Pemantauan Oven Despatch Secara Real-Time Berbasis Web

Tugas Akhir

Oleh:
Mochammad Nuruddin Mustaqim (4212111003)
Zidan Gunawan Kusumah (4212111028)

Program Studi Teknik Mekatronika
Jurusan Teknik Elektro
Politeknik Negeri Batam
2024

Pernyataan Keaslian Tugas Akhir

Saya yang bertandatangan dibawah ini menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya yang berjudul : "Pemantauan *Oven Despatch* Secara *Real-Time* Berbasis Web" adalah **hasil karya sendiri**, **diselesaikan tanpa menggunakan bahan- bahan yang tidak diizinkan**, dan **bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.** Semua referensi yang dikutip atau dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan saya ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Batam, 21 Januari 2024

Mochammad Nuruddin Mustaqim
NIM: 4212111003

Zidan Gunawan Kusumah
NIM: 4212111028

Lembar Pengesahan

Tugas Akhir disusun untuk memenuhi salah satu syarat memperoleh gelar
Sarjana Terapan Teknik (S.Tr.T)
di
Politeknik Negeri Batam

Oleh:
Zidan Gunawan Kusumah (4212111028)
Mochammad Nuruddin Mustaqim (4212111003)

Tanggal Sidang: 21 Januari, 2025

Disetujui oleh :

1. **Handri Toar, S.ST., M.Tr.T.**
   NIK: 198408292021211002

1. **Abdurahman Dwijotomo S.ST., M.Sc.**
   NIK: 122257

2. **Ir Indra Hardian Mulyadi S.T., M.Eng., Ph.D**
   NIK: 117179

Pemantauan *Oven Despatch* Secara *Real-Time* Berbasis Web

Abstrak

Saat ini, pemantauan suhu pada oven industri sering dilakukan secara manual, yang rentan terhadap kesalahan dan tidak memungkinkan monitoring *real-time* dari jarak jauh. Untuk mengatasi masalah ini, dikembangkan sistem pemantauan suhu oven Despatch berbasis IoT yang terintegrasi dengan *database cloud*. Sistem ini bertujuan untuk memberikan solusi *monitoring real-time* terhadap suhu dan status pintu oven, serta mempermudah *user* dalam pengambilan keputusan berbasis data. Sistem ini menggunakan mikrokontroler NodeMCU ESP32, sensor suhu MAX6675, dan sensor inframerah, dan dapat menyimpan data menggunakan platform *Firebase Realtime Database*. Hasil pengujian menunjukkan bahwa sistem mampu memantau suhu dengan akurasi ±1°C setelah proses kalibrasi dan menyimpan data pengukuran dalam format PDF untuk dokumentasi dan analisis. Metode yang digunakan termasuk perancangan perangkat keras, perangkat lunak, dan mekanikal, serta kalibrasi termokopel untuk memastikan akurasi pengukuran suhu. Dengan menggunakan React.js, antarmuka pengguna berbasis web yang responsif membuat pengelolaan dan akses data lebih mudah. Selain itu, jika suhu melebihi batas, sistem akan mengirimkan notifikasi otomatis. Peneliti berhasil mengembangkan sistem pemantauan berbasis *web* yang sepenuhnya fungsional, mampu menampilkan data suhu dan status pintu oven secara *real-time*, sehingga menjadi langkah penting dalam modernisasi proses industri dan berkontribusi signifikan terhadap peningkatan efisiensi operasional.

Kata kunci: *Web*, MAX6675, NodeMCU, Oven, Firebase

Oven Despatch Web-based Real-Time Monitoring
Abstract

Currently, temperature monitoring in industrial ovens is often done manually, which is prone to errors and does not allow real-time monitoring remotely. To solve this problem, an IoT-based Despatch oven temperature monitoring system integrated with a cloud database was developed. This system aims to provide a solution for real-time monitoring of oven door temperature and status, and facilitate users in making data-based decisions. This system uses NodeMCU ESP32 microcontroller, MAX6675 temperature sensor, and infrared sensor, and can store data using Firebase Realtime Database platform. The test results show that the system is able to monitor the temperature with an accuracy of ±1°C after the calibration process and save the measurement data in PDF format for documentation and analysis. The methods used include hardware, software, and mechanical design, as well as thermocouple calibration to ensure temperature measurement accuracy. Using React.js, the responsive web-based user interface makes data management and access easier. In addition, if the temperature exceeds the limit, the system will send an automatic notification. The researchers successfully developed a fully functional web-based monitoring system, capable of displaying temperature data and oven door status in real-time, thus becoming an important step in the modernization of industrial processes and contributing significantly to the improvement of operational efficiency.

Keywords: Web, MAX6675, NodeMCU, Oven, Firebase

Kata Pengantar

Dengan penuh rasa syukur kepada Allah Subhanahu Wa Ta'ala, kami mengucapkan alhamdulillahirabbil'alamin atas rahmat, hidayah, dan karunia-Nya, yang memungkinkan kami untuk menyelesaikan tugas akhir yang berjudul "Pemantauan *Oven Despatch* Secara *Real-Time* Berbasis Web". Tugas akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Terapan Teknik (S.Tr.T) pada program studi Teknik Mekatronika di Jurusan Teknik Elektro, Politeknik Negeri Batam.

Kami menyadari bahwa laporan ini masih memiliki banyak kekurangan dan jauh dari kata sempurna, disebabkan oleh keterbatasan kemampuan kami sebagai penulis. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun untuk perbaikan di masa mendatang.

Dalam proses penulisan laporan tugas akhir ini, kami senantiasa mendapatkan bimbingan, dorongan, dan semangat dari berbagai pihak. Dengan demikian, kami ingin menyampaikan rasa terima kasih yang tulus kepada pembimbing yang terhormat, Yth. Bapak Handri Toar, S.ST., M.Tr.T., selaku Dosen Pembimbing. Beliau telah dengan sabar meluangkan waktu, tenaga, dan pemikirannya untuk membimbing penulis dalam menyusun laporan ini. Kami juga ingin menyampaikan rasa terima kasih yang mendalam kepada:

1. Ayah dan Ibu kami yang selalu supportif terhadap pendidikan anaknya.
2. Yth. Ir. Bambang Hendrawan, S.T., M.S.M. selaku Direktur Politeknik Negeri Batam.
3. Yth. Bapak Dr.Ir. Budi Sugandi, S.T., M.Eng., IPM selaku Kepala Jurusan Teknik Elektro Politeknik Negeri Batam.
4. Yth. Bapak Ir. Indra Hardian Mulyadi, S.T., M.Eng., Ph.D selaku Ketua Program Studi Teknik Mekatronika Politeknik Negeri Batam.
5. Yth. Bapak Diono S.Tr. T., M.Sc selaku Wali Dosen Mekatronika kelas malam khusus karyawan angkatan 2021 di Politeknik Negeri Batam.
6. Seluruh Dosen dan Staf karyawan Teknik Elektro di Politeknik Negeri Batam yang telah sabar membimbing dan mengajarkan ilmu pengetahuan selama masa perkuliahan.
7. Teman-teman seperjuangan Teknik Mekatronika kelas malam angkatan 2021 yang saling mendukung dan membantu satu sama lain.
8. Dalam perjalanan ini, kami ingin menyampaikan terima kasih yang tulus kepada pasangan kami, Naomi Wulandari dan Riska Kusumawati, yang selalu mendukung dan memberikan semangat.
9. Teman-teman di luar bangku perkuliahan serta rekan-rekan kerja yang selalu mendukung, menyemangati, membantu dan bisa mengerti keadaan penulis.

10. Seluruh mahasiswa dan alumni Teknik Elektro Program Studi Mekatronika Politeknik Negeri Batam serta seluruh pihak yang tidak dapat disebutkan satu-persatu.

Batam, 2 Januari 2024

Mochammad Nuruddin Mustaqim	Zidan Gunawan Kusumah
NIM: 4212111003	NIM: 4212111028

Daftar Isi

Daftar Gambar

Daftar Tabel

Bab 1. Pendahuluan

1.1. Latar Belakang

Teknologi *Internet of Things* (IoT) adalah perubahan paradigma baru yang memungkinkan teknologi untuk menghubungkan apa pun dengan siapa pun, kapan pun, dan di mana pun. Ini memiliki kemampuan untuk sepenuhnya mengubah sektor manufaktur [1]. Di zaman modern, internet telah menjadi salah satu kebutuhan manusia utama, bersama dengan makanan, air, pakaian, dan tempat berlindung. Meskipun internet baru berumur beberapa dekade, IoT telah banyak diterapkan di banyak bidang. Salah satunya adalah IoT, di mana benda-benda memiliki kemampuan untuk melakukan hal-hal tertentu dan dapat berkomunikasi dengan satu sama lain melalui jaringan internet. Dengan mengaktifkan objek-objek ini untuk "berkomunikasi", mereka dapat mengumpulkan data penting melalui sensor. Data ini kemudian dapat dikirim ke sistem *backend* untuk dianalisis dan dilakukan tindakan tambahan [1].

Dalam penelitian yang dilakukan oleh YP Kondratenko dan kawan-kawan, dijelaskan bagaimana pendekatan IoT dapat diimplementasikan untuk otomatisasi sistem industri kompleks. Penggunaan teknologi ini memungkinkan *monitoring* dan analisis data secara *real-time*, sehingga meningkatkan kemampuan prediktif dan respon terhadap kondisi operasional yang berubah-ubah [2].

Selain itu, C Fadhilah dan A Zubaidi dalam desain sistem deteksi kebakaran dini pada oven tembakau menunjukkan bahwa penggunaan IoT dalam sistem pemantauan dapat meningkatkan keselamatan dan efisiensi operasional. Sistem ini memungkinkan deteksi dini dan respon cepat terhadap kondisi abnormal, yang sangat penting dalam lingkungan produksi yang berisiko tinggi seperti oven industri. Dengan memanfaatkan IoT, sistem pemantauan dapat mengirimkan data *real-time* kepada pengelola untuk segera mengambil tindakan preventif atau korektif [3].

Dalam sistem monitoring oven saat ini terletak pada beberapa aspek krusial yang mempengaruhi efisiensi dan efektivitas operasional. Pertama, data kinerja oven tidak terorganisir dengan baik, karena masih mengandalkan sistem pencatatan manual dalam bentuk *hardcopy* yang dikumpulkan di ruang *Preventive Maintenance* yang berpotensi menyebabkan kesulitan dalam akses dan analisis data secara cepat dan akurat. Kedua, sistem monitoring yang ada masih menggunakan *paper chart*, yang kurang praktis dan membutuhkan waktu lebih lama untuk mendapatkan gambaran keseluruhan kinerja oven. Ketiga, keterbatasan dalam membandingkan kinerja antar oven secara langsung dan *real-time* menyulitkan proses *monitoring* dan *troubleshooting* yang efisien. Oleh karena itu, diperlukan sebuah solusi yang dapat mengintegrasikan data kinerja oven secara digital, menyajikannya dalam satu layar monitor untuk kemudahan akses, serta memungkinkan perbandingan langsung antar oven untuk meningkatkan efisiensi operasional dan pemecahan masalah.

Dengan demikian, pemanfaatan teknologi *Internet of Things* (IoT) dalam berbagai sektor industri telah menunjukkan potensi besar dalam meningkatkan efisiensi operasional dan kualitas produksi. *Oven Despatch*, yang banyak digunakan dalam proses produksi, memerlukan sistem pemantauan yang handal untuk mengoptimalkan kinerja dan meminimalkan risiko kegagalan yang dapat mempengaruhi kualitas produk dan efisiensi operasional. Sehingga, penulis membuat suatu sistem pemantauan suhu yang dapat dipantau dari jarak jauh. Dengan sistem ini *oven* dapat dipantau secara langsung melalui tampilan *display* ataupun dipantau secara *online* dari jarak jauh melalui *web*.

1.2. Rumusan Masalah

Dari latar belakang diatas dapat diperoleh beberapa rumusan masalahnya, yaitu:

1. Bagaimana cara menerapkan pembacaan suhu dan status pintu *oven* yang dapat diakses dari jarak jauh?
2. Bagaimana cara menyediakan fitur yang memungkinkan pengguna menyimpan data grafik suhu secara langsung dalam format PDF?
3. Bagaimana merancang antarmuka pengguna (GUI) yang intuitif untuk sistem pemantauan otomatis pada *oven Despatch* sehingga pengguna dengan berbagai latar belakang teknis dapat memahami dan mengoperasikan sistem dengan mudah?

1.3. Tujuan

Dari rumusan masalah diatas dapat diperoleh target luarannya, yaitu:

1. Sistem dapat menampilkan data suhu dan status pintu *oven* secara *real-time* melalui prototipe sistem *monitoring* berbasis *web*, yang disertai dengan dokumen teknis.
2. Mengembangkan fitur yang memungkinkan penyimpanan data grafik suhu dalam format PDF.
3. Merancang antarmuka pengguna yang responsif dan intuitif untuk sistem pemantauan oven.

1.4. Manfaat

Dari rumusan masalah diatas dapat diperoleh beberapa manfaatnya, yaitu:

1. Dengan akses jarak jauh, operator dapat memantau suhu dan status pintu *oven* secara *real-time* dan merespons masalah atau perubahan dengan cepat, mengurangi *downtime*.
2. Dengan hasil dokumentasi grafik yang dihasilkan dalam format PDF, pengguna dapat dengan mudah melakukan tindak lanjut langkah-langkah *Preventive Maintenance*.
3. Dengan membuat antarmuka pengguna yang intuitif dan responsif, mempermudah pengguna dari berbagai latar belakang teknis untuk memahami, mengoperasikan, dan memanfaatkan sistem secara efisien.

