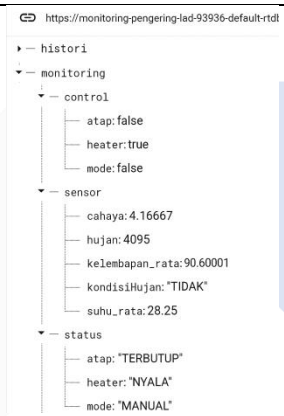
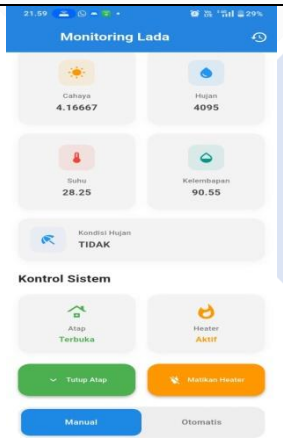
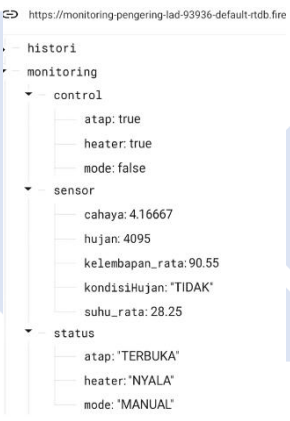
b.) Pengujian Komunikasi ESP32 ke Aplikasi Android

Komunikasi antara ESP32 dan aplikasi Android dilakukan menggunakan *Firebase Realtime Database* sebagai perantara. Proses pengujian dilakukan dari ESP32 ke Aplikasi dengan langkah sebagai berikut:

- ESP32 mengirimkan data suhu, kelembapan, intensitas cahaya, kondisi hujan, status *heater*, atap, dan mode ke Firebase pada *path monitoring*.
- Aplikasi Android mengambil data tersebut dan menampilkannya pada *Screen pertama* yaitu tampilan *monitoring* dan kontrol sistem**.**
- Data juga disimpan ke *path* histori di Firebase dengan format /histori/tanggal/waktu dan ditampilkan pada aplikasi Android *screen history*.

Tabel 4. 2 Pengujian Komunikasi ESP32 ke Aplikasi Android (*Monitoring*)

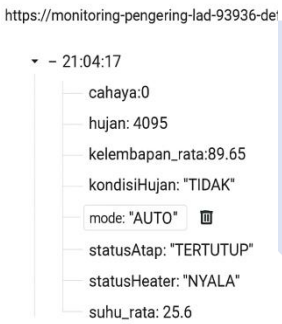
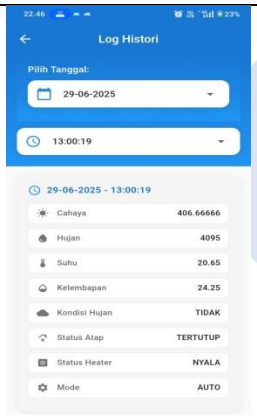
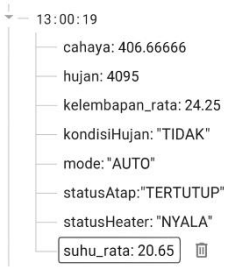
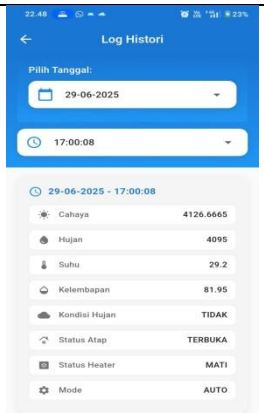
Percobaan	Tampilan Aplikasi	Data Firebase	Keterangan
1			Data yang ditampilkan menunjukkan integrasi berjalan dengan baik antara aplikasi dan firebase.

2			Data yang ditampilkan menunjukkan integrasi berjalan dengan baik antara aplikasi dan firebase.
3			Data yang ditampilkan menunjukkan integrasi berjalan dengan baik antara aplikasi dan firebase.
4			Data yang ditampilkan menunjukkan integrasi berjalan dengan baik antara aplikasi dan firebase.

Berdasarkan Tabel 4.2 di atas, data dari ESP32 dapat diterima dan ditampilkan di aplikasi Android dengan jeda yang sangat minim, yaitu kurang dari 2 detik. Hal ini menunjukkan bahwa proses sinkronisasi data secara *real-time* antara ESP32, *Firebase Realtime Database*, dan aplikasi Android berjalan dengan sangat baik dan stabil. Setiap pembaruan data sensor seperti suhu, kelembapan, intensitas cahaya, dan kondisi hujan dapat langsung dikirim dan diterima oleh

aplikasi tanpa mengalami keterlambatan yang signifikan. Respons sistem yang cepat ini sangat penting untuk memastikan pengguna dapat memantau kondisi lingkungan pengering lada secara akurat. Dengan performa sinkronisasi yang tinggi ini, sistem dapat dikategorikan sebagai sistem pemantauan dan kendali berbasis IoT yang responsif dan *real-time*.

Tabel 4. 3 Pengujian Komunikasi ESP32 dengan Aplikasi Android (Histori)

Percobaan	Tampilan Aplikasi	Data Firebase	Keterangan
1			Histori data yang ditampilkan menunjukkan integrasi berjalan dengan baik antara aplikasi dan firebase.
2			Histori data yang ditampilkan menunjukkan integrasi berjalan dengan baik antara aplikasi dan firebase.
3			Histori data yang ditampilkan menunjukkan integrasi berjalan dengan baik antara aplikasi dan firebase.

Berdasarkan hasil tampilan aplikasi android yang terkoneksi dengan sistem alat menggunakan Flutter, antarmuka pengguna dibentuk melalui kombinasi berbagai widget yang merepresentasikan komponen visual seperti teks, bentuk, dan animasi. Setiap elemen tampilan disusun dari widget sederhana yang dapat digabung menjadi struktur yang lebih kompleks. Rancangan aplikasi yang dibuat telah mempertimbangkan keterbatasan performa perangkat, desain antarmuka yang disesuaikan dengan kebutuhan pengguna serta tampilan yang sederhana, menarik, dan mudah digunakan sesuai dengan penelitian [22].

4.5. Pengujian Sistem Kontrol Otomatis dan Manual

Pengujian sistem kontrol otomatis dilakukan untuk memastikan bahwa ESP32 dapat mengatur buka/tutup atap dan nyala/mati pemanas secara mandiri berdasarkan data dari sensor suhu (DHT22), cahaya (BH1750), dan *raindrop sensor*. Selain itu, dilakukan juga uji kontrol manual yang dioperasikan melalui aplikasi Android untuk buka/tutup atap dan nyala/mati heater.

Sistem kontrol otomatis dirancang agar atap terbuka jika intensitas cahaya > 50 lux, tidak hujan (nilai sensor > 3500), dan suhu ≥ 25°C. Sebaliknya, atap akan tertutup jika terdeteksi hujan atau suhu < 25°C. Saat atap tertutup, *heater* otomatis menyala untuk menjaga suhu dalam ruang pengering tetap optimal. Jika atap terbuka, *heater* otomatis dimatikan.

Tabel 4. 4 Pengujian Sistem Otomatis

Kondisi	Aksi Sistem	Status Atap	Status Heater	Waktu Respons
Cahaya tinggi, suhu tinggi, tidak hujan (Normal)	Motor bergerak membuka atap	Terbuka	Mati	2 detik
Hujan terdeteksi	Motor bergerak menutup atap	Tertutup	Nyala	2 detik

Suhu < 25°C	Motor bergerak menutup atap	Tertutup	Nyala	2 detik
Cahaya redup dan tidak hujan (mendung)	Motor bergerak menutup atap	Tertutup	Nyala	3 detik

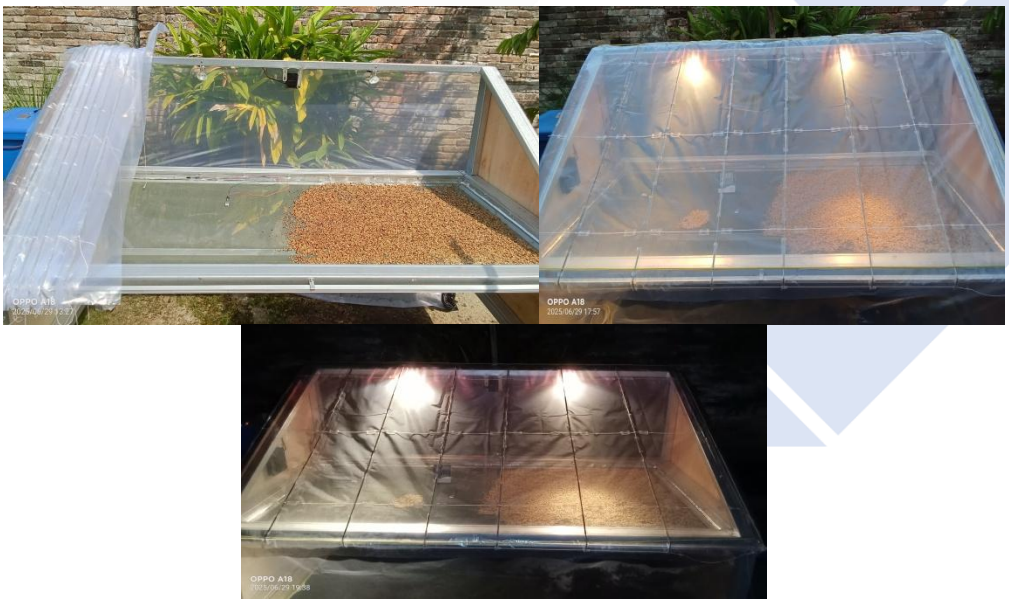
Berdasarkan hasil pengujian, sistem otomatis menunjukkan kemampuan merespons perubahan kondisi lingkungan secara cepat dan tepat, dengan waktu respons sekitar 1–3 detik. Hal ini membuktikan bahwa logika otomatis yang ditanamkan dalam mikrokontroler ESP32 telah berjalan sesuai fungsi, baik dalam pengoperasian atap maupun sistem pemanas berdasarkan parameter suhu, kelembapan, intensitas cahaya, dan hujan. Responsivitas sistem memberikan kontribusi besar terhadap efisiensi proses pengeringan. Meskipun demikian, performa sistem masih bergantung pada kestabilan koneksi internet, mengingat proses komunikasi data dilakukan melalui jaringan Wi-Fi. Ketergantungan ini menjadi salah satu aspek yang perlu diperhatikan untuk pengembangan lebih lanjut, terutama jika sistem akan diterapkan di lokasi dengan jaringan yang kurang stabil.