1.5. Batasan

Untuk mewujudkan sistem pemantauan ini ada beberapa batasan masalah, yaitu:

1. Cakupan yang diambil mencakup kegiatan *monitoring* suhu, tidak termasuk mengontrol suhu.
2. *Web* yang dibangun untuk pemantauan sementara ini di*hosting* dengan layanan *free hosting.*
3. Sensor yang dibuat hanya untuk memantau suhu pada *oven* dan pintu *oven* ketika dalam keadaan terbuka atau tertutup, tidak beserta penggerak pintu.

1.6. Work Breakdown Structure

Tabel 1 merupakan pembagian tugas dan tanggung jawab setiap anggota untuk tercapainya hasil yang sesuai:

Tabel 1. Work Breakdown Structure

No	Nama	Tugas dan Tanggung Jawab dalam Tim
1	Mochammad Nuruddin Mustaqim	Pemantauan Melalui *Web*
2	Zidan Gunawan Kusumah	Pemantauan Melalui Perangkat

Bab 2. Tinjauan Pustaka

1.1. Penelitian Terkait

Tabel 2 merupakan rangkuman studi-studi sebelumnya, untuk membantu mengindentifikasi kurangnya informasi tentang pembuatan alat:

Tabel 2. Penelitian Terkait

No	Tahun	Penulis	Jurnal	Metode	Kelemahan
1	2019	Firdaus Hashim , Roslina Mohamad , Murizah Kassim , Saiful Izwan Suliman , Nuzli Mohamad Anas , Ahmad Zaki Abu Bakar	*Implementation of Embedded Real-Time Monitoring Temperature and Humidity System*	Arduino, IoT, Node-FRED, LoRa radio, dan *Things Network gateway*	Penelitian ini belum melakukan evaluasi kinerja sistem secara menyeluruh dalam kondisi nyata sehingga keamanan dan kehandalannya belum teruji sepenuhnya.
2	2023	Dr.S.VIMAL RAJ, K.SRIVASAN, R.VIJAY, K.VEL PRASANTH, B.SARATH PRANAV	*REAL-TIME-CLOCK USING ARDUINO*	DS3231 *Module,* Arduino, I2C *Protocol,* LCD *Screen*	Analisis dan pembahasan hanya terfokus pada penggunaan modul RTC saja

3	2019	Alaa Abdulhady Jaber, Firas Khalil Ibrahim Al-Mousawi, Hayder Sabeeh Jasem	*Internet of Things Based Industrial Environment Monitoring and Control: A Design Approach*	NodeMCU, *Gas and Temperature Sensors, Buzzer, Blynk Platform*	Penelitian ini belum melakukan uji coba ekstensif untuk mengetahui ketahanan sistem dalam jangka panjang
4	2019	Thiago dos Santos Alves, Edivaldo da Silva Alves Júnior, Fabiana Rocha Pinto, Claudio Henrique Albuquerque Rodrigues	*Development of a System for Monitoring and Control a Resin Drying Oven Using IoT*	ESP8266, DHT11*, free server, liquid crystal display*	Rentang suhu yang dianalisis masih terbatas, hanya 50-60°C.
5	2021	Yogatama Wishnu Pandu Prayudha , Sayid Muhammad Fadhil dan Sentot Novianto	Rancang Bangun Sistem Pengukuran Alat Thermobath sebagai Alat Kalibrasi Temperatur dengan Sistem Arduino Uno	Sensor Temperatur Termokopel Tipe-K, Arduino Uno, *Pressure Transmitter*	Ruang lingkup penelitian hanya mengenai rancang bangun sistem pengukuran thermobath tanpa Internet of Things

6	2020	Arulananth T S, Baskar M, P. Hari babu , R. Divya Sree , R.Sanjana, S.Bhavana	*IR Sensor Based Obstacle Detection and Avoiding Robot*	*IR Sensor, Relay-Based Switching Mechanism*	Penelitian berfokus hanya pada komponen sensor IR dan kendali motor robot. Tidak menjelaskan komponen IoT lain.
7	2021	Siti Purwanti, Anita Febriani, Mardeni, Yuda Irawan	*Temperature Monitoring System for Egg Incubators Using* Raspberry Pi3 *Based on Internet of Things (IoT)*	*Webcam Camera,* DHT11, Raspberry Pi3, *Smartphone*	Rentang suhu yang dapat diukur lebih sempit, yaitu hanya memantau suhu inkubator telur sekitar 39°C.
8	2023	Sebastianus Pehan Makin, Nachrowie, Subairi	Penerapan Metode *Fuzzy Sugeno* pada Otomatisasi *Oven* Pengering Ikan Asin Berbasis *IoT*	*Fuzzy Logic,* Arduino, ESP32, DHT22, RTC, LCD, aplikasi *Android, Blynk,* MATLAB	Penelitian ini hanya fokus pada proses pengeringan ikan asin saja tanpa memonitor pintu *oven*.

9	2019	Ekohariadi, Yeni Anistyasari, Muhamad Syarriefuddin Zuhrie, Ricky Eka Putra	Mesin *Oven* Pengering Cerdas Berbasis *Internet of Things* (IoT)	Pembuatan *Oven*, Sensor suhu termokopel, ESP 8266	Penelitian ini lebih berfokus terhadap pembuatan *oven* yang pengujiannya masih dalam skala laboratorium
10	2023	Kusmiyati, Arga Dwi Pambudi, Zaenal Arifin, Sari Ayu Wulandari, Muhammad Agus Purnomo, Kristoforus Adrian Setiadi, dan Nia Yunita Listianingrum	*Monitoring* Sistem Kontrol Mesin *Drying* Kopi Secara *Real Time* Berbasis IoT	ESP32, DHT22, *Smartphone* , Arduino, *Blynk*	Penelitian lebih berfokus ke pembuatan mesin pengering kopi yang berfokus pada parameter suhu dan kelembaban. Rentang suhu yang dikendalikan mesin pengering kopi hanya sampai 80°C.

Pada sub bab ini Untuk meningkatkan pengawasan dan analisis data produksi, penelitian ini akan menggabungkan teknologi IoT dengan aplikasi. Dengan menerapkan sistem pemantauan otomatis pada *Oven Despatch*, diharapkan dapat membuat keputusan operasional yang lebih baik dengan data yang relevan.

Penelitian oleh Firdaus Hashim, Roslina Mohamad, Murizah Kassim, Saiful Izwan Suliman, dan Nuzli Mohamad Anas dalam jurnal "*Implementation of Embedded Real-Time Monitoring Temperature and Humidity System"* membahas tentang implementasi sistem pemantauan suhu dan kelembapan secara *real-time* menggunakan *platform Embedded Arduino*. Berdasarkan hasil pengujian, Sistem pemantauan suhu dan kelembapan secara *real-time* berhasil dirancang dan memberikan pemahaman terhadap hubungan parameter lingkungan. Dengan demikian, tujuan penelitian untuk merealisasikan IoT dalam pemantauan parameter lingkungan tercapai. Namun, Penelitian ini belum melakukan evaluasi kinerja sistem secara menyeluruh dalam kondisi nyata sehingga keamanan dan kehandalannya belum teruji sepenuhnya [4]. Selanjutnya penelitian oleh Siti Purwanti, Anita Febriani, Mardeni, dan Yuda Irawan tentang "*Temperature Monitoring System for Egg Incubators Using Raspberry Pi3 Based on Internet of Things (IoT)"* secara abstrak, penelitian ini bertujuan untuk merancang sistem pemantauan suhu inkubator telur secara *real-time* menggunakan kamera *webcam, sensor* DHT11*,* Raspberry Pi3*,* dan perangkat *smartphone*. Sistem akan mendeteksi suhu menggunakan *sensor* DHT11, memantau kondisi telur menggunakan kamera *webcam*, serta mengontrol suhu menggunakan Raspberry Pi3 dan aplikasi *smartphone*. Tujuan penelitian ini adalah untuk merealisasikan pemantauan kondisi telur secara *real-time* dalam proses pengeraman serta menjaga kestabilan suhu di dalam inkubator telur. Sistem dirancang dan diuji coba untuk memverifikasi kinerjanya. Namun, rentang suhu yang dapat diukur lebih sempit, yaitu hanya memantau suhu inkubator telur sekitar 39°C [5]. Selain itu , penelitian oleh Thiago dos Santos Alves, Edivaldo da Silva Alves Júnior, dan Fabiana Rocha Pinto tentang "*Development of a System for Monitoring and Control a Resin Drying Oven Using IoT"* membahas tentang pengembangan sistem untuk pemantauan dan pengontrolan *oven* pengeringan resin menggunakan IoT). Secara abstrak, penelitian ini bertujuan untuk merancang prototipe sistem pemantauan dan pengontrolan *oven* pengeringan resin secara nirkabel menggunakan Mikrokontroler ESP8266, sensor suhu MAX6675, LCD, *relay* dan *broker* MQTT. Berdasarkan hasil pengujian, sensor suhu MAX6675 dapat membaca suhu *oven* pengeringan resin diantara 130 °C hingga 150 °C. Namun, penggunaan ESP8266 memiliki daya pemrosesan lebih rendah karena hanya memiliki satu core MCU, sehingga kapasitas untuk mengeksekusi aplikasi yang kompleks terbatas.

Penggunaan teknologi *Internet of Things* (IoT) diharapkan dapat membuat keputusan operasional yang lebih baik berdasarkan data relevan. Berdasarkan penelitian terdahulu, mikrokontroller ESP32 dipilih karena memiliki kapasitas pemrosesan yang lebih besar dibandingkan ESP8266. ESP32 memiliki dua core MCU sehingga mampu mengeksekusi aplikasi IoT yang kompleks. Untuk mendeteksi suhu, sensor MAX6675 dipilih karena mampu membaca suhu antara 0°C - 1024°C yang sesuai dengan rentang operasi *Oven Despatch* - VRC2-19-1E.

Selain itu, ntpServer juga digunakan untuk membantu *user* mendapatkandata yang reliabel dan *real time* [6]. Pengawasan dan pengolahan data suhu secara*real-time* diharapkan dapat mendukung proses pengambilan keputusan. Dengan menerapkan sistem pemantauan otomatis menggunakan ESP32 dan MAX6675 diharapkan dapat meningkatkan kinerja pengawasan dan menghasilkan analisis data produksi yang lebih relevan untuk keperluan pengambilan keputusan operasional yang lebih baik.

1.2. Tinjauan Teori

1.2.1. Sistem IoT (Internet of Things)

Teknologi *Internet of Thing* (IoT) memungkinkan pengguna mengakses secara*online* [7]. Orang-orang dapat terhubung secara terus menerus dengan IoT. IoT telah mengubah segalanya, baik di bisnis, pertanian, atau lingkungan umum.

Adanya sensor yang berfungsi sebagai memasukan data di lapangan, perangkat pengolah data sensor yang terhubung ke jaringan internet, dan tampilan antarmuka pengguna (UI) dari data di lapangan adalah komponen penting dari pembentukan sistem IoT sederhana [8].

Pada penelitian ini, sistem IoT digunakan untuk mengirimkan data suhu produk ke database dan menampilkannya pada *interface*. Mikrokontroler NodeMCU mengolah data suhu dari hasil pembacaan sensor, yang kemudian dikirim ke database dan tampilan *interface* menggunakan konektivitas jaringan *Wi-Fi*.

1.2.2. NodeMCU

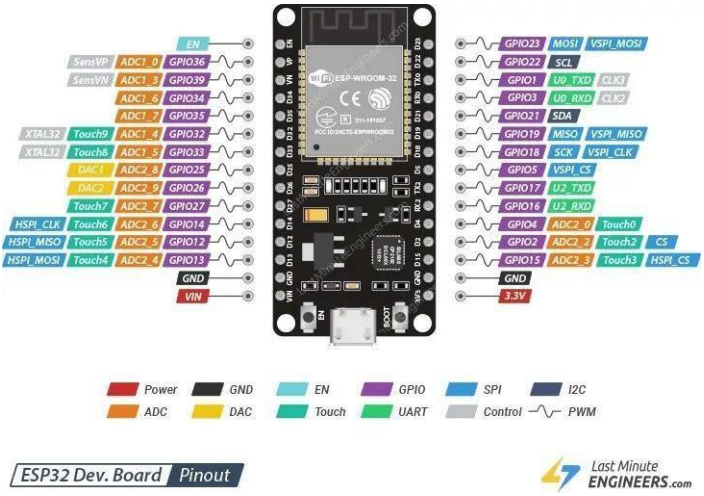
NodeMCU ESP32 adalah sebuah papan elektronik berbasis *chip* yang memiliki fungsi serupa dengan mikrokontroler dan dilengkapi dengan kemampuan koneksi internet *(WiFi)* [9]. Sehingga, NodeMCU ini sangat mendukung pengembangan sistem aplikasi berbasis internet [10]. Seperti Arduino, NodeMCU mendukung bahasa pemrograman C/C++ dan dapat diprogram menggunakan Arduino IDE *(Integrated Development Environtment).*

NodeMCU ESP 32 memiliki beberapa pin yang sangat penting untuk terhubung dengan perangkat eksternal, seperti pin SPI dan pin GPIO. Pin *S*PI digunakan untukberkomunikasi menggunakan protokol SPI, sedangkan pin GPIO berfungsi sebagai pin *input-output* yang fleksibel. Selain itu, NodeMCU 32 memiliki port micro USB yang dapat digunakan untuk pemrograman dan mengupload kode program ke dalam memory NodeMCU ESP32.

Ini menunjukkan tampilan dari board mikrokontroler ESP32, yang memiliki bentuk persegi panjang dengan berbagai komponen elektronik terpasang di atasnya. Gambar 1 merupakan bentuk fisik dari papan NodeMCU ESP32 :

Gambar 1. NodeMCU ESP32 Board.

Ini menampilkan diagram detail dari board ESP32 beserta label untuk setiap pin dan fungsinya. Diagram ini menunjukkan berbagai pin input/output, pin daya, pin ground, dan pin khusus lainnya yang tersedia pada board NodeMCU ESP32, memberikan gambaran komprehensif tentang konektivitas dan fungsionalitas board tersebut. Gambar 2 merupakan papan NodeMCU ESP32 lengkap dengan *pinout*-nya:



Gambar 2. NodeMCU ESP32 Pinout.

1.2.3. Modul *Driver Thermocouple*

Modul MAX6675 memiliki fungsi mengukur suhu dalam rentang suhu yang luas. Ini sangat cocok untuk aplikasi seperti pemantauan suhu sistem yang membutuhkan kecepatan respon cepat dan akurasi tinggi. Sistem pengukuran ini menggunakan termokopel tipe K sensor suhu dengan modul MAX 6675 yang dapat membaca dari 0 hingga +1024°C [12]. Modul ini terdiri dari sebuah papan sirkuit kecil berwarna biru dengan beberapa komponen elektronik di atasnya. Gambar 3 merupakan bentuk fisik dari Modul *Driver Thermocouple* MAX6675 yang akan digunakan:

Gambar 3. MAX6675.

1.2.4. Termokopel Tipe-K

Ada berbagai jenis termokopel, seperti Tipe-J, Tipe-K, Tipe-E, Tipe-T, dll., berdasarkan kombinasi logam atau paduan yang digunakan untuk kedua kabel dan memiliki ketahanan korosi yang sangat baik, sehingga dapat digunakan di sebagian besar aplikasi. Modul memiliki kemampuan untuk memproses berbagai sinyal suhu yang dikumpulkan oleh termokopel tipe-K. Kemudian, melalui penggunaan konverter *(ADC)* dalam modul, modul dapat mengkonversi sinyal analog menjadi sinyal digital [12].

Namun, termokopel yang paling banyak digunakan dalam aplikasi industri adalah tipe-K, karena termokopel ini dapat diprediksi bereaksi pada rentang suhu yang luas (sekitar -328 °F hingga +2300 °F) dan memiliki sensitivitas sekitar 41 µV/°C. Terdiri dari kawat positif yang terbuat dari Chromel (paduan nikel-kromium) dan kawat negatif yang terbuat dari Alumel (paduan nikel-aluminium). Gambar 4 merupakan bentuk fisik dari Termokopel yang memiliki Tipe-K:

Gambar 4. Termokopel.

1.2.5. Sensor IR

IR Sensor dapat mendeteksi apakah pintu *oven* terbuka atau tertutup. Ini dilakukan dengan memasang sensor di bagian atas pintu, sehingga dia dapat melihat apakah pintu ada di bawahnya. Sensor inframerah mengirimkan radiasi infra merah dari LED inframerah ke area bawah *oven.*

Saat pintu terbuka, sebagian radiasi akan menembus lurus ke lantai tanpa pantulan kembali ke photodiode. Saat pintu tertutup, sebagian radiasi akan dipantulkan oleh permukaan pintu [13]. Gambar 5 menunjukkan komponen-komponen utama dari sebuah sensor IR (Infrared):

Gambar 5. Sensor IR.

1.2.6. Buzzer

Buzzer adalah bagian elektronik yang dapat membuat suara atau nada. Berbagai jenis alarm, indikator, dan sistem peringatan menggunakan buzzer. Buzzer aktif dan pasif adalah dua jenis buzzer yang paling umum digunakan. Masing-masing memiliki fitur dan cara bekerja yang berbeda. Pada gambar 6 dapat dilihat bentuk fisik dari buzzer:

Gambar 6. Buzzer.

1.2.7. Pembuatan *Cover* Komponen Elektrikal

Cover merupakan komponen penting dalam melindungi perangkat elektrikal dari faktor eksternal yang dapat mempengaruhi kinerjanya. *Cover* dirancang untuk memberikan tingkat perlindungan terhadap debu dan cairan. Standar ini mengatur bagaimana *cover* harus dibuat agar aman dan sesuai dengan kebutuhan perlindungan yang ditentukan. Untuk lingkungan ruangan, tingkat perlindungan IP54 umumnya memadai, melindungi dari debu dan percikan air. Tingkat perlindungan ini cocok untuk lingkungan dalam ruangan, ideal untuk peralatan di industri ringan atau komersial, melindungi dari akumulasi debu dan percikan air ringan, namun tidak cocok untuk paparan air langsung atau lingkungan ekstrem. IP54 sering digunakan pada peralatan elektrikal di pabrik dan sistem kontrol, memungkinkan ventilasi terbatas untuk pendinginan komponen, tetapi tetap memerlukan pemeriksaan dan perawatan berkala meskipun terlindungi. Pengujian terhadap cover ini dapat dilakukan sesuai dengan ketentuan yang terdapat dalam SNI IEC 60529:2014. Melalui proses pengujian ini, dapat dipastikan bahwa cover memenuhi standar yang diperlukan untuk melindungi komponen elektrikal dari debu dan kelembapan, sehingga meningkatkan keandalan serta ketahanan perangkat. Dengan pelaksanaan pengujian yang sesuai, dapat dibuktikan bahwa perangkat yang dilindungi oleh cover ini akan beroperasi secara optimal dalam kondisi yang telah ditentukan.

1.2.8. FireBase

Penggunaan Firebase dalam sistem pemantauan oven Despatch akan membawa perubahan dan keuntungan signifikan bagi proyek ini. Firebase adalah platform pengembangan aplikasi yang menyediakan berbagai layanan, termasuk database NoSQL yang dikenal sebagai Firestore, yang sangat cocok untuk menyimpan data sensor dan IoT dalam format fleksibel. Firestore memungkinkan data disimpan dalam bentuk dokumen JSON, yang mudah diakses dan diubah.

Dalam implementasinya, data dari sensor suhu MAX6675 dan sensor IR untuk pintu oven akan disimpan sebagai dokumen dalam koleksi Firestore. Setiap dokumen akan merepresentasikan satu titik data, memungkinkan penanganan struktur data yang fleksibel dan dinamis. Untuk mengirim data dari NodeMCU ESP32 ke Firebase, Anda dapat menggunakan Firebase SDK untuk mengautentikasi dan mengirim data secara langsung ke Firestore melalui API yang telah disediakan.

Untuk antarmuka *web*, pengembangan bisa dilakukan menggunakan *framework* modern seperti React atau Vue.js, yang berkomunikasi dengan API *backend* untuk mengambil dan menampilkan data dari Firebase. Visualisasi data bisa ditingkatkan dengan penggunaan *library* seperti Chart.js atau D3.js.

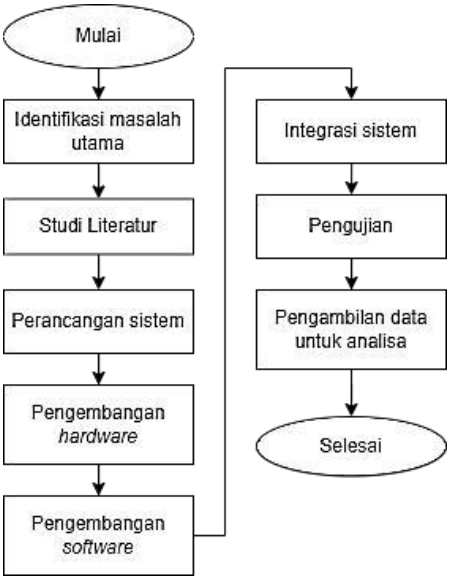
1.2.9. Visual Studio

Visual Studio adalah *Integrated Development Environtment* (IDE) yang dikembangkan oleh Microsoft yang memungkinkan Anda membuat GUI untuk aplikasi bisnis, personal, konsol, Windows, dan *Web*. Editor teks ini mendukung bahasa pemrograman JavaScript, Typescript, dan Node.js secara langsung, serta bahasa pemrograman lainnya dengan bantuan plugin yang dapat dipasang melalui marketplace Visual Studio Code, seperti C++, C#, Python, Go, Java Script, PHP, dan sebagainya [15]. Bahasa pemrograman yang dipakai untuk perancangan GUI pada proyek penelitian ini adalah Java Script.

Bab 3. Metodologi Penelitian

3.1. Perancangan

Gambar 7 ini merupakan flowchart pelaksanaan dari penelitian yang akan dilakukan :



Gambar 7. *Flowchart* pelaksanaan penelitian.

Pertama, identifikasi masalah akan menjelaskan masalah dalam sistem pemantauan saat ini, di mana proses pemantauan dilakukan secara manual dan sulit untuk memantau *oven* secara *real-time* dari luar ruangan.

Setelah itu studi literatur adalah tahap awal penelitian. Dengan melakukan studi literatur, maka dapat menentukan rancangan proyek dan mendapatkan informasi yang dibutuhkan untuk proyek tersebut. Tahap berikutnya adalah perancangan sistem. Baik dari segi *hardware* (mekanikal dan elektrikal) maupun *software* (*user interface* dan program).

Tahap selanjutnya perancangan sistem akan menjelaskan bagaimana sistem terdiri dari sensor, mikrokontroler, dan cloud server, serta perancangan program untuk mikrokontroler dan antarmuka *monitoring* berbasis *web*. Sistem akan menggunakan thermocouple untuk memantau suhu *oven* secara konsisten.

Mikrokontroler NodeMCU ESP32 akan terhubung ke sensor untuk mengontrol dan mengumpulkan data suhu dari sensor.

Pada tahap pengembangan *hardware*, akan dilakukan pengintegrasian sensor suhu dan waktu pada mikrokontroler. Pengembangan modul nirkabel untuk komunikasi antara mikrokontroler dan cloud server. Serta pembuatan *cover* untuk melindungi komponen elektrikalnya.

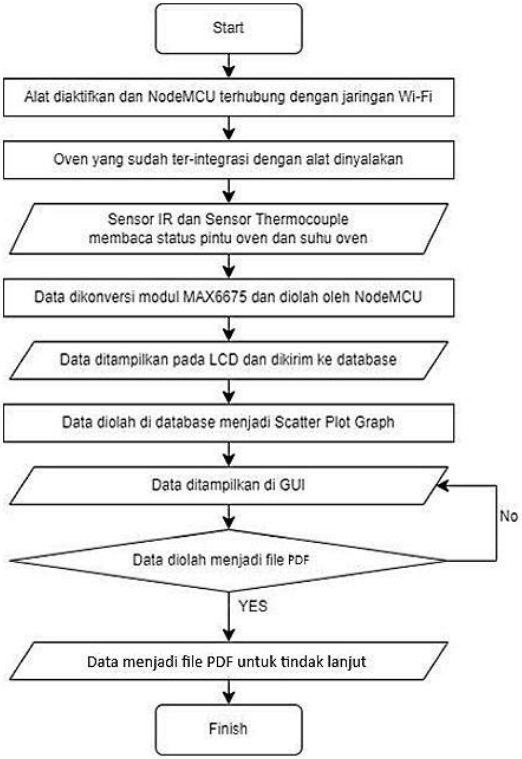
Selanjutnya tahap pengembangan *software*, perangkat lunak yang dibutuhkan akan dikembangkan. Program C++ akan digunakan untuk membuat program untuk mikrokontroller yang dapat mengontrol sensor suhu, mengumpulkan data suhu secara berkala, dan mengirimkan data tersebut ke *server cloud*. Selain itu, program untuk *server cloud* akan dikembangkan menggunakan platform *firebase*. Ini akan memungkinkan program untuk menerima data dari mikrokontroller, menyimpan dan mengolah data ke dalam basis data cloud, dan antarmuka *web real-time* untuk menampilkan data suhu secara *real-time*.

Pada tahap integrasi system mikrokontroler akan menggunakan jaringan nirkabel untuk menghubungkan dan mengirimkan data ke *cloud server*. *Cloud server* akan menggunakan *firebase* untuk menyimpan dan mengolah data, dan platform XAMMP digunakan untuk membangun antarmuka pemantauan berbasis *web*. Pengguna dapat memantau kondisi suhu *oven* secara *real-time* dari mana saja yang terhubung ke internet melalui antarmuka ini.

Setelah semua perangkat lunak dan perangkat keras disiapkan, langkah selanjutnya adalah menguji sistem secara keseluruhan. Pengujian sangat penting untuk memastikan bahwa sistem bekerja sesuai fungsi dan spesifikasi yang diharapkan. Hasil pengujian akan menjadi dasar untuk melakukan perbaikan dan penyempurnaan sistem sebelum diimplementasikan pada aplikasi sebenarnya.

Urutan langkah kerja alat adalah; setelah *oven* mulai memanaskan IC, NodeMCU diaktifkan dan terhubung ke jaringan *Wi-Fi* untuk memungkinkan data berkomunikasi. *Oven* yang telah terintegrasi dengan sistem dinyalakan untuk memulai proses pemantauan. Termokopel mengukur suhu di dalam *oven* dan sensor *IR* mendeteksi apakah pintu *oven* terbuka atau tertutup. Data suhu dari termokopel dikonversi ke modul MAX6675 dan kemudian diproses oleh NodeMCU. Data yang telah diproses dikirim ke database untuk analisis dan penyimpanan tambahan sebelum ditampilkan pada layar LCD. Grafik *scatter plot* yang menunjukkan hubungan antara *variabel* yang dipantau kemudian dibuat dari data yang disimpan di *database*. Pada grafik *scatter plot* dan data lainnya ditampilkan pada antarmuka pengguna grafis (GUI) untuk memudahkan pemantauan dan analisis. Setelah itu, sistem memeriksa apakah data perlu diolah menjadi file PDF untuk analisis lebih lanjut. Jika tidak, proses kembali ke langkah sebelumnya (data ditampilkan di GUI).