Tabel 4. 5 Pengujian Sistem Manual

Kontrol Manual Atap	Kontrol Manual Heater	Aksi Sistem	Status Atap	Status Heater	Waktu Respons
Buka	Nyala	Motor bergerak membuka atap	Terbuka	Nyala	7 detik
Buka	Mati	Motor bergerak membuka atap	Terbuka	Mati	6 detik

Tutup	Nyala	Motor bergerak menutup atap	Tertutup	Nyala	5 detik
Tutup	Mati	Motor bergerak menutup atap	Tertutup	Mati	5 detik

Hasil pengujian pada Tabel 4.5 menunjukkan bahwa sistem manual dapat merespons *input* dari pengguna melalui tombol kontrol pada aplikasi Android dalam waktu sekitar 5 hingga 7 detik. Respons ini menandakan bahwa fungsi sistem berjalan dengan baik, meskipun terdapat jeda yang disebabkan oleh proses pengiriman data melalui jaringan internet. Kualitas koneksi *Wi-Fi* menjadi faktor utama yang memengaruhi kecepatan respon, terutama saat sinyal tidak stabil.



Gambar 4. 6 Pengujian Sistem Otomatis dan Manual

4.6. Pengambilan Data Pengujian Pengeringan Lada

Proses pengambilan data dilakukan untuk menilai kinerja sistem dalam membaca dan mengontrol suhu, kelembapan, serta kondisi lingkungan selama pengeringan lada. Data dikirim setiap 5 detik ke Firebase, lalu ditampilkan secara

real-time di LCD, Serial Monitor, dan aplikasi Android. Selama pengujian, seluruh data otomatis tersimpan ke Firebase berdasarkan tanggal dan waktu.

Gambar 4. 7 Pengujian Pengeringan Lada

Pengambilan data pengujian dilakukan dengan 2 parameter uji yaitu yang pertama mengeringkan lada sebanyak ± 1000 butir dan yang kedua mengeringkan lada sebanyak ± 1,5 kg menggunakan alat dan pengeringan secara tradisional. Pengujian pertama (± 1000 butir lada), lada yang sebelum dikeringkan ditimbang menggunakan tibangan digital untuk mengetahui efektifitas alat pengeringan melalui hasil timbang berat lada dengan kadar air sebelum pengeringan dan setelah pengeringan. Proses penimbangan lada sebelum dikeringkan dapat dilihat pada Gambar 4.15 dan 4.16. Hasil penimbangan berat lada saat setelah dikeringkan dapat dilihat pada Gambar 4.17.

Gambar 4. 8 Kalibrasi Timbangan

Gambar 4. 9 Timbangan Berat Lada sebelum dikeringkan

Gambar 4. 10 Timbangan Berat Lada setelah dikeringkan

Tabel 4. 6 Hasil Pengujian Pengeringan Lada (± 1000 butir)

Hari ke-	Pengeringan dengan Alat		Pengeringan Tradisional (Konvensional)	
	Berat Awal	Berat Akhir	Berat Awal	Berat Akhir
	53,0 gr	35,2 gr	51,3 gr	43,2 gr
1 (13.00)				
(17.00)				

2 (08.00)		
(13.00)		

Berdasarkan hasil pengujian yang ditunjukan pada Tabel 4.6, Alat pengering bekerja dengan baik. Dapat dilihat dari hasil pengeringan, perubahan warna pada lada selama proses pengeringan sangat tterlihat ketika masih basah hingga kering. Pengeringan dilakukan selama 2 hari dengan hasil pengeringan oleh alat pengering dapat menurunkan kadar air 33,58% dari berat awal 53,0gr ke 35,2gr/ ±1000 butir lada. Perhitungan presentase pengurangan kadar air sebagai berikut [6][25][26]:

$$\text{Persentase Pengurangan Kadar Air} = \frac{\text{Berat Awal} - \text{Berat Akhir}}{\text{Berat Awal}} \times 100\%$$

$$\text{Persentase} = \frac{53{,}0 - 35{,}2}{53{,}0} \times 100\% = \frac{17{,}8}{53{,}0} \times 100\% = 33{,}58\%$$

Hasil pengeringan tradisional (konvensional) dapat menurunkan kadar air 15,79% dari berat awal 51,3gr ke 43,2gr/ ±1000 butir lada. Perhitungan presentase pengurangan kadar air sebagai berikut:

$$\text{Persentase} = \frac{51{,}3 - 43{,}2}{51{,}3} \times 100\% = \frac{8{,}1}{51{,}3} \times 100\% = 15{,}79\%$$

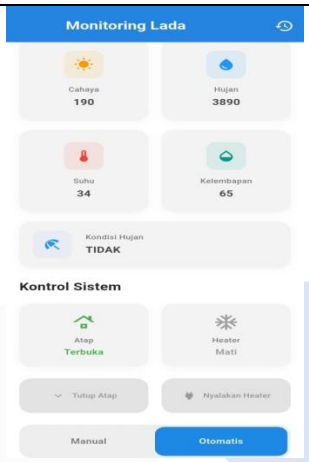
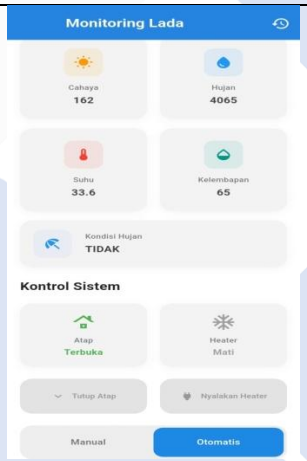
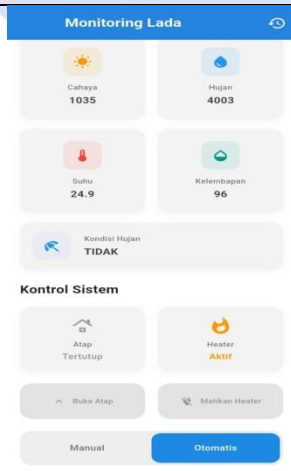
Hasil di atas menunjukan bahwa pengeringan lada menggunakan alat ini lebih cepat kering karena dibantu oleh elemen *heater* saat kondisi cuaca hujan, mendung ataupun pada malam hari serta sistem alat yang responsif terhadap perubahan kondisi cuaca dapat membantu petani dalam menghadapi resiko gagal kering saat cuaca hujan.

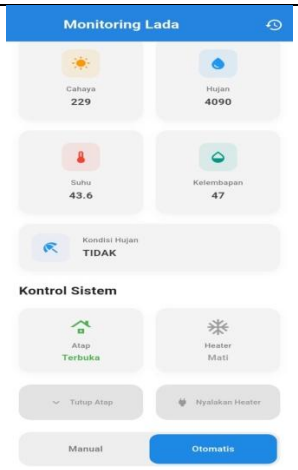
Pengujian kedua yaitu mengeringkan lada sebanyak ± 1,5 kg yang dilakukan bersamaan dengan pengujian pertama namun peletakan lada dipisah agar dapat membandingkan waktu pengeringan dengan kuantitas lada yang lebih banyak dan lebih sedikit.

Tabel 4. 7 Pengujian Pengeringan Lada (± 1,5kg)

Hari ke-	Pengeringan dengan Alat	Pengeringan Tradisional (Konvensional)
1 (13.00)		
(17.00)		
2 (08.00)		
(13.00)		

Tabel 4. 8 Data Pengujian Sistem Pengeringan Lada Berbasis IoT

Hari ke-	Data di LCD	Data di Aplikasi Android	Alat ukur suhu dan kelembapan (HTC-1)
1 (13.00)			
(17.00)			
2 (08.00)			

(13.00)			

Berdasarkan Tabel 4.6 dan 4.7 di atas, terlihat jelas perbedaan warna lada dengan perubahan per waktu saat proses pengeringan menggunaakan alat yang di buat dengan pengeringan secara tradisional (konvensional) untuk kuantitas ±1000 butir dan ±1,5 kg lada. Hasil pengeringan dengan alat lebih merata dan lebih cepat karena penggunaan *heater fan* dan lampu halogen sebagai pemanas bisa mendapatkan hasil pengeringan yang lebih merata seperti penelitian yang dilakukan oleh [19][21].

Hasil pengujian juga menunjukkan bahwa sistem *monitoring* pada alat pengering mampu melakukan pengukuran dan menampilkan data suhu secara langsung melalui LCD *display* dan aplikasi android, dengan tingkat error sebesar 0,5%, yang berarti memiliki akurasi mencapai 99,5% mendekati hasil pada penelitian [4].

Data yang tampil pada LCD, aplikasi Android, dan alat ukur suhu kelembapan (HTC-1) terlihat sinkron tetapi sewaktu-waktu terjadi perbedaan yang cukup signifikan pada nilai suhu dan kelembapan karena DHT22 memiliki akurasi kelembapan sekitar ±2–5% RH, dan suhu ±0.5°C, sedangkan HTC-1 seringkali tidak dikalibrasi secara pabrik dengan standar tinggi, sehingga bisa memiliki selisih lebih besar, bahkan hingga ±10% RH.

4.7. Perhitungan Penggunaan Daya

Perhitungan penggunaan daya pada alat pengering lada berbasi *Internet of Things* ini diambil *study case* pengeringan menggunakan pemanas hanya pada saat malam hari. Atap tertutup untuk mencegah masuknya kelembapan dari lingkungan luar. Pemanas dan kipas sirkulasi aktif untuk menjaga suhu ruang pengering selama 13 jam, dari pukul 18.00 hingga 07.00 WIB.

Mikrokontroler ESP32 berfungsi sebagai pusat kendali yang membaca data dari dua sensor DHT22, sensor BH1750, dan sensor hujan analog (*raindrop sensor*). Data ditampilkan melalui LCD I2C 16x2 dan aplikasi android sebagai informasi pemantauan. Konsumsi daya dari sensor dan mikrokontroler relatif kecil, namun tetap dihitung karena bekerja terus-menerus selama sistem aktif.