Pada gambar 8 Jika *user* ingin mengolah data menjadi file PDF, data tersebut diolah dan disimpan dalam format file PDF untuk keperluan analisis lebih lanjut atau pelaporan. Gambar 8 ini merupakan *flowchart* cara kerja alat :



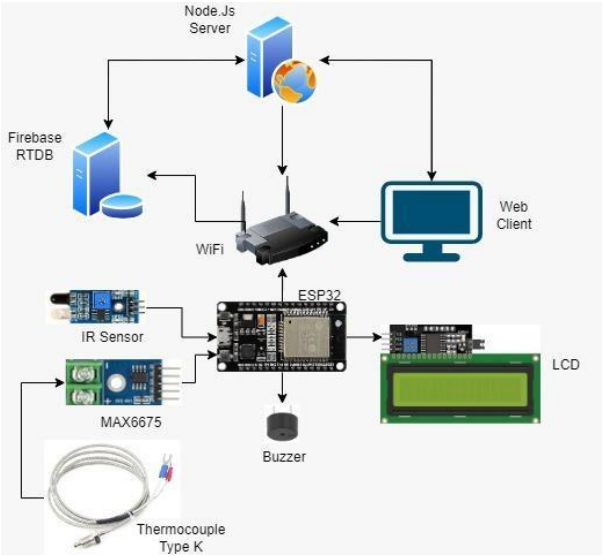
Gambar 8. *Flowchart* cara kerja.

Perancangan pembuatan alat terbagi menjadi dua bagian, yaitu:

1. Perancangan perangkat keras (hardware)
   Merupakan rancangan pembuatan alat.
2. Perancangan perangkat lunak (software)
   Merupakan rancangan pembuatan aplikasi mulai dari GUI hingga pemrograman alat.

3.1.1. Perancangan Perangkat Keras (Hardware)

Gambar 9 merupakan diagram blok dari sistem yang menghubungkan antar komponen yang digunakan :



Gambar 9. Diagram blok sistem *hardware*.

Termokopel merupakan komponen *input* atau masukan yang berperan untuk memberikan sinyal masukan. Sensor termokopel untuk mengukur suhu di dalam *oven*. IR *proximity sensor* mendeteksi status pintu *oven* (terbuka atau tertutup). Modul MAX6675 dan NodeMCU merupakan komponen pemroses sinyal yang diterima dari komponen input. LCD, *database MySQL* dan *Graphical User Interface* merupakan komponen output atau keluaran yang menampilkan hasil dari pemrosesan sinyal dari komponen pemroses sinyal. Sementara LCD menerima sinyal langsung dari pin NodeMCU, *MySQL* dan GUI menerima sinyal olahan dari NodeMCU melalui jaringan *Wi-Fi*.

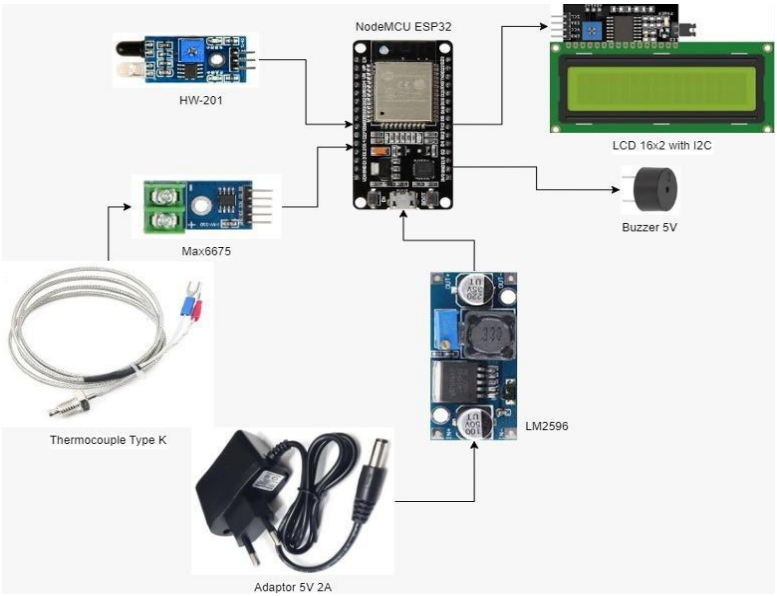
Lalu, dalam pembuatan *cover* komponen elektrikal menggunakan plastik ABS (*Acrylonitrile Butadiene Styrene*) berkualitas tinggi yang kuat. Desain cover untuk menampung komponen elektrikal dengan ventilasi berupa lubang-lubang kecil berdiameter 1-2 mm, berjarak 5 mm antar lubang, serta lubang untuk kabel keluar cover, ukuran disesuaikan dengan diameter kabel yang digunakan. Memasang *cable gland* IP54 pada lubang kabel keluar *cover* untuk penyegelan yang baik.

3.1.1.1. Perancangan Elektrikal

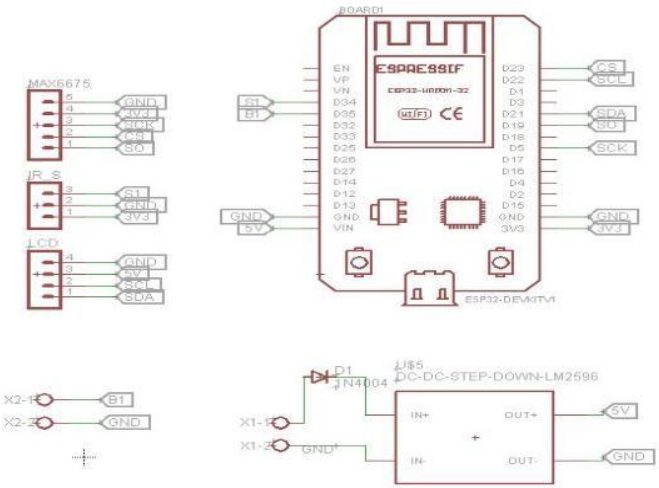
ESP32 merupakan mikrokontroler yang berfungsi sebagai pusat pengolahan data dalam rangkaian ini. Komponen input seperti modul MAX6675 yang terhubung dengan thermocouple type K untuk membaca suhu, sementara modul HW-201 mendeteksi kondisi pintu. Data hasil pengukuran ditampilkan pada LCD 16x2 melalui modul LCM1602 IIC, semua terhubung ke pin GPIO pada ESP32.

Modul MAX6675 menerima data suhu dari thermocouple type K dan mengubahnya menjadi sinyal digital yang dapat diproses oleh ESP32. Tegangan kerja modul ini sebesar 3,3V, sehingga suplai daya diberikan melalui pin 3,3V pada board ESP32. Sebaliknya, modul LCM1602 IIC yang digunakan untuk mengontrol LCD 16x2 memerlukan tegangan 5V, sehingga dayanya diambil dari pin VIN pada ESP32 yang disuplai melalui adaptor 5V 2A. Komponen diode 1N4007 digunakan untuk melindungi rangkaian dari arus balik yang dapat merusak perangkat.

Pada gambar 10 regulator tegangan LM2596 menyesuaikan tegangan input 5V agar stabil untuk beberapa perangkat. Selain itu, buzzer terhubung ke ESP32 sebagai indikator suara, diaktifkan melalui pin GPIO. Dengan konfigurasi ini, rangkaian dapat mengintegrasikan berbagai komponen input dan output secara optimal. Pada gambar 10 dan 11 merupakan skema elektrikal pada alat yang dirancang menggunakan aplikasi Draw.io dan Eagle:



Gambar 10. Rancangan elektrikal.



Gambar 11. *Schematic* elektrikal.

Tabel 3 merupakan koneksi pin antar komponen sesuai skema elektrikal pada Gambar 10:

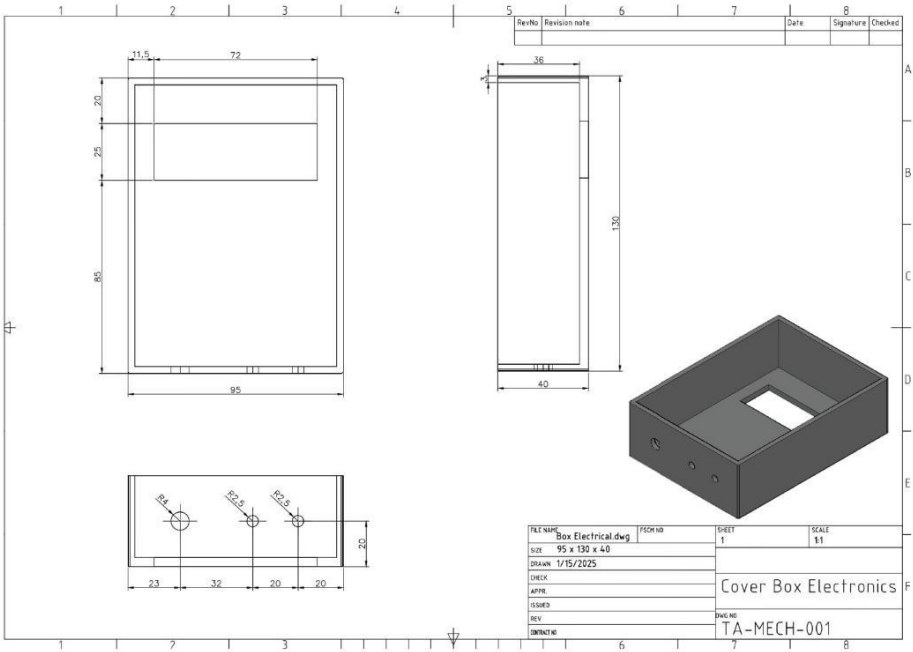
Tabel 3. Koneksi pin antar komponen

Komponen	Pin	Dikoneksikan dengan Pin	Pada komponen	Keterangan
MAX6675	+	Kabel Merah	Thermocouple type K	-
	-	Kabel Biru		-
	GND	GND	ESP32 Dev Kit V1	-
	VCC	3V3(3.3V+)		-
	SCK	D5 (GPIO 5)		-
	CS	D23 (GPIO 23)		-
	SO	D19 (GPIO 19)		-
LCD 16X2	GND	GND	LCM1602 I2C	-
	VCC	VCC		-
	VEE	VEE		-
	RS	RS		-
	RW	RW		-
	EN	EN		-
	D0	D0		-
	D1	D1		-
	D2	D2		-
	D3	D3		-
	D4	D4		-

	D5	D5		-
	D6	D6		-
	D7	D7		-
	LED+	LED+		-
	LED-	LED-		-
LCM1602 I2C	VCC	VIN (5V+)	ESP32 Dev Kit V1	I 2C address 0x27, LED backlight jumper terpasang
	GND	GND		
	SCL	D22 (GPIO 22)		
	SDA	D21 (GPIO 21)		
IR Obstacle Avoidance Sensor	VCC	3V3 (3.3V+)		-
	GND	GND		-
	OUT	D34 (GPIO 34)		-
Buzzer	+	D35 (GPIO 35)		-
Koneksi untuk jalur *power* E SP32 Dev Kit V1				
LM2596 DC-DC Step Down	IN+	Kabel Merah	Adaptor 5V 2A	Diode 1n4007
	IN-	Kabel Hitam		-
	OUT+	VIN	ESP32 Dev Kit V1	-
	OUT-	GND		-

3.1.1.2. Perancangan Mekanikal

Rancangan mekanik diperlukan untuk membuat cover dengan baik. Adanya perencanaan juga memudahkan peneliti untuk membuat cover untuk projek ini. Cover berfungsi untuk mengamankan komponen yang digunakan. Dimensi panjang dan lebar cover disesuaikan dengan ukuran PCB serta modul yang digunakan. Panel box yang berbahan dasar Filament PLA+ dibentuk menggunakan mesin router yang kemudian di-potting menggunakan resin untuk melindungi komponen yang berada didalam cover. Pada gambar 12 merupakan desain alat yang akan dibuat yang dirancang menggunakan aplikasi AutoCAD 2020:



Gambar 12. Desain mekanikal.

Panel box yang berbahan dasar Filament PLA+ dibentuk menggunakan mesin router yang kemudian di-potting menggunakan resin untuk melindungi komponen yang berada didalam cover.

3.1.1.3. Desain Sistem di Oven Despatch

Dalam rangka memastikan efektivitas dan keamanan operasional oven, penempatan alat-alat pemantauan dan pengendalian memiliki peranan yang sangat penting. Oleh karena itu, pada gambar 13 merupakan desain yang tepat untuk penempatan alat ini akan mempengaruhi kinerja sistem secara keseluruhan:

Gambar 13. Desain Sistem di Oven Despatch

Gambar tersebut menunjukkan oven despatch yang dilengkapi dengan IR Sensor dan termokopel untuk pengukuran suhu. Oven ini berfungsi untuk pemanasan atau pengeringan IC dengan suhu yang stabil. IR Sensor untuk mengetahui kondisi pintu, sementara termokopel type k menggunakan modul MAX6675 untuk menghasilkan tegangan yang berkaitan dengan suhu. Semua data dari sensor ini diproses oleh alat mikrokontroler, yang memungkinkan kontrol dan pemantauan suhu oven secara *real-time*, meningkatkan efisiensi dan akurasi dalam proses.

Tabel 4 merupakan koneksi pin antar komponen sesuai skema elektrikal pada Gambar 12:

Tabel 4. Keterangan *part* pada alat.