Beban daya terbesar berasal dari *heater fan* berdaya 120 watt dan dua lampu halogen masing-masing 50 watt. Kipas DC 3,6 watt menyala bersamaan untuk membantu distribusi panas di dalam ruang pengering. Motor DC hanya aktif dua kali dalam sehari selama total 1 menit (0,0167 jam), sehingga konsumsi dayanya sangat kecil dibandingkan komponen lain.

Tabel 4. 9 Perhitungan Daya Komponen Sistem

No	Komponen	Jumlah	Daya/Unit (W)	Waktu Aktif (jam)	Energi (Wh)
1	ESP32	1	1	13	13
2	Sensor DHT22	2	0.025	13	0.65
3	Sensor BH1750	1	0.005	13	0.065
4	*Raindrop sensor*	1	0.1	13	1.3
5	LCD 16x2 I2C	1	0.15	13	1.95
6	Heater Fan PTC 12V	1	120	13	1560
7	Lampu Halogen 50W 12V	2	50	13	1300
8	Kipas DC 12V	1	3.6	13	46.8
9	Motor DC Atap (2x30 detik)	1	14.4	0.0167	0.24
	Total Energi				2923.97 Wh

Berdasarkan Tabel 4.9, sistem pengering lada berbasis IoT mengonsumsi total energi sebesar 2923.97 Wh atau 2.92 kWh selama 13 jam operasi di malam

hari dengan kondisi cuaca lembap. Apabila dihitung penggunaan dalam 1 jam alat beroperasi mengonsumsi energi sebesar 225.145 Wh atau 0.22515 kWh**.** Konsumsi daya terbesar berasal dari *heater fan* (120 W) dan dua lampu halogen (masing-masing 50 W) yang menyumbang lebih dari 97% dari total energi harian.

Alat pengering ini dirancang untuk mengeringkan lada dalam kapasitas sekitar 5–7 kg per siklus (13 jam *heater* menyala)**.** Dengan demikian, estimasi efisiensi energi per kilogram (*Specific Energy Consumption* (SEC)) adalah sebagai berikut:

$$\text{Efisiensi Energi Spesifik} = \frac{\text{Energi total yang digunakan (kWh)}}{\text{Berat bahan yang dikeringkan (kg)}}$$

$$\text{Efisiensi Energi} = \frac{2.92\ \text{kWh}}{5\ \text{kg}} = 0.584\ \text{kWh/kg} \quad \text{s. d.} \quad \frac{2.92\ \text{kWh}}{7\ \text{kg}} = 0.417\ \text{kWh/kg}$$

Artinya, konsumsi energi berkisar antara 0.42–0.58 kWh per kg lada, tergantung jumlah bahan yang diproses. Sedangkan oven pengering konvensional umumnya menggunakan daya 1000–1500 W, dengan waktu pengeringan sekitar 6–8 jam. Maka konsumsi energi per sesi berkisar 6–12 kWh. Jika kapasitas pengeringannya sama yaitu 5-7 kg, maka energi per kg lada bisa mencapai 0.86–2.4 kWh/kg, jauh lebih tinggi dibanding alat pengering dengan sistem IoT yang dibuat.

Sistem pengering lada menggunakan teknologi IoT memiliki tingkat efisiensi energi yang jauh lebih baik dibandingkan oven pengering konvensional. Konsumsi energi harian sebesar 2.92 kWh menjadikan alat ini lebih ekonomis dan mampu menekan biaya operasional. Pemanfaatan sensor serta kendali melalui aplikasi memberikan fleksibilitas tinggi dalam pemantauan dan pengaturan sistem.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil pengujian alat dapat diambil kesimpulan sebagai berikut:

- Alat yang dibuat memiliki respons yang cukup cepat sekitar 1-2 detik terhadap perubahan cuaca. Sensor cahaya BH1750 dan *raindrop sensor* dapat mengontrol sistem dengan baik, serta sistem dapat mengukur, membaca, dan menampilkan hasilnya pada LCD *display* dan aplikasi Android dengan tingkat presentase *error* sensor DHT22 0,5% untuk suhu dan 5% untuk kelembaban.
- Alat yang dirancang dan dibuat dengan sistem *monitoring* dan kontrol berbasis *Internet of Things* (IoT) telah terintegrasi antara ESP32 dengan *interface* pengguna yaitu aplikasi Android dan LCD sehingga dapat memudahkan pekerjaan petani dalam memantau dan mengontrol proses pengeringan lada.
- Perbandingan antara hasil pengeringan dengan alat yang dibuat dan pengeringan secara tradisional (konvensional) dapat dilihat dari hasil penimbangan berat lada (± 1000 butir) sebelum dikeringkan dan setelah dikeringkan yaitu dengan alat pengering dapat menurunkan kadar air 33,58% sedangkan pengeringan tradisional hanya menurunkan kadar air 15,79%. Waktu pengeringan secara keseluruhan pada alat lebih cepat yaitu dalam waktu 2 hari, sedangkan pengeringan tradisional membutuhkan waktu 3-4 hari.

Alat pengering lada berbasis IoT yang dikembangkan terbukti efektif dalam mempercepat proses pengeringan dan meningkatkan efisiensi kerja petani. Integrasi antara sensor, mikrokontroler ESP32, Firebase dan aplikasi Android memberikan kemudahan dalam pemantauan serta kontrol jarak jauh secara *real-time*. Hasil pengujian menunjukkan bahwa sistem ini mampu menjadi solusi alternatif yang lebih modern dan efisien dibandingkan metode pengeringan tradisional.

5.2. Saran

Berdasarkan penelitian yang telah dilakukan dapat dikembangkan dan dioptimalkan dalam beberapa hal seperti berikut:

- Pemilihan bahan konstruksi alat: pemilihan bahan yang tepat seperti bahan untuk atap agar pergerakan buka dan tutup sistem *smart roof* dapat berjalan lancar.
- Kinerja Sistem Internet of Things (IoT) : kinerja masih bergantung pada kestabilan koneksi internet, sehingga perlu dipertimbangkan solusi tambahan untuk lokasi dengan jaringan tidak stabil seperti fitur mode *offline* atau sistem penyimpanan data sementara secara lokal untuk menjamin kontinuitas sistem.
- Limit Switch untuk Posisi Atap : tambahkan dua limit switch (bukaan penuh & tutupan penuh) di mekanisme atap. Bila motor mencapai salah satu posisi, ESP32 otomatis memutus arus motor sehingga dapat menghindari overrun dan keausan gigi.
- Integrasi Notifikasi *Real-Time:* Tambahkan modul notifikasi di aplikasi Android (via Notifier atau Push Notification) untuk peringatan otomatis saat terjadi kondisi diluar kendali seperti hujan deras, suhu di bawah batas aman, atau heater gagal menyala, sehingga respon pengguna lebih cepat dalam menanggapi hal tersebut.

DAFTAR PUSTAKA

[1] Pusat Data dan Sistem Informasi Pertanian, “Outlook Komoditas Lada,” Pusat Data dan Sistem Informasi Pertanian Sekretariat Jenderal Kementerian Pertanian 2022. Accessed: Apr. 27, 2025. [Online]. Available: https://satudata.pertanian.go.id/assets/docs/publikasi/FINAL_OUTLOOK_LADA_2022.pdf

[2] F. Kartawiria, D. Hapsari, R. Hutami, S. Nurhalimah, F. Uzwatania, and K. Sapanli, “Proses Produksi Lada Putih Bangka Bubuk (Muntok White Pepper) Di PT . Izzah Globalindo Indonesia - Bangka Selatan,” vol. 3, pp. 11305–11311, 2024.

[3] A. A. Salim and E. Fitriani, “Rancang Bangun Alat Ukur Kadar Air Dan Pengering Biji-Bijian Berbasis Mikrokontroller,” *Electr. J. Rekayasa dan Teknol. Elektro*, vol. 18, no. 2, pp. 121–127, 2024, doi: 10.23960/elc.v18n2.2611.

[4] R. Pratama, N. Imron, I. Feriadi, and Z. Saputra, “Perancangan dan Pengujian Sistem Monitoring Operasi Alat Pengering Lada Menggunakan Teknologi Arduino,” *Pros. Semin. Nas. Inov. Teknol. Terap. - SNITT 2022*, pp. 92–98, 2022.

[5] H. Natawidjaya, M. U. Ametung, N. Nurjannah, Nuraini, and S. Sitompul, Pedoman Teknis Penanganan Pascapanen Lada. 2012. [Online]. Available: https://repository.pertanian.go.id/bitstreams/82ed5ad0-0728-49a6-a961-d9f3be404f59/download

[6] M. A. Ningrum, “Pengeringan Lada (Piper Nigrum L.) Petik Hijau Dengan Alat Pengering Hybrid Tipe Rak,” Universitas Lampung Bandar Lampung, 2022.

[7] O. P. A. Amane, R. W. Febriana, A. Riduan, and A. O. Cahyaningrum,

Penerapan Dan Pemanfaatan *Internet Of Things* (IoT) Di Berbagai Bidang. PT. Sonpedia Publishing Indonesia, 2023. [Online]. Available: https://books.google.co.id/books?id=8zWqEAAAQBAJ&lpg=PP1&lr&pg=PR1#v=onepage&q&f=false

[8] G. Heru Sandi and Y. Fatma, "Pemanfaatan Teknologi Internet of Things (Iot) Pada Bidang Pertanian," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 7, no. 1, pp. 1–5, 2023, doi: 10.36040/jati.v7i1.5892.

[9] N. P. Sari, N. M. Adriansyah, and V. S. W. Prabowo, "Aplikasi Monitoring Data Cuaca Pada Smart Weather Station Menggunakan MIT App Inventor," *e-Proceeding Eng.*, vol. 11, no. 6, pp. 6514–6520, 2024, [Online]. Available: https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/download/25007/23954/49423

[10] Fujiyama, "Sistem Kendali: Pengertian, Sejarah, Bentuk Dasar, Komponen Utama, dan Contohnya," Telkom University Surabaya. Accessed: Apr. 27, 2025. [Online]. Available: https://surabaya.telkomuniversity.ac.id/sistem-kendali-pengertian-sejarah-bentuk-dasar-komponen-utama-dan-contohnya/

[11] A. Maharani, R. Kusumanto, and A. Rahman, "Sistem Monitoring Suhu Dan Kelembaban Pada Alat Pengering Padi Berbasis Solar Cell," *J. Teliska*, vol. 16, no. 2, pp. 39–44, 2023, doi: 10.5281/zenodo.8188076.