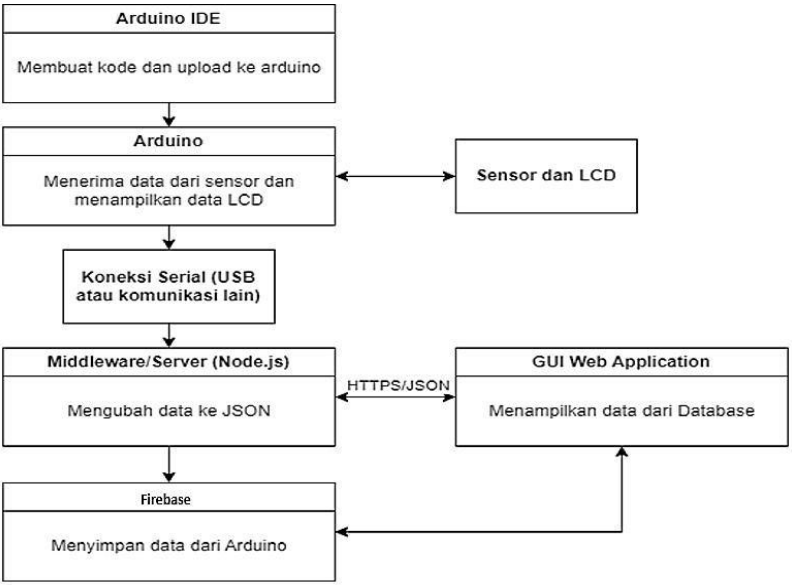
No	Nama Part	Keterangan	Material dan Dimensi	
1	IR Infrared	Sensor suhu inframerah mengukur suhu tanpa kontak langsung dengan objek. Ia menggunakan radiasi inframerah yang dipancarkan oleh objek untuk menentukan suhu.	10 mm x 10 mm x 5 mm	
2	MAX6675 K-type Thermocouple	Converter suhu dalam rentang yang luas (biasanya -200°C hingga 1350°C)	3 cm x 2.5 cm	
3	ESP32 Devkit V1	Untuk menghubungkan berbagai sensor dan perangkat, serta mengirim data ke server atau aplikasi lain melalui jaringan.	55 mm x 25 mm.	
4	Buzzer	Sebagai alat peringatan atau notifikasi dalam berbagai aplikasi, seperti alarm suhu atau pengingat.	Plastik	12 mm
5	LCD 1602 16x2	Menampilkan informasi, seperti suhu yang diukur	70 x 29 x 6 mm	
6	LM2596 - DC Converter	Mengubah tegangan input yang lebih tinggi ke rendah	4 cm x 2 cm x 1 cm	
7	Temperature Sensor Termocouple	Mengukur suhu dalam rentang yang luas (biasanya -200°C hingga 1350°C)	30 cm	
8	Cover Box Elektronik	Wadah yang digunakan untuk melindungi komponen elektronik dari debu, kelembapan, dan kerusakan fisik.	PLA+	130x40x95mm

Berikut adalah rincian mengenai posisi optimal dari masing-masing alat yang akan digunakan dalam sistem pemantauan oven Despatch:

1. Mikrokontroler diletakkan di luar oven despatch untuk menghindari paparan langsung terhadap panas dan sudah dilengkapi dengan *cover box* sesuai dengan SNI IEC 60529:2014. Hal ini bertujuan untuk menjaga kinerja mikrokontroler tetap stabil.
2. Sensor inframerah dipasang di dalam oven, tepatnya di posisi yang strategis untuk mendeteksi keadaan pintu oven atau tertutup.
3. Termokople diletakkan di dalam oven, berfungsi untuk mengukur suhu dengan presisi tinggi. Penempatan ini memungkinkan termocouple untuk memberikan informasi suhu yang *real-time*.

3.1.2. Perancangan Perangkat Lunak (*Software*)

Gambar 14 memberikan gambaran lengkap alur data dari sensor, melalui Arduino, ditampilkan pada LCD, disimpan di *database*, dan kemudian ditampilkan pada GUI Web:

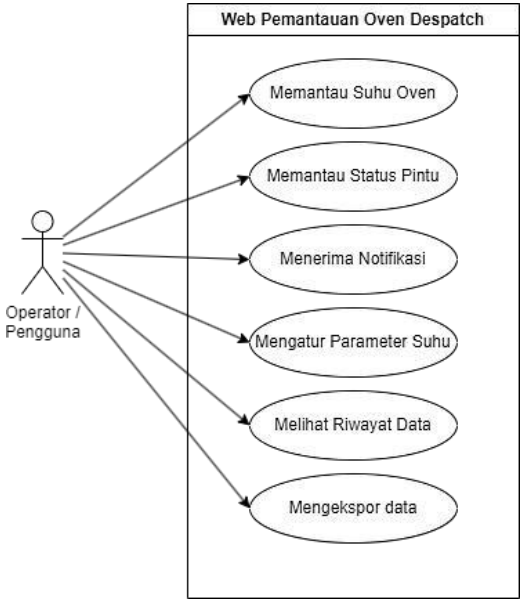


Gambar 14. Diagram blok sistem *software*.

Sistem dimulai dengan pembuatan kode di Arduino IDE, yang kemudian diunggah ke papan Arduino. Arduino berfungsi untuk mengumpulkan data dari sensor yang terhubung dan menampilkan data tersebut pada LCD sebagai aktuator. Selain menampilkan data pada LCD, Arduino juga mengirimkan data melalui koneksi serial (seperti USB) ke komputer. Di komputer, sebuah aplikasi *middleware* atau server (misalnya Node.js) menerima data dari koneksi serial dan mengonversi data tersebut menjadi format JSON. Data dalam format JSON ini kemudian disimpan dalam *database* NoSQL Firebase. Untuk menampilkan data kepada pengguna, sebuah antarmuka pengguna grafis (GUI) berbasis web dibuat menggunakan teknologi *web* seperti HTML, CSS, dan JavaScript (misalnya, menggunakan *framework* seperti React atau Angular). Aplikasi *web* ini mengambil data dari Firebase melalui API yang disediakan oleh server, biasanya menggunakan HTTP dan RESTful API. Dengan demikian, sistem ini memungkinkan pemantauan dan pengelolaan data sensor secara efisien dengan tampilan *real-time* pada LCD dan antarmuka *web* yang responsif.

3.1.3. Use-Case Standard UI/UX

Gambar 15 memberikan gambaran tentang alur *use-case* dengan standar UI/UX:



Gambar 15. *Use-Case* diagram blok sistem *software*.

User Interface adalah persepsi dan respons seseorang saat menggunakan produk, sistem, atau jasa. Sementara itu, *user experience* adalah input dan output yang langsung terhubung ke sistem pengguna akhir, metode desain berpusat pengguna untuk Website[16]. *Web* pemantauan otomatis pada *oven despatch* dengan IoT ini memiliki beberapa *use-case* utama yang mencerminkan fungsionalitasnya. Operator atau pengguna dapat memantau suhu oven dan status pintu secara *real-time* melalui antarmuka *web*, memberikan visibilitas yang tinggi terhadap proses produksi. Sistem juga dirancang untuk mengirimkan notifikasi kepada pengguna jika suhu oven keluar dari rentang yang diinginkan, memungkinkan respons cepat terhadap anomali. Pengguna dapat melihat dan menganalisis data historis suhu oven, serta mengatur parameter suhu yang diinginkan dan batas untuk notifikasi, memberikan fleksibilitas dalam pengelolaan proses.

Sistem juga dirancang untuk mengirimkan notifikasi kepada pengguna jika suhu oven keluar dari rentang yang diinginkan, memungkinkan respons cepat terhadap anomali. Pengguna dapat melihat dan menganalisis data historis suhu oven, serta mengatur parameter suhu yang diinginkan dan batas untuk notifikasi, memberikan fleksibilitas dalam pengelolaan proses. Keunggulan sistem ini terletak pada kemampuannya untuk diakses dari jarak jauh, memungkinkan pemantauan dan pengelolaan dari lokasi yang berbeda melalui internet. Fitur ekspor data memungkinkan pengguna untuk mengunduh informasi suhu dan status pintu untuk analisis lebih mendalam, mendukung pengambilan keputusan berbasis data.

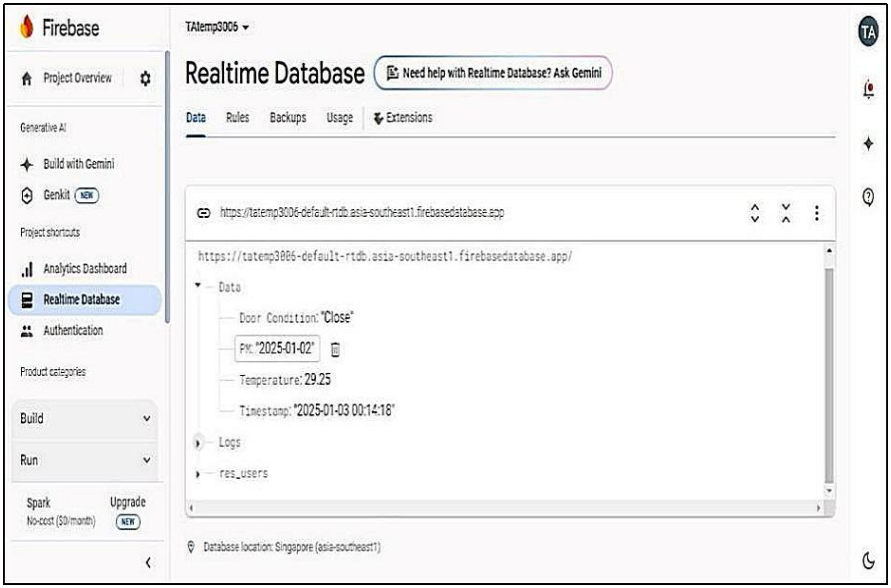
Selain itu, sistem mencatat dan menampilkan waktu produksi menggunakan modul ntpServer, memberikan wawasan tambahan tentang efisiensi proses. Secara keseluruhan, *use case* ini mencakup aspek-aspek kunci dari pemantauan otomatis, termasuk pengumpulan data *real- time*, notifikasi, analisis historis, dan aksesibilitas jarak jauh, yang semuanya bertujuan untuk meningkatkan efisiensi dan kualitas proses produksi yang melibatkan oven despatch.

3.1.3.1. *Graphical User Interface (GUI)*

Graphic User Interface (GUI) digunakan untuk menampilkan data pengukuran suhu dan kondisi terbuka atau tertutupnya pintu pada oven Despatch VRC2-19-1E serta jadwal *preventive maintenance*. Informasi yang ditampilkan pada GUI diantaranya adalah suhu *real-time*, kondisi pintu oven, grafik data pengukuran suhu, dan riwayat pencatatan hasil pengukuran suhu.

Sistem pemantauan oven ini memanfaatkan *Firebase Realtime Database* sebagai *backend* untuk menyimpan dan menyinkronkan data secara *real-time*. Firebase Realtime Database adalah sebuah database NoSQL yang berbasis cloud yang memungkinkan data disimpan dalam format JSON dan disinkronkan secara instan ke semua perangkat yang terhubung.

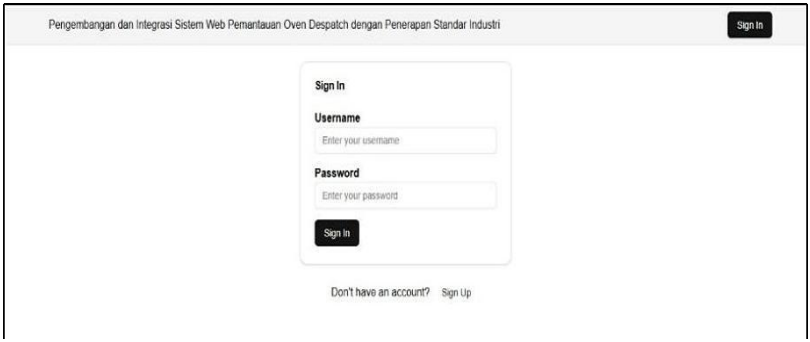
Pada gambar 16 **Data** berisi informasi realtime oven, lalu pada ***Logs*** berisi riwayat pencatatan dan ***res_users*** berisi informasi pengguna yang dipanggil saat pengguna ingin masuk ke tampilan *dashboard*:



Gambar 16. *Realtime Database.*

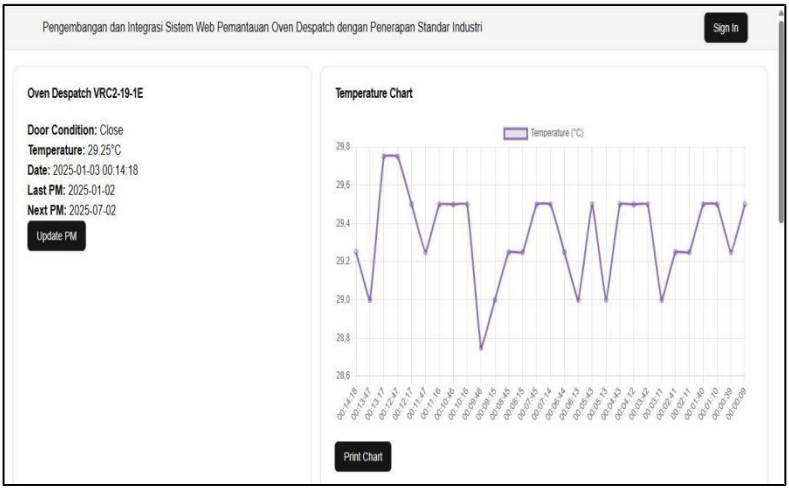
Pengembangan antarmuka pengguna (GUI) sistem ini mengadopsi kerangka kerja React.Js. Komponen-komponen antarmuka, seperti grafik suhu dan riwayat pencatatan, dibangun secara modular menggunakan React.Js. Di sisi *backend*, Firebase digunakan sebagai *database* NoSQL yang fleksibel untuk menyimpan data suhu oven secara *real-time*. Untuk menghubungkan *frontend* dan *backend*, kami menggunakan Node.js sebagai server. Server Node.js berfungsi sebagai lapisan tengah (middleware) yang menangani permintaan dari React.js, berinteraksi dengan Firebase, dan mengirimkan *respons* kembali ke *frontend*.