[12] N. Nurdiana, A. Azis, and P. Perawati, "Perancangan Pengendali Temperatur pada Alat Pengering Makanan Berbasis IoT," *Electrician*, vol. 16, no. 3, pp. 247–252, 2022, doi: 10.23960/elc.v16n3.2263.

[13] F. Saputra, D. Ryana Suchendra, and M. Ikhsan Sani, "Implementasi Sistem Sensor Dht22 Untuk Menstabilkan Suhu Dan Kelembapan Berbasis Mikrokontroller Nodemcu Esp8266 Pada Ruangan Implementation of Dht22 Sensor System To Stabilize Temperature and Humidity Based on Microcontroller Nodemcu Esp8266 in Space," *Proceeding Appl. Sci.*, vol.

6, no. 2, p. 1977, 2020.

[14] A. Khuriati, “Sistem Pemantau Intensitas Cahaya Ambien dengan Sensor BH1750 Berbasis Mikrokontroler Arduino Nano,” *Berk. Fis.*, vol. 25, no. 13, pp. 105–110, 2022, [Online]. Available: https://ejournal.undip.ac.id/index.php/berkala_fisika/article/download/50740/22683

[15] T. B. Bano, I. G. A. Widagda, N. L. P. Trisnawati, I. M. S. Wibawa, I. K. Putra, and I. N. Sandi, “Perancangan Alat Ukur Intensitas Cahaya menggunakan Sensor BH1750 Berbasis Mikrokontroler ATMega328P,” *Kappa J.*, vol. 8, no. 1, pp. 95–101, 2024, doi: https://doi.org/10.29408/kpj.v8i1.24917.

[16] A. Lestari and E. Abdulrahman, “Rancang Bangun Modul Raindrop Dan IoT Sebagai Pengendali Penjemur Jagung Marning,” *JTERAF (Jurnal Tek. Elektro Raflesia)*, vol. 1, no. 2, 2021, [Online]. Available: https://ejournal.polraf.ac.id/index.php/JTERAF/article/view/110

[17] Seeedstudio, “JGA25-370 Geared Motor,” media.digikey.com. Accessed: Jul. 04, 2025. [Online]. Available: https://media.digikey.com/pdf/Data Sheets/Seeed Technology/114090046_Web.pdf

[18] E. Aris Prastyo, “Driver Motor L298N,” Arduino Indonesia. Accessed: Apr. 27, 2025. [Online]. Available: https://www.arduinoindonesia.id/2022/10/driver-motor-l298n.html

[19] R. H. Borrong, Z. Arifin, and V. M. Afma, “Penggunaan Heater Sebagai Alternatif Efektif Untuk Pemanas Pada Mesin Pengering Ikan Dan Kerupuk,” *J. Profisiensi*, vol. 12, no. 2, pp. 145–153, 2024, [Online]. Available: https://www.journal.unrika.ac.id/index.php/jurnalprofisiensi/article/download/6931/pdf

[20] N. Romana, “Rancang Bangun Aplikasi Pantau Dan Kendali Mobile Mesin

Pengering Biji Kopi Berbasis *Internet Of Things*," *Laporan Akhir Proyek Akhir*. 2024.

[21] B. P. Purnomo and A. Fzahruddin, "Perancangan dan Pengujian Cabinet Pengering Jagung Berbasis Lampu Pijar Holagen untuk Meningkatkan Efisiensi dan Kebersihan Proses Pengolahan," *Innov. Technol. Methodical Res. J.*, vol. 3, no. 3, p. 9, 2024, doi: 10.47134/innovative.v3i3.106.

[22] A. N. S. K. Putra and M. G. Ramdani, "Desain Internet of Things Pada Sistem Monitoring Kendali Lampu Berbasis Web Dan Mobile Application," *Inform. Mulawarman J. Ilm. Ilmu Komput.*, vol. 18, no. 2, p. 105, 2023, doi: 10.30872/jim.v18i2.5856.

[23] J. M. Suhendro, M. Sudarma, and D. C. Khrisne, "Rancang Bangun Aplikasi Seluler Penyedia Jasa Perawatan Dan Kecantikan Menggunakan Framework Flutter," *J. SPEKTRUM*, vol. 8, no. 2, pp. 68–82, 2021.

[24] M. F. Nur'ilham, A. Novianty, and F. C. Hasibuan, "Implementasi Framework Flutter Untuk Aplikasi Mobile Pada Pengembangan Air Purifier Berbasis IoT dan Sistem Cerdas," *e-Proceeding Eng.*, vol. 11, no. 6, p. 6765, 2024.

[25] P. E. Wicaksono, "Penentuan Kadar Kandungan Air Pada Biji Kopi Arabika Dengan Teknik Laser-Induced Breakdown Spectroscopy (LIBS)," *Skripsi ITS Fak. Tek. Jur. Tek. Fis.*, p. 90, 2018.

[26] A. Nurfitriyani, M. S. Triyastuti, L. M. Shitophyta, B. R. Wahidi, and I. Mukhaimin, "Perhitungan Kadar Air, Rendemen dan Uji Organoleptik pada Ikan Asin," *Media Teknol. Has. Perikan.*, no. 01, pp. 45–55, 2024, doi: 10.35800/mthp.12.1.2024.53300.

LAMPIRAN

Lampiran 1

Daftar Riwayat Hidup

DAFTAR RIWAYAT HIDUP

1. Data Pribadi

Nama lengkap : Ejy Gustiansyah

Tempat & tanggal lahir : Sungailiat, 04 Agustus 2003

Alamat rumah : Perum.Griya Banin Anshor No.18

Telp : -

Hp : 083800683311

Email : eji.gustiansyah@gmail.com

Jenis kelamin : Laki-laki

Agama : Islam

2. Riwayat Pendidikan

SD Muhammadiyah Sungailiat	2010-2016
SMP Negeri 1 Sungailiat	2016-2019
SMA Negeri 1 Sungailiat	2019-2021

3. Pendidikan Non-Formal

-

Sungailiat, 15 Juli 2025

Ejy Gustiansyah

DAFTAR RIWAYAT HIDUP

1. Data Pribadi

Nama lengkap : Fira Sagita

Tempat & tanggal lahir : Sungailiat, 24 Mei 2004

Alamat rumah : Jl. Cendrawasih I, SD.Harapan No.537 Sungailiat

Telp : -

Hp : 085758260503

Email : firasagita942@gmail.com

Jenis kelamin : Perempuan

Agama : Islam

2. Riwayat Pendidikan

SD Negeri 8 Sungailiat	2011-2017
SMP Negeri 2 Sungailiat	2017-2020
SMA Negeri 1 Sungailiat	2020-2022

3. Pendidikan Non-Formal

-

Sungailiat, 15 Juli 2025

Fira Sagita

Lampiran 2

Code Sistem Alat Pengering

```
#include <WiFi.h>
#include <FirebaseESP32.h>
#include <DHT.h>
#include <Wire.h>
#include <BH1750.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <Time.h>
#include <LiquidCrystal_I2C.h>

// Pin Configuration
#define DHTPIN1 4
#define DHTPIN2 5
#define DHTTYPE DHT22
#define motorIn1 18  //BUKA ATAP
#define motorIn2 19  // TUTUP ATAP
#define heaterPin 27 // pin heater fan, lampu halogen dan fan dc
#define rainPin 35

DHT dht1(DHTPIN1, DHTTYPE);
DHT dht2(DHTPIN2, DHTTYPE);
BH1750 lightMeter;    //Sensor Cahaya
LiquidCrystal_I2C lcd(0x27, 16, 2);   //LCD

// WiFi & Firebase Config
const char* ssid = "Raaa";
const char* password = "**********";
```

```
#define FIREBASE_HOST "monitoring-pengering-lad-93936-default-rtdb.firebaseio.com"

#define FIREBASE_AUTH "3mCFZgcb7aef2YKDcevSBn5zw9R4BrMNtnuM2CnO"

FirebaseData fbdo;

FirebaseConfig config;

FirebaseAuth auth;


// NTP Client

WiFiUDP ntpUDP;

NTPClient timeClient (ntpUDP, "pool.ntp.org", 25200);


// Data Sensor

float suhu1 = 0, suhu2 = 0, suhuRata = 0;

float hum1 = 0, hum2 = 0, humRata = 0;

int rainValue = 0;

float ldrValue = 0;


// Status Sistem

bool atapBuka = false;        // true=TERBUKA atau false=TERTUTUP

bool heaterNyala = false;      // true=NYALA atau false=MATI

bool modeOtomatis;


//last status tersimpan untuk pengambilan data histori

bool lastSavedAtap = true;

bool lastSavedHeater = true;


//last status untuk pengambilan nilai kontrol manual
```

```
bool lastManualAtap = false;
bool lastManualHeater = false;

// Threshold
const int rainThreshold = 3500;
const int suhuThreshold = 25;
const int cahayaThreshold = 50;

// Kontrol Motor
unsigned long waktuGerakMotor = 0;
bool motorSedangGerak = false;
enum MotorArah { NONE, BUKA, TUTUP };
MotorArah arahMotor = NONE;

// Timing
unsigned long lastSensorUpdate = 0;
const unsigned long sensorInterval = 500;
unsigned long lastFirebaseUpdate = 0;
const unsigned long firebaseInterval = 500;
unsigned long lastModeCheck = 0;
const unsigned long modeCheckInterval = 1000;

String waktu;
String tanggal;

void setup() {
  Serial.begin(115200);
```

```
lcd.init(); lcd.backlight();
lcd.setCursor(0, 0); lcd.print("  PROYEK AKHIR  ");
lcd.setCursor(0, 1); lcd.print("   FIRA & EJY   ");

// Inisialisasi Pin
pinMode(motorIn1, OUTPUT);
pinMode(motorIn2, OUTPUT);
pinMode(heaterPin, OUTPUT);
pinMode(rainPin, INPUT);
digitalWrite(motorIn1, LOW);
digitalWrite(motorIn2, LOW);
digitalWrite(heaterPin, LOW);
dht1.begin(); dht2.begin();
Wire.begin(21, 22);
lightMeter.begin();

// Koneksi WiFi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) delay(500);
Serial.println("\nTerhubung ke WiFi");
Serial.print("IP: "); Serial.println(WiFi.localIP());
lcd.clear();
lcd.setCursor(0, 0); lcd.print("WiFi OK");
lcd.setCursor(0, 1); lcd.print(WiFi.localIP());
delay(1000);

// Inisialisasi Firebase
```

```
  config.host = FIREBASE_HOST;
  config.signer.tokens.legacy_token = FIREBASE_AUTH;
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

  // Inisialisasi Waktu
  timeClient.begin();
  while (!timeClient.update()) timeClient.forceUpdate();
  Serial.println("Sistem siap!");

  //Inisialisasi nilai awal dari firebase
  if (Firebase.getBool(fbdo, "/monitoring/control/mode")) {
    modeOtomatis = fbdo.boolData();
  } else {
    modeOtomatis = true; // fallback kalau gagal baca
  }
}

String getFormattedTime() {
  timeClient.update();
  return timeClient.getFormattedTime();
}

String getFormattedDate() {
  timeClient.update();
  time_t epochTime = timeClient.getEpochTime();
  struct tm *ptm = gmtime(&epochTime);
```

```
  char dateString[11];
  snprintf(dateString, sizeof(dateString), "%02d-%02d-%04d",
          ptm->tm_mday, ptm->tm_mon + 1, ptm->tm_year + 1900);
  return String(dateString);
}

void bacaSensor() {
  float suhu1Temp = dht1.readTemperature();
  float hum1Temp = dht1.readHumidity();
  float suhu2Temp = dht2.readTemperature();
  float hum2Temp = dht2.readHumidity();

  // Rata-rata suhu dan kelembapan hanya dari sensor valid
  int countSuhu = 0;
  float totalSuhu = 0;
  if (!isnan(suhu1Temp)) { totalSuhu += suhu1Temp; countSuhu++; }
  if (!isnan(suhu2Temp)) { totalSuhu += suhu2Temp; countSuhu++; }
  suhuRata = countSuhu > 0 ? totalSuhu / countSuhu : 0;

  humRata = 0;
  if (!isnan(hum1Temp) && !isnan(hum2Temp)) {
    humRata = (hum1Temp + hum2Temp) / 2;
  } else if (!isnan(hum1Temp)) {
    humRata = hum1Temp;
  } else if (!isnan(hum2Temp)) {
    humRata = hum2Temp;
  }
```

```
  rainValue = analogRead(rainPin);
  ldrValue = lightMeter.readLightLevel();
  waktu = getFormattedTime();
  tanggal = getFormattedDate();
}

void tampilkanDataLCD() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("S:"); lcd.print(suhuRata, 1);
  lcd.print("C H:"); lcd.print(humRata, 1); lcd.print("%");

  lcd.setCursor(0, 1);
  lcd.print("C:"); lcd.print((int)ldrValue);
  lcd.print("  ");
  lcd.print(rainValue < rainThreshold ? "HUJAN" : "TDKHJN");
  lcd.print(" ");
  lcd.print(modeOtomatis == true ? "A" : "M");
}

void updateFirebaseData() {
// Update data firebase sensor untuk aplikasi Android
  Firebase.setString(fbdo, "/monitoring/tanggal", tanggal);
  Firebase.setString(fbdo, "/monitoring/waktu", waktu);
  Firebase.setFloat(fbdo, "/monitoring/sensor/suhu_rata", suhuRata);
  Firebase.setFloat(fbdo, "/monitoring/sensor/kelembapan_rata", humRata);
```

```
  Firebase.setInt(fbdo, "/monitoring/sensor/hujan", rainValue);
  Firebase.setString(fbdo, "/monitoring/sensor/kondisiHujan", rainValue < rainThreshold ? "YA" : "TIDAK");
  Firebase.setFloat(fbdo, "/monitoring/sensor/cahaya", ldrValue);
  // Update status untuk monitoring
  Firebase.setString(fbdo, "/monitoring/status/atap", atapBuka ? "TERBUKA" : "TERTUTUP");
  Firebase.setString(fbdo, "/monitoring/status/heater", heaterNyala ? "NYALA" : "MATI");
  Firebase.setString(fbdo, "/monitoring/status/mode", modeOtomatis ? "AUTO" : "MANUAL");
}

void saveHistoryData() {      //Format histori data
  FirebaseJson json;
  String path = "histori/" + tanggal + "/" + waktu;
  json.set("suhu_rata", suhuRata);
  json.set("kelembapan_rata", humRata);
  json.set("hujan", rainValue);
  json.set("cahaya", ldrValue);
  json.set("kondisiHujan", rainValue < rainThreshold ? "YA" : "TIDAK");
  json.set("statusAtap", atapBuka ? "TERBUKA" : "TERTUTUP");
  json.set("statusHeater", heaterNyala ? "NYALA" : "MATI");
  json.set("mode", modeOtomatis ? "AUTO" : "MANUAL");

  if (Firebase.setJSON(fbdo, path.c_str(), json)) {
    Serial.println("✓ Data histori berhasil disimpan");
  } else {
```

```
    Serial.println("✗ Gagal simpan histori: " + fbdo.errorReason());
  }
}

void tampilkanDataSerial() {      //Format data serial mmonitor
  Serial.println("\n====== DATA SENSOR ======");
  Serial.printf("Suhu Rata-rata: %.2f °C\n", suhuRata);
  Serial.printf("Kelembapan Rata-rata: %.2f %%\n", humRata);
  Serial.printf("Cahaya: %.2f lux\n", ldrValue);
  Serial.printf("Hujan: %d (ADC)\n", rainValue);
  Serial.println("Kondisi Hujan: " + String(rainValue < rainThreshold ? "YA" :
"TIDAK"));
  Serial.println("\n=== Status Sistem ===");
  Serial.println("Status Atap: " + String(atapBuka ? "TERBUKA" :
"TERTUTUP"));
  Serial.println("Status Heater: " + String(heaterNyala ? "NYALA" : "MATI"));
  Serial.println("Mode: " + String(modeOtomatis ? "AUTO" : "MANUAL"));
  Serial.println("\n=== Waktu ===");
  Serial.println("Tanggal: " + tanggal);
  Serial.println("Waktu: " + waktu);
  Serial.println("========================");
}

void bukaAtap() {
  if (motorSedangGerak || atapBuka == true) return;

  digitalWrite(motorIn1, HIGH);
  digitalWrite(motorIn2, LOW);
```

```
  arahMotor = BUKA;
  motorSedangGerak = true;
  waktuGerakMotor = millis();
  atapBuka = true;
  Serial.println("Membuka atap...");
  // Update Firebase
  Firebase.setBool(fbdo, "/monitoring/status/atap", atapBuka);
}

void tutupAtap() {
  if (motorSedangGerak || atapBuka == false) return;

  digitalWrite(motorIn1, LOW);
  digitalWrite(motorIn2, HIGH);
  arahMotor = TUTUP;
  motorSedangGerak = true;
  waktuGerakMotor = millis();
  atapBuka = false;
  Serial.println("□ Menutup atap...");
  // Update Firebase
  Firebase.setBool(fbdo, "/monitoring/status/atap", atapBuka);
}

void stopMotor() {
  digitalWrite(motorIn1, LOW);
  digitalWrite(motorIn2, LOW);
  arahMotor = NONE;
```

```
  motorSedangGerak = false;
  Serial.println("Motor berhenti");
}

void nyalakanHeater() {
  if (heaterNyala == true) return;

  digitalWrite(heaterPin, HIGH);
  heaterNyala = true;
  Firebase.setBool(fbdo, "/monitoring/status/heater", heaterNyala);
  Serial.println("Heater NYALA");
}

void matikanHeater() {
  if (heaterNyala == false) return;

  digitalWrite(heaterPin, LOW);
  heaterNyala = false;
  Firebase.setBool(fbdo, "/monitoring/status/heater", heaterNyala);
  Serial.println("Heater MATI");
}

void kontrolOtomatis() {
  bool hujan = (rainValue < rainThreshold);
  bool gelapAtauDingin = (ldrValue < cahayaThreshold) || (suhuRata <
suhuThreshold);
  bool atapHarusTertutup = hujan || gelapAtauDingin;
```

```
  if (hujan && atapBuka == true && !motorSedangGerak) {
    Serial.println("□ HUJAN TERDETEKSI - Menutup atap");
    tutupAtap();
  } else if (!hujan && !gelapAtauDingin && atapBuka == false &&
!motorSedangGerak) {
    Serial.println("Kondisi cerah - Membuka atap");
    bukaAtap();
  } else if (!hujan && gelapAtauDingin && atapBuka == true &&
!motorSedangGerak) {
    Serial.println("Gelap/Dingin - Menutup atap");
    tutupAtap();
  }

  // Kontrol heater: nyala hanya jika atap tertutup DAN (hujan ATAU dingin)
  if (atapBuka == false && atapHarusTertutup) {
    nyalakanHeater();
  } else {
    matikanHeater();
  }
}

void kontrolManual() {
  bool manualAtap = lastManualAtap;
  bool manualHeater = lastManualHeater;

  if (Firebase.getBool(fbdo, "/monitoring/control/atap")) {
    manualAtap = fbdo.boolData();
```

```
  }

  if (Firebase.getBool(fbdo, "/monitoring/control/heater")) {
    manualHeater = fbdo.boolData();
  }

  if (manualAtap != lastManualAtap) {
    if (manualAtap) {
      bukaAtap();
    } else {
      tutupAtap();
    }
    lastManualAtap = manualAtap; // simpan status terakhir
  }

  if (manualHeater != lastManualHeater) {
    if (manualHeater) {
      nyalakanHeater();
    } else {
      matikanHeater();
    }
    lastManualHeater = manualHeater;
  }
}


void loop() {
```

```
  timeClient.update();

  // Baca sensor dan update Firebase setiap interval
  if (millis() - lastSensorUpdate >= sensorInterval) {
    lastSensorUpdate = millis();
    bacaSensor();
    updateFirebaseData();
    tampilkanDataLCD();
    tampilkanDataSerial();
  }

  // Simpan histori perubahan atap/heater
  if (atapBuka != lastSavedAtap || heaterNyala != lastSavedHeater) {
    Serial.println("Menyimpan histori karena perubahan status atap atau heater..." +
waktu);
    saveHistoryData();
    lastSavedAtap = atapBuka;
    lastSavedHeater = heaterNyala;
  }

  // Hentikan motor jika sudah selesai bergerak
  if (motorSedangGerak && millis() - waktuGerakMotor >= 30000) {
    stopMotor();
  }

  // Cek mode kontrol dari Firebase
  if (millis() - lastModeCheck >= modeCheckInterval) {
```

```
    lastModeCheck = millis();
    if (Firebase.getBool(fbdo, "/monitoring/control/mode")) {
      modeOtomatis = fbdo.boolData();
      Serial.println("Mode dari Firebase: " + String(modeOtomatis ? "AUTO" :
"MANUAL"));
    } else {
      Serial.println("□ Gagal membaca mode dari Firebase: " +
fbdo.errorReason());
    }
  }

  // Jalankan kontrol otomatis/manual sesuai mode
  if (modeOtomatis) {
    kontrolOtomatis();
  } else {
    kontrolManual();
  }

  delay(100);
}
```