Pada gambar 17 dan 18 *Web* ini menggunakan dua tampilan yaitu *Sign in* dan *Dashboard*. *Sign-in* pada gambar 17 berfungsi untuk *Sign-in user*:

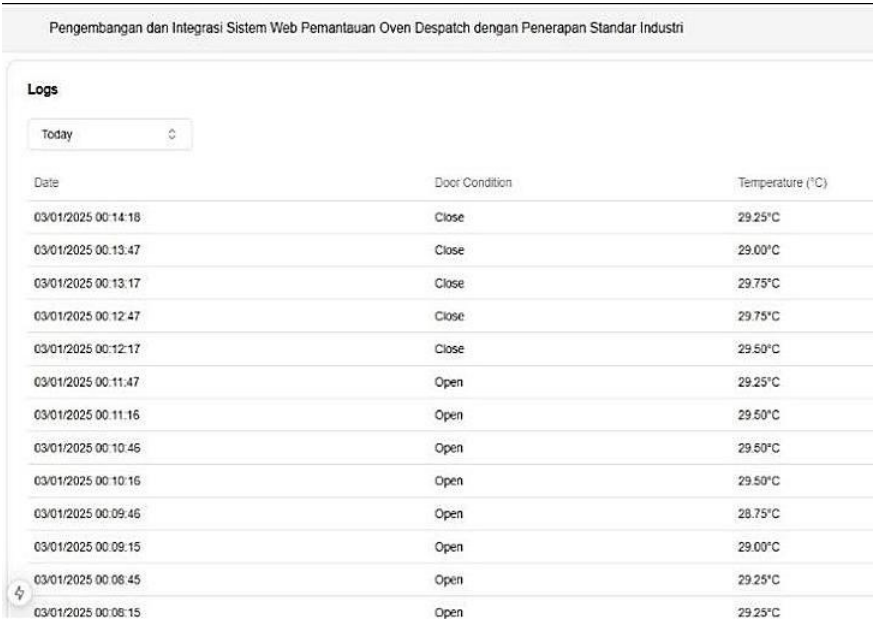


Gambar 17. *Sign In.*

Gambar 18 dan 19 merupakan tampilan dari *dashboard monitoring* oven dispatch menampilkan tiga *card*. *Card* pertama menampilkan data *realtime*, *card* kedua menampilkan grafik data, dan *card* ketiga menampilkan riwayat pencatatan. Pada gambar 18 dan Gambar 19 Tampilan dari *dashboard monitoring* oven dispatch:



Gambar 18. Tampilan *Dashboard* 1.



Gambar 19. Tampilan *Dashboard* 2.

Pada gambar 20 dan 21 Node.js berperan krusial sebagai lapisan tengah (middleware) dalam sistem pemantauan oven ini. Dengan menempatkan Node.js di antara frontend React.js dan backend Firebase, kita dapat meningkatkan keamanan data, melakukan validasi data secara lebih terperinci, dan mengabstraksi kunci API Firebase. Dengan demikian, data yang tersimpan di Firebase menjadi lebih terlindungi dari akses yang tidak sah dan sistem secara keseluruhan menjadi lebih aman dan reliabel. Seperti pada gambar 20, merupakan modul yang digunakan untuk server dan Pada gambar 21 menunjukkan program dependencies yang digunakan untuk membuat dashboard:

```
1  const express = require('express');
2  const bcrypt = require('bcryptjs');
3  const cors = require('cors');
4  const axios = require('axios');
5  const { Server } = require('socket.io');
6  const http = require('http');
7  require('dotenv').config();
8  const jwt = require('jsonwebtoken');
```

Gambar 20. Modul Server.

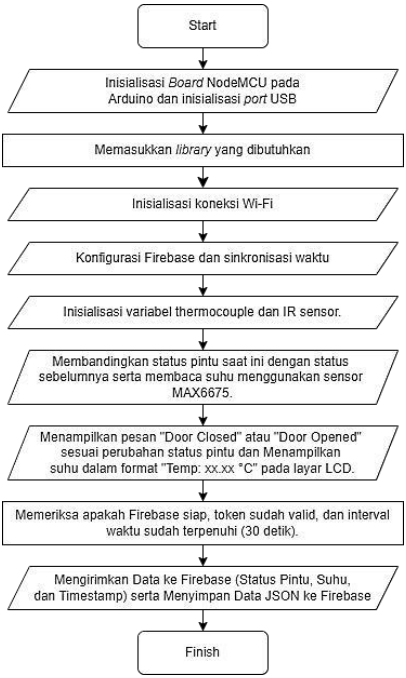
```
"use client"

import * as React from "react"
import { format, addMonths} from "date-fns"
import { io } from "socket.io-client"
import { Check, ChevronsUpDown, CalendarIcon } from "lucide-react"
import { Button } from "@/components/ui/button"
import { Calendar } from "@/components/ui/calendar"
import {Command, CommandInput, CommandEmpty, CommandGroup, CommandItem, CommandList,} from "@/components/ui/command"
import {Popover, PopoverContent, PopoverTrigger,} from "@/components/ui/popover"
import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@/components/ui/table"
import { Line } from 'react-chartjs-2'
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  Title,
  Tooltip,
  Legend,
} from "chart.js"
import { Card, CardHeader, CardTitle, CardContent, CardFooter } from "@/components/ui/card"
import axios from 'axios';
```

Gambar 21. Dependencies.

Urutan langkah kerja aplikasi ini adalah: Masuk ke chrome dan akses http://localhost:3000 setelah di *run* melalu VScode dan *server* juga dijalankan. Tampilan *Sign in* akan muncul sebelum memasuki tampilan *dashboard*. Setelah *user Sign in,* maka tampilan *dashboard* muncul. Data akan muncul secara otomatis. Tombol *Update PM* berfungsi untuk memperbarui data tanggal dilakukannya *maintenance* dan tombol *Print Chart* berfungsi untuk mencetak grafik ke format *file* pdf. Pada tombol *Sign in* yang dapat dipilih oleh *user* untuk keluar dari tampilan *dashboard*.

3.1.3.2. Pemrograman Arduino IDE

Pada gambar 22 merupakan perencanaan pemrograman akan memberikan kemudahan dalam menentukan struktur pemrograman. Struktur pemrograman alat tugas akhir ini dirancang menggunakan Arduino IDE *(Integrated Development Environment)*. Gambar 22 merupakan flowchart pemrograman pada Arduino IDE:



Gambar 22. Flowchart pemrograman alat Tugas Akhir.

Sebelum melakukan pemrograman, *board* pada Arduino IDE diatur menjadi ESP32 Arduino (ESP32 Dev *Module*) karena mikrokontroler yang digunakan pada alat ini adalah NodeMCU ESP32. Pada baris awal pemrograman perlu menambahkan *library-library* yang diperlukan dari Modul MAX6675 dan LCD agar bisa berkomunikasi dengan ESP8266. Selain itu pada proyek ini memerlukan *library* tambahan agar ESP8266 bisa berkomunikasi dengan *database* server.

Gambar 23 merupakan *library* yang digunakan pada pemrograman ini:

```
#include <Arduino.h>
#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include <LiquidCrystal_I2C.h>
#include <max6675.h>
#include <Wire.h>
#include "time.h"

#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"
```

Gambar 23. Pemrograman *library*.

Berikut ini merupakan fungsi dari *library* yang kami gunakan:

1. **Arduino.h**
   Library utama Arduino untuk fungsi dasar seperti *pinMode, digitalRead, digitalWrite,* dll.
2. **WiFi.h**
   Library untuk koneksi WiFi pada ESP32 seperti menghubungkan perangkat ke jaringan dan mendapatkan alamat IP.
3. **Firebase_ESP_Client.h**
   Library untuk integrasi dengan *Firebase Realtime Database* yang memungkinkan pengiriman dan pengambilan data dari *Firebase Realtime Database. Library* ini mendukung otentikasi menggunakan konfigurasi FirebaseAuth dan FirebaseConfig, serta memiliki kemampuan untuk memperbarui token secara otomatis melalui *callback*.
4. **LiquidCrystal_I2C.h**
   Library untuk mengontrol LCD berbasis komunikasi I2C yang memungkinkan menampilkan teks, mengatur posisi kursor, dan mengontrol *backlight* LCD.
5. **MAX6675.h**
   Library untuk membaca data dari sensor suhu MAX6675 yang dirancang khusus untuk bekerja dengan termokopel tipe K, memberikan pembacaan suhu dalam derajat Celsius.
6. **Wire.h**
   Library ini digunakan untuk komunikasi NodeMCU dengan I2C devices
7. **time.h**
   Library untuk mendapatkan waktu lokal menggunakan NTP yang mengambil waktu lokal dari server NTP untuk digunakan dalam logging data.
8. **#include "addons/TokenHelper.h" dan #include "addons/RTDBHelper.h"**
   File tambahan untuk mendukung pengelolaan token autentikasi Firebase dan mempermudah operasi pada *Realtime Database*.

Pada gambar 24 program pengolahan dan menampilkan di LCD data MAX6675 yang ada pada checkTemp(); serta kondisi pintu yang ada pada doorCond() diletakkan pada function utama yaitu void loop(). Pada gambar 24 merupakan tampilan program dari Gambar function void loop():

```
void loop() {
  sendData();
  doorCond();
  checkTemp();
}
```

Gambar 24. Function void loop.

Lalu, pada gambar 25 function sendData() berfungsi untuk mengirimkan data sensor secara periodik ke *Firebase Realtime Database* dalam aplikasi IoT. Fungsi ini memastikan pengiriman hanya terjadi jika koneksi Firebase siap, autentikasi berhasil, dan interval waktu telah terpenuhi. Data yang dikirim mencakup status pintu, suhu dari sensor thermocouple, dan timestamp yang diperoleh dari waktu lokal. Informasi tersebut dikirim baik secara individual maupun dalam format JSON ke *database Firebase* pada *path* yang ditentukan. Pada gambar 25 merupakan tampilan program dari function sendData():

```
106 void sendData() {
107   if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 30000 || sendDataPrevMillis == 0)){
108     sendDataPrevMillis = millis();
109
110     struct tm timeinfo;
111     if(!getLocalTime(&timeinfo)){
112       Serial.println("Failed to obtain time");
113       return;
114     }
115
116     String doorStatus = (currentState == HIGH) ? "Open" : "Close";
117
118     if (Firebase.RTDB.setString(&fbdo, "Data/Door Condition", doorStatus)){
119       Serial.println("PASSED");
120       Serial.println("PATH: " + fbdo.dataPath());
121       Serial.println("TYPE: " + fbdo.dataType());
122     }
123     else {
124       Serial.println("FAILED");
125       Serial.println("REASON: " + fbdo.errorReason());
126     }
127
128     float temperature = thermocouple.readCelsius();
129     float calTemp = temperature+cal;
130     if (Firebase.RTDB.setFloat(&fbdo, "Data/Temperature", calTemp)){
131       Serial.println("PASSED");
132       Serial.println("PATH: " + fbdo.dataPath());
133       Serial.println("TYPE: " + fbdo.dataType());
134     }
135     else {
```

```
    Serial.println("FAILED");
    Serial.println("REASON: " + fbdo.errorReason());
  }

  char Time[20];
  strftime(Time,20, "%F %T", &timeinfo);
  if (Firebase.RTDB.setString(&fbdo, "Data/Timestamp", Time)){
    Serial.println("PASSED");
    Serial.println("PATH: " + fbdo.dataPath());
    Serial.println("TYPE: " + fbdo.dataType());
  }
  else {
    Serial.println("FAILED");
    Serial.println("REASON: " + fbdo.errorReason());
  }
  timestamp = getTime();

  parentPath= databasePath + "/" + String(timestamp);
  json.set(doorPath.c_str(), doorStatus);
  json.set(tempPath.c_str(),String (calTemp));
  json.set(timePath.c_str(), Time);
  Serial.printf("Set json... %s\n", Firebase.RTDB.setJSON(&fbdo, parentPath.c_str(), &json) ? "ok" : fbdo.errorReason().c_str());
 }
}
```

Gambar 25. Function sendData.

Dan terakhir pada gambar 26 function getTime() digunakan untuk mendapatkan timestamp dan memastikan sinkronisasi waktu dengan NTP. Pada gambar 26 merupakan tampilan program dari function getTime():

```
unsigned long getTime() {
  time_t now;
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    return(0);
  }
  time(&now);
  return now;
}
```

Gambar 26. Function getTime.

3.2. Alat, Bahan dan Lokasi

Tabel 4 merupakan rincian biaya pengerjaan proyek Tugas Akhir ini, peneliti memerlukan alat dan bahan serta lokasi pengambilan data. Lokasi pengerjaan serta pengambilan *prototype* Tugas Akhir dilakukan di PT Ark Engineering. Untuk pengerjaan mekanik dari *box cover* memerlukan jasa tenaga wiraswasta. Namun untuk perakitan semua *part* tetap dilakukan di PT Ark Engineering. Pada tabel 5 merupakan apa saja yang dibutuhkan dalam membuat alat, merinci pada komponen yang di perlukan:

Tabel 5. Alat dan Bahan.