Lampiran 3

Code Pembuatan Aplikasi Android Berbasis Flutter

Code: main.dart

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/foundation.dart';
import 'package:monitoring_app/screen/home.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  if (kIsWeb) {
    await Firebase.initializeApp(
      options: const FirebaseOptions(
        apiKey: "AIzaSyAbDbt4K-PHFEw3zoPSS7S_tD5TK9vtp6s",
        authDomain: "monitoring-pengering-lad-93936.firebaseapp.com",
        databaseURL: "https://monitoring-pengering-lad-93936-default-
rtdb.firebaseio.com",
        projectId: "monitoring-pengering-lad-93936",
        storageBucket: "monitoring-pengering-lad-93936.appspot.com",
        messagingSenderId: "906504301176",
        appId: "1:906504301176:android:1ceccb01256fd09c9f5b8d",
        measurementId: "G-XXXXXXX",
      ),
    );
  } else {
    await Firebase.initializeApp();
  }
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
```

```
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'IoT Monitoring App',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: const HomePage(),
    );
  }
}
```

Code: home.dart

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:monitoring_app/screen/history.dart';

class HomePage extends StatefulWidget {
  const HomePage({super.key});
  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> with TickerProviderStateMixin
{
  final DatabaseReference monitoringRef =
FirebaseDatabase.instance.ref('monitoring');
  final DatabaseReference controlRef =
FirebaseDatabase.instance.ref('monitoring/control');
```

```
Map<dynamic, dynamic>? monitoringData;
bool atapStatus = false;
bool heaterStatus = false;
bool modeOtomatis = false;
late AnimationController _animationController;
late Animation<double> _fadeAnimation;

@override
void initState() {
 super.initState();

 _animationController = AnimationController(
  vsync: this,
  duration: const Duration(milliseconds: 500),
 );
 _fadeAnimation = Tween<double>(begin: 0.0, end: 1.0).animate(
  CurvedAnimation(parent: _animationController, curve: Curves.easeInOut),
 );

 // Listen data monitoring
 monitoringRef.onValue.listen((event) {
  final data = event.snapshot.value as Map<dynamic, dynamic>;
  setState(() {
   monitoringData = data;
  });
  _animationController.forward();
```

```
  });

  // Listen data control (atap, heater, mode)
  controlRef.onValue.listen((event) {
    final data = event.snapshot.value as Map<dynamic, dynamic>;
    setState(() {
      atapStatus = data['atap'] == true;
      heaterStatus = data['heater'] == true;
      modeOtomatis = data['mode'] == true;
    });
  });
}

@override
void dispose() {
  _animationController.dispose();
  super.dispose();
}

// Kirim perubahan ke Firebase tanpa menimpa node control
void updateControl({bool? atap, bool? heater, bool? mode}) {
  final Map<String, dynamic> updateData = {};
  if (atap != null) updateData['atap'] = atap;
  if (heater != null) updateData['heater'] = heater;
  if (mode != null) updateData['mode'] = mode;
  controlRef.update(updateData);
}
```

```
@override
Widget build(BuildContext context) {
  final sensor = monitoringData?['sensor'] ?? {};
  final status = monitoringData?['status'] ?? {};
  final tanggal = monitoringData?['tanggal'] ?? '-';
  final waktu = monitoringData?['waktu'] ?? '-';

  return Scaffold(
    backgroundColor: const Color.fromARGB(255, 255, 255, 255),
    body: monitoringData == null
        ? _buildLoadingScreen()
        : CustomScrollView(
            slivers: [
              _buildAppBar(tanggal, waktu),
              SliverToBoxAdapter(
                child: FadeTransition(
                  opacity: _fadeAnimation,
                  child: Padding(
                    padding: const EdgeInsets.all(16.0),
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        _buildStatusCard(),
                        const SizedBox(height: 20),
                        _buildSensorSection(sensor),
                        const SizedBox(height: 20),
```

```
                    _buildControlSection(status),
                  ],
                ),
              ),
            ),
          ),
        ),
      );
    }
    Widget _buildLoadingScreen() {
      return Container(
        decoration: BoxDecoration(
          gradient: LinearGradient(
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
            colors: [
              const Color.fromARGB(255, 133, 189, 253),
              const Color.fromARGB(255, 44, 89, 211),
            ],
          ),
        ),
        child: const Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              CircularProgressIndicator(
```

```
            valueColor: AlwaysStoppedAnimation<Color>(Colors.white),
            strokeWidth: 3,
          ),
          SizedBox(height: 20),
          Text(
            'Memuat data monitoring...',
            style: TextStyle(
              color: Colors.white,
              fontSize: 16,
              fontWeight: FontWeight.w500,
            ),
          ),
        ],
      ),
    ),
  );
}

Widget _buildAppBar(String tanggal, String waktu) {
return SliverAppBar(
  expandedHeight: 200,
  floating: false,
  pinned: true,
  backgroundColor: Colors.blue.shade600,
  actions: [
    IconButton(
      icon: const Icon(Icons.history_rounded, color: Colors.white),
```

```
    onPressed: () {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: const Text("Membuka histori..."),
          backgroundColor: Colors.yellow.shade900,
          behavior: SnackBarBehavior.floating,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(10),
          ),
        ),
      );
      Future.delayed(const Duration(milliseconds: 300), () {
        Navigator.push(
          context,
          MaterialPageRoute(builder: (_) => const HistoriPage()),
        );
      });
    },
  ),
],
flexibleSpace: FlexibleSpaceBar(
  title: const Text(
    'Monitoring Lada',
    style: TextStyle(
      color: Colors.white,
      fontWeight: FontWeight.bold,
    ),
```

```
),
background: Container(
  decoration: BoxDecoration(
    gradient: LinearGradient(
      begin: Alignment.topCenter,
      end: Alignment.bottomCenter,
      colors: [
        Colors.blue.shade900,
       Colors.blue.shade100,
      ],
    ),
  ),
  child: Positioned(
    top: 80,
    left: 20,
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          tanggal,
          style: const TextStyle(
            color: Colors.white70,
            fontSize: 14,
          ),
        ),
        Text(
          waktu,
```

```
                style: const TextStyle(
                  color: Colors.white,
                  fontSize: 18,
                  fontWeight: FontWeight.w600,
                ),
              ),
            ],
          ),
        ),
      ),
    ),
  );
}

 Widget _buildStatusCard() {
   return Container(
     width: double.infinity,
     padding: const EdgeInsets.all(20),
     decoration: BoxDecoration(
       gradient: LinearGradient(
         colors: [Colors.yellow.shade900, Colors.yellow.shade600],
         begin: Alignment.topLeft,
         end: Alignment.bottomRight,
       ),
       borderRadius: BorderRadius.circular(20),
       boxShadow: [
         BoxShadow(
```

```
      color: const Color.fromARGB(255, 105, 105, 105).withOpacity(0.3),
      blurRadius: 15,
      offset: const Offset(0, 5),
    ),
  ],
),
child: Row(
  children: [
    Container(
      padding: const EdgeInsets.all(12),
      decoration: BoxDecoration(
        color: Colors.white.withOpacity(0.2),
        borderRadius: BorderRadius.circular(15),
      ),
      child: Icon(
        modeOtomatis ? Icons.auto_mode : Icons.touch_app,
        color: Colors.white,
        size: 28,
      ),
    ),
    const SizedBox(width: 16),
    Expanded(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            'Mode Sistem',
```

```
          style: TextStyle(
            color: Colors.white.withOpacity(0.9),
            fontSize: 14,
          ),
        ),
        const SizedBox(height: 4),
        Text(
          modeOtomatis ? 'Otomatis' : 'Manual',
          style: const TextStyle(
            color: Colors.white,
            fontSize: 20,
            fontWeight: FontWeight.bold,
          ),
        ),
      ],
    ),
  ),
  Container(
    padding: const EdgeInsets.symmetric(horizontal: 12, vertical: 6),
    decoration: BoxDecoration(
      color: Colors.white.withOpacity(0.2),
      borderRadius: BorderRadius.circular(20),
    ),
    child: Row(
      mainAxisSize: MainAxisSize.min,
      children: [
        Container(
```

```
                width: 8,
                height: 8,
                decoration: BoxDecoration(
                  color: const Color.fromARGB(255, 50, 235, 59),
                  shape: BoxShape.circle,
                ),
              ),
              const SizedBox(width: 6),
              const Text(
                'Aktif',
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 12,
                  fontWeight: FontWeight.w600,
                ),
              ),
            ],
          ),
        ),
      ],
    ),
  );
}

Widget _buildSensorSection(Map sensor) {
  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
```

```
      children: [
        const Text(
          'Data Sensor',
          style: TextStyle(
            fontSize: 20,
            fontWeight: FontWeight.bold,
            color: Colors.black87,
          ),
        ),
        const SizedBox(height: 16),
        GridView.count(
          crossAxisCount: 2,
          shrinkWrap: true,
          physics: const NeverScrollableScrollPhysics(),
          crossAxisSpacing: 16,
          mainAxisSpacing: 16,
          childAspectRatio: 1.1,
          children: [
            _buildSensorCard('Cahaya', sensor['cahaya'], Icons.wb_sunny,
Colors.orange),
            _buildSensorCard('Hujan', sensor['hujan'], Icons.water_drop, Colors.blue),
            _buildSensorCard('Suhu', sensor['suhu_rata'], Icons.thermostat,
Colors.red),
            _buildSensorCard('Kelembapan', sensor['kelembapan_rata'], Icons.opacity,
Colors.teal),
          ],
        ),
        const SizedBox(height: 16),
```

```
      _buildWeatherCard(sensor['kondisiHujan']),
    ],
  );
 }


 Widget _buildSensorCard(String title, dynamic value, IconData icon, Color
color) {
  return Container(
    padding: const EdgeInsets.all(16),
    decoration: BoxDecoration(
      color: const Color.fromARGB(255, 243, 243, 243),
      borderRadius: BorderRadius.circular(16),
      boxShadow: [
        BoxShadow(
          color: Colors.grey.withOpacity(0.1),
          blurRadius: 10,
          offset: const Offset(0, 3),
        ),
      ],
    ),
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Container(
          padding: const EdgeInsets.all(12),
          decoration: BoxDecoration(
            color: color.withOpacity(0.1),
```

```
      borderRadius: BorderRadius.circular(12),
    ),
    child: Icon(
      icon,
      color: color,
      size: 24,
    ),
  ),
  const SizedBox(height: 12),
  Text(
    title,
    style: TextStyle(
      fontSize: 12,
      color: Colors.grey[600],
      fontWeight: FontWeight.w500,
    ),
  ),
  const SizedBox(height: 4),
  Text(
    value.toString(),
    style: TextStyle(
      fontSize: 16,
      fontWeight: FontWeight.bold,
      color: Colors.grey[800],
    ),
  ),
],
```

```
    ),
  );
}

Widget _buildWeatherCard(dynamic kondisiHujan) {
  final isRaining = kondisiHujan == 'Hujan';
  return Container(
    width: double.infinity,
    padding: const EdgeInsets.all(16),
    decoration: BoxDecoration(
      color: const Color.fromARGB(255, 243, 243, 243),
      borderRadius: BorderRadius.circular(16),
      boxShadow: [
        BoxShadow(
          color: Colors.grey.withOpacity(0.1),
          blurRadius: 10,
          offset: const Offset(0, 3),
        ),
      ],
    ),
    child: Row(
      children: [
        Container(
          padding: const EdgeInsets.all(12),
          decoration: BoxDecoration(
            color: (isRaining ? Colors.blue : const Color.fromARGB(255, 144, 179,
245)).withOpacity(0.1),
```

```
    borderRadius: BorderRadius.circular(12),
  ),
  child: Icon(
    isRaining ? Icons.beach_access : Icons.beach_access,
    color: isRaining ? Colors.blue : Colors.blue,
    size: 24,
  ),
),
const SizedBox(width: 16),
Expanded(
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Text(
        'Kondisi Hujan  ',
        style: TextStyle(
          fontSize: 12,
          color: Colors.grey[600],
          fontWeight: FontWeight.w500,
        ),
      ),
      const SizedBox(height: 4),
      Text(
        kondisiHujan.toString(),
        style: TextStyle(
          fontSize: 16,
          fontWeight: FontWeight.bold,
```

```
                color: Colors.grey[800],
              ),
            ),
          ],
        ),
      ),
    ],
  ),
 );
}

Widget _buildControlSection(Map status) {
 return Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
   const Text(
    'Kontrol Sistem',
    style: TextStyle(
     fontSize: 20,
     fontWeight: FontWeight.bold,
     color: Colors.black87,
    ),
   ),
   const SizedBox(height: 16),
   _buildDeviceStatusCards(),
   const SizedBox(height: 20),
   _buildControlButtons(),
```

```
      const SizedBox(height: 20),
      _buildModeToggle(),
    ],
  );
}

Widget _buildDeviceStatusCards() {
  return Row(
    children: [
      Expanded(
        child: _buildDeviceCard(
          'Atap',
          atapStatus ? 'Terbuka' : 'Tertutup',
          atapStatus ? Icons.roofing : Icons.roofing_outlined,
          atapStatus ? Colors.green : Colors.grey,
        ),
      ),
      const SizedBox(width: 16),
      Expanded(
        child: _buildDeviceCard(
          'Heater',
          heaterStatus ? 'Aktif' : 'Mati',
          heaterStatus ? Icons.whatshot : Icons.ac_unit,
          heaterStatus ? Colors.orange : Colors.grey,
        ),
      ),
    ],
```

```
    );
  }

  Widget _buildDeviceCard(String title, String status, IconData icon, Color color)
{
    return Container(
      padding: const EdgeInsets.all(16),
      decoration: BoxDecoration(
        color: const Color.fromARGB(255, 243, 243, 243),
        borderRadius: BorderRadius.circular(16),
        boxShadow: [
          BoxShadow(
            color: Colors.grey.withOpacity(0.1),
            blurRadius: 10,
            offset: const Offset(0, 3),
          ),
        ],
      ),
      child: Column(
        children: [
          Icon(
            icon,
            color: color,
            size: 32,
          ),
          const SizedBox(height: 8),
          Text(
```

```
          title,
          style: TextStyle(
            fontSize: 12,
            color: Colors.grey[600],
            fontWeight: FontWeight.w500,
          ),
        ),
        const SizedBox(height: 4),
        Text(
          status,
          style: TextStyle(
            fontSize: 14,
            fontWeight: FontWeight.bold,
            color: color,
          ),
        ),
      ],
    ),
  );
}

Widget _buildControlButtons() {
  return Column(
    children: [
      Row(
        children: [
          Expanded(
```

```
              child: _buildControlButton(
                label: atapStatus ? 'Tutup Atap' : 'Buka Atap',
                icon: atapStatus ? Icons.keyboard_arrow_down :
Icons.keyboard_arrow_up,
                color: Colors.green,
                onPressed: modeOtomatis ? null : () {
                  updateControl(atap: !atapStatus);
                  _showSnackBar(atapStatus ? 'Menutup atap...' : 'Membuka atap...');
                },
              ),
            ),
            const SizedBox(width: 16),
            Expanded(
              child: _buildControlButton(
                label: heaterStatus ? 'Matikan Heater' : 'Nyalakan Heater',
                icon: heaterStatus ? Icons.power_off : Icons.power,
                color: Colors.orange,
                onPressed: modeOtomatis ? null : () {
                  updateControl(heater: !heaterStatus);
                  _showSnackBar(heaterStatus ? 'Mematikan heater...' : 'Menyalakan
heater...');
                },
              ),
            ),
          ],
        ),
      ],
    );
```

```
}

Widget _buildControlButton({
  required String label,
  required IconData icon,
  required Color color,
  required VoidCallback? onPressed,
}) {
  return Container(
    height: 60,
    child: ElevatedButton(
      onPressed: onPressed,
      style: ElevatedButton.styleFrom(
        backgroundColor: onPressed == null ? Colors.grey[300] : color,
        foregroundColor: Colors.white,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(16),
        ),
        elevation: onPressed == null ? 0 : 3,
      ),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(icon, size: 18),
          const SizedBox(width: 8),
          Flexible(
            child: Text(
```

```
            label,
            style: const TextStyle(
              fontSize: 12,
              fontWeight: FontWeight.w600,
            ),
            textAlign: TextAlign.center,
          ),
        ),
      ],
    ),
  ),
 );
}

Widget _buildModeToggle() {
  return Container(
    padding: const EdgeInsets.all(4),
    decoration: BoxDecoration(
      color: Colors.grey[200],
      borderRadius: BorderRadius.circular(16),
    ),
    child: Row(
      children: [
        Expanded(
          child: _buildModeButton(
            'Manual',
            !modeOtomatis,
```

```
            () {
              updateControl(mode: false);
              _showSnackBar('Beralih ke mode Manual');
            },
          ),
        ),
        Expanded(
          child: _buildModeButton(
            'Otomatis',
            modeOtomatis,
            () {
              updateControl(mode: true);
              _showSnackBar('Beralih ke mode Otomatis');
            },
          ),
        ),
      ],
    ),
  );
 }

  Widget _buildModeButton(String label, bool isActive, VoidCallback onPressed)
{
    return GestureDetector(
      onTap: isActive ? null : onPressed,
      child: Container(
        padding: const EdgeInsets.symmetric(vertical: 16),
```

```
      decoration: BoxDecoration(
        color: isActive ? Colors.blue.shade600 : Colors.transparent,
        borderRadius: BorderRadius.circular(12),
      ),
      child: Text(
        label,
        textAlign: TextAlign.center,
        style: TextStyle(
          color: isActive ? Colors.white : Colors.grey[600],
          fontWeight: FontWeight.w600,
        ),
      ),
    ),
  );
}

void _showSnackBar(String message) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(message),
      duration: const Duration(seconds: 1),
      backgroundColor: Colors.yellow.shade900,
      behavior: SnackBarBehavior.floating,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(10),
      ),
    ),
```

```
    );
  }
}
```