No	Nama Barang		Jumlah	Harga	Total
1	Bahan	IR Infrared	2 Pcs	Rp15,000	Rp30,000
2		MAX6675 K-type Thermocouple	2 Pcs	Rp70,000	Rp140,000
3		ESP32 Devkit V1	2 Pcs	Rp100,000	Rp200,000
4		Buzzer	2 Pcs	Rp5,000	Rp10,000
5		LCD 16x2	2 Pcs	Rp40,000	Rp80,000
6		Cover Box Elektronik	2 Pcs	Rp265,000	Rp530,000
7		Buku UI/UX	1 Pcs	Rp135,000	Rp270,000
8		File SNI IEC 60529:2014	1 Pcs	Rp75,000	Rp150,000
9		Cable Jumper F-F 20cm	4 Pcs	Rp15,000	Rp60,000
10		LM2596 - DC Converter	2 Pcs	Rp12,000	Rp24,000
11		Epoxy + Resin	1 kg	Rp220,000	Rp220,000
12		T-block PCB	4 Pcs	Rp1,500	Rp6,000
13		Dioda IN4007	6 Pcs	Rp500	Rp3,000
14		I2C LCD	2 Pcs	Rp15,000	Rp30,000
15	Alat	Solder + Timah	1 Set	Rp -	Rp -
16		Laptop	2 Set	Rp -	Rp -
13		Cable data type C	1 Set	Rp -	Rp -
14		1 Set Obeng	1 Set	Rp -	Rp -
Total Keseluruhan					Rp1,753,000

3.3. Kalibrasi *Thermocouple*

Kalibrasi *thermocouple* dilakukan dengan membandingkan nilai pengukuran suhu yang dihasilkan oleh termokopel dengan suhu yang diukur oleh perangkat pengukur suhu yang sudah terkalibrasi pada oven (termometer digital). Tujuannya adalah untuk mendapatkan pembacaan suhu yang akurat. Untuk mendapatkan faktor kalibrasi, kami melakukan pengambilan data yang dihasilkan dan mencatat pembacaan suhu dari termokopel dan alat pengukur suhu terkalibrasi secara bersamaan.

Pada gambar 27 data suhu yang dihasilkan oleh termokopel dapat dipantau melalui serial monitor pada Arduino IDE menggunakan kode program untuk mendapatkan *raw value* pembacaan *thermocouple* yang telah dikonversi oleh MAX6675. Gambar 27 adalah program untuk membaca nilai suhu dari *thermocouple* menggunakan modul MAX6675:

```
#include "max6675.h"

int thermoDO = 19;
int thermoCS = 23;
int thermoCLK = 5;

MAX6675 thermocouple(thermoCLK, thermoCS, thermoDO);

void setup() {
  Serial.begin(9600);

  Serial.println("MAX6675 test");
  // wait for MAX chip to stabilize
  delay(500);
}

void loop() {
  // basic readout test, just print the current temp

  Serial.print("C = ");
  Serial.println(thermocouple.readCelsius());

  // For the MAX6675 to update, you must delay AT LEAST 250ms between reads!
  delay(5000);
}
```

Gambar 27. Kode program untuk mendapatkan *raw Value.*

Tabel 6 berikut ini merupakan nilai keluaran terhadap pembacaan suhu pada oven yang telah diketahui:

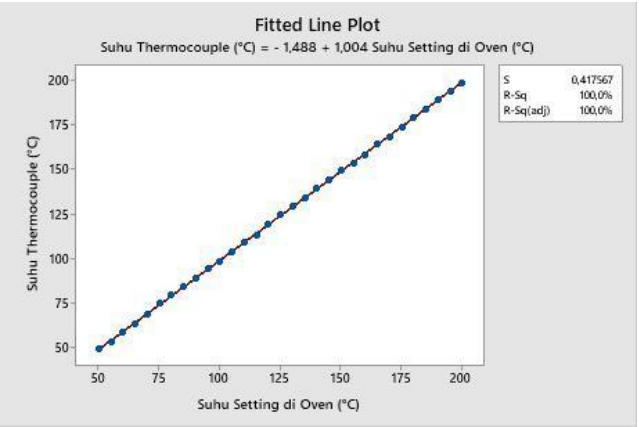
Tabel 6. Pembacaan Suhu.

Suhu Setting di Oven (°C)	Suhu Thermocouple (°C)	Selisih (°C)
50	48.7	1.3
55	53.1	1.9
60	58.3	1.7
65	63.1	1.9
70	68.9	1.1
75	74.5	1.5
80	79.3	0.7
85	84.2	0.8
90	88.6	1.4
95	94.3	0.7
100	98.5	1.5
105	103.9	1.1
110	109.2	0.8
115	113.4	1.6
120	119.1	0.9
125	124.6	1.4
130	129.2	0.8
135	134.5	0.7
140	139.3	0.7
145	144	1.1
150	149.4	0.6
155	153.7	1.3
160	158.1	1.3
165	164.3	0.7
170	168.8	1.2

175	174.2	0.8
180	179.5	0.5
185	184.1	0.9
190	189.4	0.6
195	194.2	0.8
200	199	1

Untuk mengevaluasi akurasi pengukuran suhu yang dilakukan oleh termokopel dengan membandingkannya terhadap pengaturan suhu oven. Dengan menggunakan model regresi linier, analisis ini bertujuan untuk menentukan hubungan matematis antara suhu yang diukur dan suhu yang diatur, serta memastikan bahwa termokopel memberikan pembacaan yang konsisten dan akurat dalam rentang suhu yang diuji. Berdasarkan jenis data yang tersebar pada grafik kami mengetahui bahwa Pada grafik di atas, ditampilkan hubungan antara suhu yang diukur oleh termokopel (°C) dan pengaturan suhu oven (°C).

Pada gambar 28 model regresi linier yang ditampilkan memiliki persamaan Suhu Thermocouple (°C) = - 1,488 + 1,004 Suhu Setting di Oven (°C) Nilai R-squared yang mencapai 100% menunjukkan bahwa model ini menjelaskan semua variasi data, menunjukkan kesesuaian yang sangat baik antara pengukuran suhu dengan pengaturan oven. Titik-titik yang tersebar di sekitar garis regresi menunjukkan konsistensi hasil pengukuran. Gambar 28 menjelaskan jenis data yang tersebar kami mengetahui bahwa pada grafik, ditampilkan hubungan antara suhu yang diukur oleh termokopel (°C) dan pengaturan suhu oven (°C) pada Tabel 6:



Gambar 28. Suhu yang diukur oleh termokopel (°C) dan pengaturan suhu oven (°C).

Pada gambar 29 setelah memperoleh persamaan regresi nilai b, yang merupakan koefisien regresi, dapat digunakan sebagai faktor kalibrasi untuk MAX6675 dengan mengintegrasikannya ke dalam pemograman. Ketika ESP32 booting, maka nilai dari faktor kalibrasi akan diset terlebih dahulu. Sehingga nantinya hasil pengukuran suhu dari *thermocouple* sudah menunjukkan suhu yang sebenarnya. Pada gambar 29 merupakan bentuk dari pemogramannya:

```
182 void checkTemp(){
183   float temperature = thermocouple.readCelsius();
184   float calTemp = temperature * 1.004 - 1.488;
```

Gambar 29. Faktor kalibrasi pada sintak pemrograman.

3.4. Pengujian Alat

Pengujian alat dimulai dari pengoperasian cara kerja alat Tugas Akhir, pengiriman data dan hingga memonitoring hasil pemabacaan suhu.

3.4.1. Cara kerja alat

Hasil yang diharapkan dari skenario pengujian cara kerja alat adalah alat Tugas Akhir bisa mengukur suhu dari oven despatch dan kondisi pintu-nya. Berikut ini Langkah-langkah pengujian cara kerja alat :

1. Pertama, aktifkan alat dengan menghubungkan adaptor ke stop kontak. LCD akan menampilkan tulisan "*Loading...Please wait*" yang menandakan ESP32 sedang menginisialisasi koneksi WiFi dan koneksi ke Firebase Realtime Database. Pada gambar 30 Alat sedang melakukan inisialisasi jaringan:

Gambar 30. Alat sedang melakukan inisialisasi jaringan.

2. Pada gambar 31 setelah ESP32 terkoneksi pada jaringan WiFi yang sudah ditentukan dan terkoneksi dengan database, maka LCD akan menampilkan tulisan "*connected*!". Pada gambar 31 ESP32 terkoneksi pada jaringan WiFi:

Gambar 31. ESP32 terkoneksi pada jaringan WiFi.

3. Pada gambar 32 menampilkan tulisan *"connected!"*, maka LCD akan menampilkan data suhu dan kondisi pintu. Suhu akan berubah sesuai data dari modul MAX6675 dan kondisi pintu tertutup jika sensor IR mengenai/mendeteksi pintu dari oven dengan menampilkan di LCD tulisan "*Door Closed"*. Pada gambar 32 menampilkan data suhu dan kondisi pintu tutup:

Gambar 32. Menampilkan data suhu dan kondisi pintu tutup.

4. Pada gambar 33 jika kondisi pintu sedang terbuka atau sensor IR tidak mendeteksi keberadaan dari pintu maka LCD menampilkan tulisan "*Door Opened*". Data suhu tersebut serta kondisi pintu akan langsung terkirim ke database. Pada gambar 33 menampilkan data suhu dan kondisi pintu buka:

Gambar 33. Menampilkan data suhu dan kondisi pintu buka.

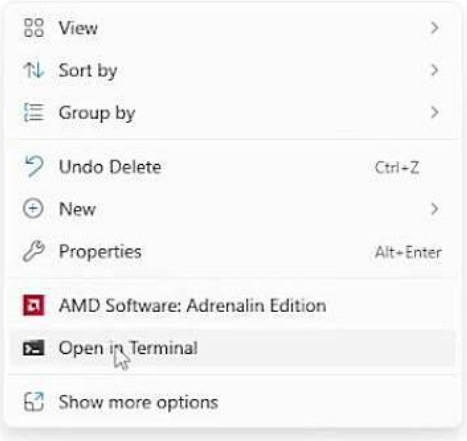
3.4.2. Pengujian antarmuka pengguna (GUI)

Hasil yang diharapkan dari skenario pengujian GUI adalah data pengukuran suhu dan kondisi pintu dapat dipantau melalui GUI serta dapat dianalisa. Berikut ini langkah-langkah dalam monitoring data pengukuran suhu dan kondisi pintu dari oven despatch:

1. Pada gambar 34 membuka ikon *folder oven-app* dengan klik dua kali pada ikon *folder* di *desktop*. Pada gambar 34 merupakan tampilan ikon folder pada *desktop*:

Gambar 34. Ikon folder.

2. Pada gambar 35 setelah folder terbuka, klik kanan lalu klik Open in Terminal. Pada gambar 35 merupakan tampilan setelah klik kanan pada folder:



Gambar 35. Tampilan klik kanan pada folder.

3. Terlihat pada gambar 36 setelah terminal terbuka, ketik node server/server.js dan tekan *enter* untuk menjalankan *server* dari *website monitoring*. Tulisan *"Client Connected"* berarti *server* sudah terhubung ke *database firebase*. Pada gambar 36 merupakan tampilan pada *desktop* bahwa *server* sudah terhubung ke *database firebase*:

Gambar 36. Server terhubung ke *database firebase*.

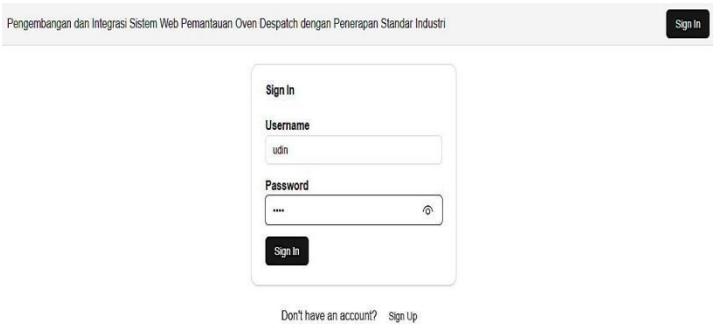
4. Pada tampilan gambar 37 klik + untuk menambah terminal lagi. Untuk mengarahkan terminal ke directory darioven-app, ketik cd "C:\Users\riska\OneDrive\Desktop\oven-app". Lalu, ketik npm run dev dan tekan enter untuk menjalankan Next.js Development. Gambar 37 merupakan tampilan untuk menjalankan Next.js Development:

Gambar 37. Menjalankan *Next.js Development*.

5. Lalu, dapat dilihat pada gambar 38 untuk membuka website dengan cara menekan ctrl dan klik http://localhost:3000. Gambar 38 merupakan tampilan untuk membuka website:

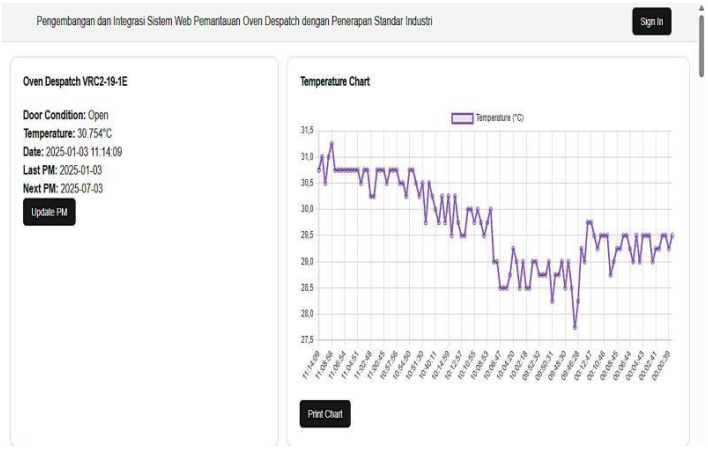
Gambar 38. Tampilan untuk membuka *website*.