Code: history.dart

```
import 'package:flutter/material.dart';
import 'package:firebase_database/firebase_database.dart';

class HistoriPage extends StatefulWidget {
  const HistoriPage({super.key});

  @override
  State<HistoriPage> createState() => _HistoriPageState();
}

class _HistoriPageState extends State<HistoriPage> {
  final DatabaseReference historiRef = FirebaseDatabase.instance.ref('histori');
  String? selectedTanggal;
  String? selectedWaktu;

  Map<String, List<Map<String, dynamic>>> groupedLogs = {};
  List<String> allTanggal = [];
  List<String> waktuUntukTanggal = [];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[50], // Background putih lembut
      appBar: AppBar(
        title: const Text(
          'Log Histori',
          style: TextStyle(
            color: Colors.white,
            fontWeight: FontWeight.w600,
            fontSize: 20,
          ),
        ),
        backgroundColor: Colors.blue[700], // App bar biru
```

```
  elevation: 0,
  centerTitle: true,
  iconTheme: const IconThemeData(color: Colors.white),
 ),
 body: StreamBuilder<DatabaseEvent>(
  stream: historiRef.onValue,
  builder: (context, snapshot) {
   if (snapshot.hasError) {
    return Center(
     child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
       Icon(
        Icons.error_outline,
        size: 64,
        color: Colors.red[300],
       ),
       const SizedBox(height: 16),
       const Text(
        "Terjadi kesalahan",
        style: TextStyle(
         fontSize: 16,
         color: Colors.black54,
        ),
       ),
      ],
     ),
    );
   }

   if (!snapshot.hasData || snapshot.data!.snapshot.value == null) {
    return Center(
     child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
       Icon(
        Icons.history,
        size: 64,
        color: Colors.grey[400],
       ),
```

```
        const SizedBox(height: 16),
        const Text(
          "Belum ada histori",
          style: TextStyle(
            fontSize: 16,
            color: Colors.black54,
          ),
        ),
      ],
    ),
  );
}

final rawData = snapshot.data!.snapshot.value as Map<dynamic, dynamic>;
groupedLogs.clear();
allTanggal.clear();

// Group data berdasarkan tanggal dan isi waktu
rawData.forEach((tanggalKey, waktuMap) {
  if (waktuMap is Map) {
    final List<Map<String, dynamic>> logList = [];

    waktuMap.forEach((waktuKey, detail) {
      if (detail is Map) {
        final log = Map<String, dynamic>.from(detail);
        log['tanggal'] = tanggalKey;
        log['waktu'] = waktuKey;
        logList.add(log);
      }
    });

    if (logList.isNotEmpty) {
      groupedLogs[tanggalKey.toString()] = logList;
      allTanggal.add(tanggalKey.toString());
    }
  }
});

allTanggal.sort((a, b) => b.compareTo(a)); // Urutkan tanggal descending
```

```
if (selectedTanggal != null && groupedLogs[selectedTanggal] != null) {
  waktuUntukTanggal = groupedLogs[selectedTanggal]!
      .map((log) => log['waktu'].toString())
      .toList();
  waktuUntukTanggal.sort((a, b) => b.compareTo(a));
} else {
  waktuUntukTanggal = [];
}

// Filter logs sesuai tanggal dan jam
final logs = <Map<String, dynamic>>[];
if (selectedTanggal == null) {
  // semua tanggal
  for (var tanggal in groupedLogs.keys) {
    logs.addAll(groupedLogs[tanggal]!);
  }
} else {
  if (groupedLogs[selectedTanggal] != null) {
    logs.addAll(groupedLogs[selectedTanggal]!);
  }
}

// Jika ada filter jam, saring lagi
final filteredLogs = selectedWaktu != null
    ? logs.where((log) => log['waktu'] == selectedWaktu).toList()
    : logs;

// Urutkan berdasarkan waktu terbaru
filteredLogs.sort((a, b) {
  final aKey = '${a['tanggal']} ${a['waktu']}';
  final bKey = '${b['tanggal']} ${b['waktu']}';
  return bKey.compareTo(aKey); // descending
});

return Column(
  children: [
    // Header dengan gradient
    Container(
      width: double.infinity,
      decoration: BoxDecoration(
```

```
  gradient: LinearGradient(
    colors: [Colors.blue[700]!, Colors.blue[500]!],
    begin: Alignment.topCenter,
    end: Alignment.bottomCenter,
  ),
),
child: Padding(
  padding: const EdgeInsets.fromLTRB(16, 0, 16, 20),
  child: Column(
    children: [
      // Filter Tanggal
      Container(
        width: double.infinity,
        decoration: BoxDecoration(
          gradient: LinearGradient(
            colors: [Colors.blue[700]!, Colors.blue[500]!],
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
          ),
        ),
        child: Padding(
          padding: const EdgeInsets.fromLTRB(16, 16, 16, 24),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              const Text(
                'Pilih Tanggal:',
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 16,
                  fontWeight: FontWeight.w600,
                ),
              ),
              const SizedBox(height: 8),
              Container(
                decoration: BoxDecoration(
                  color: Colors.white,
                  borderRadius: BorderRadius.circular(12),
                  boxShadow: [
                    BoxShadow(
```

```
                color: Colors.black.withOpacity(0.1),
                blurRadius: 8,
                offset: const Offset(0, 2),
              ),
            ],
          ),
          child: DropdownButtonFormField<String>(
            value: selectedTanggal,
            isExpanded: true,
            decoration: InputDecoration(
              hintText: 'Semua Tanggal',
              // labelText: 'Filter berdasarkan tanggal',
              labelStyle: TextStyle(color: Colors.grey[600]),
              border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(12),
                borderSide: BorderSide.none,
              ),
              filled: true,
              fillColor: Colors.white,
              prefixIcon: Icon(Icons.calendar_today, color:
Colors.blue[600]),
              contentPadding: const EdgeInsets.symmetric(horizontal:
16, vertical: 12),
            ),
            items: [
              const DropdownMenuItem(
                value: null,
                child: Text('Semua Tanggal'),
              ),
              ...allTanggal.map((tanggal) => DropdownMenuItem(
                    value: tanggal,
                    child: Text(tanggal),
                  )),
            ],
            onChanged: (value) {
              setState(() {
                selectedTanggal = value;
                selectedWaktu = null; // reset jam jika tanggal berubah
              });
            },
```

```
            ),
          ),
        ],
      ),
    ),
  ),


  // Filter Waktu (jika tanggal dipilih)
  if (selectedTanggal != null) ...[
    const SizedBox(height: 12),
    Container(
      decoration: BoxDecoration(
        color: Colors.white,
        borderRadius: BorderRadius.circular(12),
        boxShadow: [
          BoxShadow(
            color: Colors.black.withOpacity(0.1),
            blurRadius: 8,
            offset: const Offset(0, 2),
          ),
        ],
      ),
      child: DropdownButtonFormField<String>(
        value: selectedWaktu,
        isExpanded: true,
        decoration: InputDecoration(
          // labelText: 'Filter berdasarkan waktu',
          labelStyle: TextStyle(color: Colors.grey[600]),
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
            borderSide: BorderSide.none,
          ),
          filled: true,
          fillColor: Colors.white,
          prefixIcon: Icon(Icons.access_time, color: Colors.blue[600]),
          contentPadding: const EdgeInsets.symmetric(horizontal: 16,
vertical: 12),
        ),
        items: [
```

```
                const DropdownMenuItem(
                  value: null,
                  child: Text('Semua Jam'),
                ),
                ...waktuUntukTanggal.map((waktu) => DropdownMenuItem(
                      value: waktu,
                      child: Text(waktu),
                    )),
              ],
              onChanged: (value) {
                setState(() {
                  selectedWaktu = value;
                });
              },
            ),
          ),
        ],
      ),
    ),
  ),
),
// Content Area
Expanded(
  child: filteredLogs.isEmpty
      ? Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Icon(
                Icons.search_off,
                size: 64,
                color: Colors.grey[400],
              ),
              const SizedBox(height: 16),
              const Text(
                "Tidak ada data untuk filter ini",
                style: TextStyle(
                  fontSize: 16,
                  color: Colors.black54,
                ),
```

```
        ),
      ],
    ),
  )
: ListView.builder(
    padding: const EdgeInsets.all(16),
    itemCount: filteredLogs.length,
    itemBuilder: (context, index) {
      final log = filteredLogs[index];
      return Container(
        margin: const EdgeInsets.only(bottom: 12),
        decoration: BoxDecoration(
          color: Colors.grey[100], // Card abu muda
          borderRadius: BorderRadius.circular(16),
          boxShadow: [
            BoxShadow(
              color: Colors.black.withOpacity(0.08),
              blurRadius: 8,
              offset: const Offset(0, 2),
            ),
          ],
        ),
        child: Padding(
          padding: const EdgeInsets.all(16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header Card
              Row(
                children: [
                  Icon(
                    Icons.access_time,
                    color: Colors.blue[600],
                    size: 20,
                  ),
                  const SizedBox(width: 8),
                  Text(
                    "${log['tanggal']} - ${log['waktu']}",
                    style: TextStyle(
                      fontSize: 16,
```

```
                                fontWeight: FontWeight.bold,
                                color: Colors.blue[700],
                              ),
                            ),
                          ],
                        ),
                        const SizedBox(height: 12),

                        // Data Grid
                        _buildDataGrid(log),
                      ],
                    ),
                  ),
                );
              },
            ),
          ),
        ],
      );
    },
  ),
 );
}
Widget _buildDataGrid(Map<String, dynamic> log) {
 final dataItems = [
  {'icon': Icons.wb_sunny, 'label': 'Cahaya', 'value': log['cahaya'] ?? '-'},
  {'icon': Icons.water_drop, 'label': 'Hujan', 'value': log['hujan'] ?? '-'},
  {'icon': Icons.thermostat, 'label': 'Suhu', 'value': log['suhu_rata'] ?? '-'},
  {'icon': Icons.opacity, 'label': 'Kelembapan', 'value': log['kelembapan_rata'] ??
'-'},
  {'icon': Icons.cloud, 'label': 'Kondisi Hujan', 'value': log['kondisiHujan'] ?? '-'},
  {'icon': Icons.roofing, 'label': 'Status Atap', 'value': log['statusAtap'] ?? '-'},
  {'icon': Icons.heat_pump, 'label': 'Status Heater', 'value': log['statusHeater'] ??
'-'},
  {'icon': Icons.settings, 'label': 'Mode', 'value': log['mode'] ?? '-'},
 ];

 return Column(
  children: dataItems.map((item) {
   return Container(
```

```
        margin: const EdgeInsets.only(bottom: 8),
        padding: const EdgeInsets.symmetric(horizontal: 12, vertical: 8),
        decoration: BoxDecoration(
          color: Colors.white,
          borderRadius: BorderRadius.circular(8),
          border: Border.all(color: Colors.grey[200]!),
        ),
        child: Row(
          children: [
            Icon(
              item['icon'] as IconData,
              size: 18,
              color: Colors.grey[600],
            ),
            const SizedBox(width: 12),
            Expanded(
              child: Text(
                item['label'] as String,
                style: TextStyle(
                  fontSize: 14,
                  color: Colors.grey[700],
                  fontWeight: FontWeight.w500,
                ),
              ),
            ),
            Text(
              item['value'].toString(),
              style: const TextStyle(
                fontSize: 14,
                fontWeight: FontWeight.w600,
                color: Colors.black87,
              ),
            ),
          ],
        ),
      );
    }).toList(),
  );
 }
}
```