6. Pada gambar 39 *browser* bawaan laptop akan membuka *tab* baru yang berisi *website* dari *monitoring*. Selanjutnya memasukkan *Username*: udin dan *Password*: udin pada *Sign in page*. Gambar 39 merupakan tampilan untuk *Sign In*:



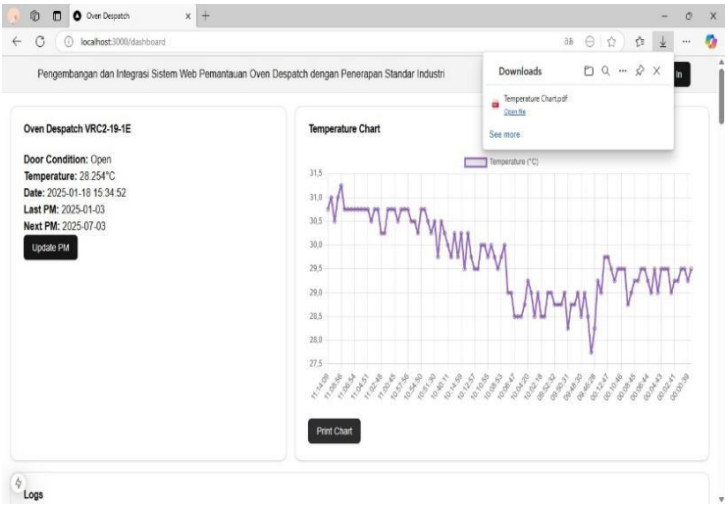
Gambar 39. Tampilan untuk *Sign In*.

7. Pada gambar 40 ketika *user* sudah *Sign in* maka akan muncul *dashboard page* yang langsung menampilkan data *realtime* dari oven, grafik data, serta riwayat pencatatan terakhir. *User* bisa memantau hasil pengukuran dari *website* ini. *User* dapat menekan tombol *Update PM* jika sudah melakukan *Preventive Maintenance* di hari tersebut dan menekan tombol *Print Chart* untuk mengekspor grafik riwayat pengukuran ke format Pdf. Gambar 40 merupakan tampilan data *realtime*:

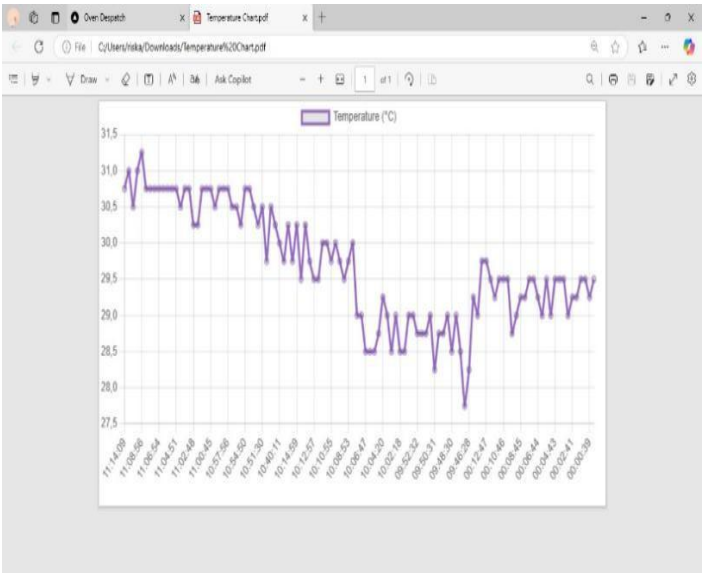


Gambar 40. Data *realtime*.

8. Selanjutnya pada gambar 41 dan 42 ketika *user* menekan tombol *Print Chart*, browser secara otomatis menghasilkan file PDF dari grafik suhu yang ditampilkan di halaman. File PDF tersebut kemudian langsung diunduh ke perangkat *user* dengan nama "Temperature Chart.pdf". Pada gambar 41 menunjukkan proses mengunduh file oleh *browser* dan pada gambar 42 tampilan grafik pada PDF yang terunduh:

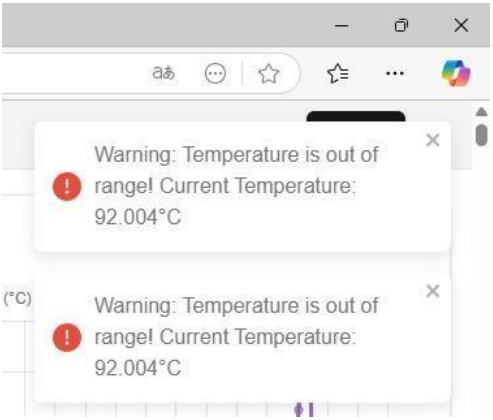


Gambar 41. Tampilan proses mengunduh *file*.



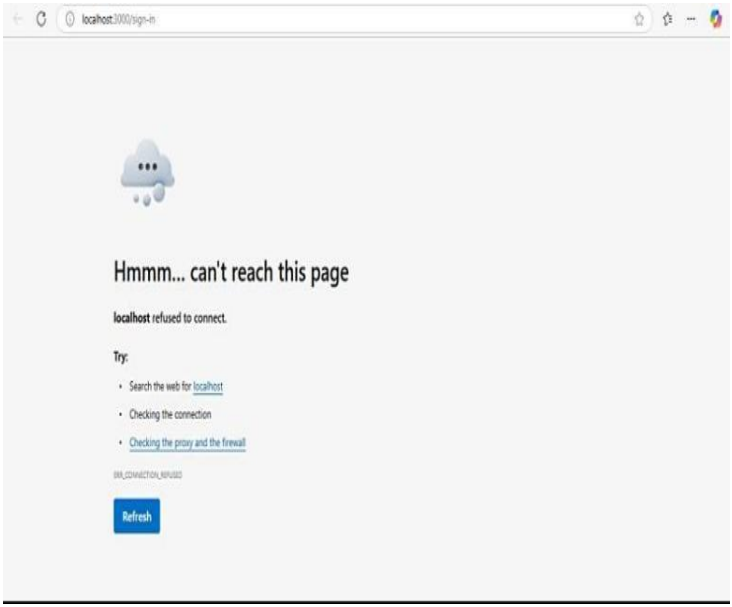
Gambar 42. Tampilan grafik pada PDF.

9. Pada gambar 43 *web* akan menampilkan notifikasi ketika suhu berada di luar rentang yang ditentukan, yaitu di bawah 140°C atau di atas 160°C. Notifikasi ini bertujuan untuk memberikan peringatan kepada pengguna agar dapat mengambil tindakan yang diperlukan untuk menjaga kondisi operasional yang aman. Pada gambar 43 ditampilkan notifikasi pada *website*:



Gambar 43. Tampilan notifikasi.

10. Terlihat pada gambar 44 merupakan tampilan *user* telah selesai memantau melalui *website* dan keluar dari *dashboard page* dengan klik tombol *Sign In* untuk kembali ke *Sign In page*. Untuk memberhentikan *server* dan Next.js Development, buka terminal *server* dan Development lalu tekan ctrl+C atau tekan tanda silang pada kanan atas terminal untuk memberhentikan serv*er* serta memberhentikan *website*. Pada gambar 44 maka *website* akan menunjukkan tampilan *website* ketika berhenti:



Gambar 44. Tampilan *website* ketika berhenti.

Bab 4. Analisa dan Pembahasan

4.1. Analisa

4.1.1. Hasil Kalibrasi Termokople

Parameter keberhasilan pengukuran massa beban pada alat tugas akhir ini dapat diketahui dengan membandingkan hasil pengukuran suhu yang dihasilkan oleh termokopel dengan suhu yang diukur oleh perangkat pengukur suhu yang sudah terkalibrasi pada oven (termometer digital). Pertama, konsistensi antara pembacaan suhu yang dihasilkan oleh termokople dan alat pengukur suhu terkalibrasi harus menunjukkan deviasi yang minimal, idealnya kurang dari satu derajat Celsius, untuk memastikan akurasi. Kedua, grafik hubungan antara nilai suhu termokople dan termometer digital harus menunjukkan pola linear, menandakan bahwa faktor kalibrasi yang ditentukan dapat diterapkan secara universal pada rentang suhu yang diuji.

Selain itu, pengulangan pengukuran pada berbagai titik suhu harus menghasilkan hasil yang konsisten, sementara pengujian ketahanan termokopel dalam jangka waktu tertentu juga penting untuk memastikan bahwa kinerjanya tetap stabil. Maka dari itu penulis melakukan pengambilan data perbandingan pengukuran suhu yang dihasilkan oleh perangkat pengukur suhu yang sudah terkalibrasi pada oven (termometer digital) dan alat tugas akhir yang dapat dilihat pada Tabel 7:

Tabel 7. Pembacaan suhu hasil pengujian pengukuran suhu setelah *thermocouple* terkalibrasi.

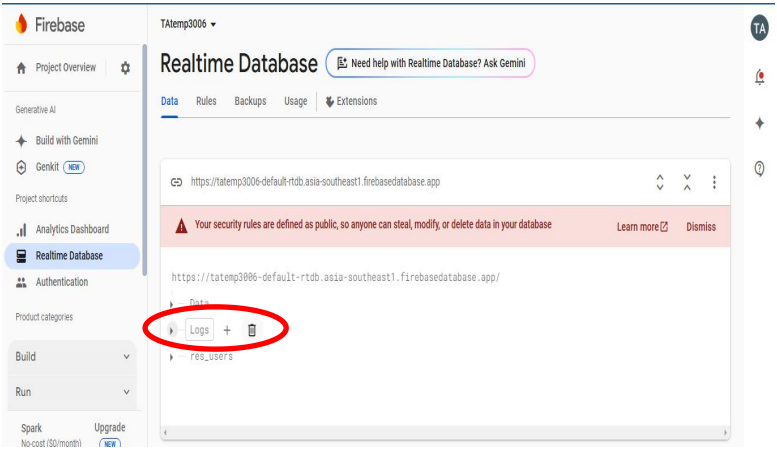
Suhu Setting di Oven (°C)	Suhu Thermocouple (°C)	Selisih (°C)
50	49.135	0.865
55	55.281	-0.281
60	59.393	0.607
65	64.093	0.907
70	70.094	-0.094
75	74.089	0.911
80	79.481	0.519
85	84.174	0.826
90	90.267	-0.267
95	94.282	0.718

100	100.153	-0.153
105	104.148	0.852
110	110.03	-0.03
115	114.481	0.519
120	120.126	-0.126
125	124.425	0.575
130	129.013	0.987
135	134.47	0.53
140	139.322	0.678
145	144.395	0.605
150	150.444	-0.444
155	154.137	0.863
160	159.174	0.826
165	164.288	0.712
170	170.215	-0.215
175	175.321	-0.321
180	179.329	0.671
185	184.13	0.87
190	189.065	0.935
195	194.162	0.838
200	200.115	-0.115

Berdasarkan data yang diperoleh pada Tabel 4.1, terdapat perbedaan dari hasil pengukuran suhu. Selisih pengukuran menunjukkan bahwa pengukuran suhu pada thermocouple memiliki perbedaan dengan suhu setting di oven. Dari data pengukuran, dapat disimpulkan bahwa selisih antara suhu setting di oven dan pembacaan suhu dari termokople berkisar antara ±0.5°C hingga ±1°C. Penulis menyadari bahwa error yang terjadi mungkin diakibatkan oleh ketidakstabilan suhu dalam oven atau karakteristik respons termokople yang tidak sempurna. Selain itu, faktor seperti posisi termokople yang mungkin tidak optimal dalam oven juga berkontribusi terhadap variabilitas ini. Meskipun demikian, hasil pengujian menunjukkan bahwa setelah proses kalibrasi dilakukan, batasan desain aplikasi monitoring suhu dengan ketelitian yang dapat diandalkan dalam rentang ±1°C telah berhasil dicapai, memberikan keyakinan bahwa sistem pengukuran berfungsi dengan baik dalam konteks aplikasi yang diinginkan.

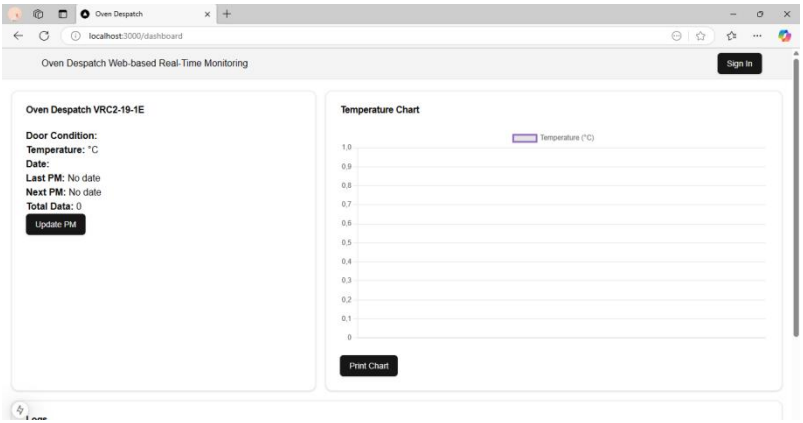
4.1.2. Pengujian Konektivitas IoT

Dalam metode pengujian ini, pada gambar 45 diampilkan langkah pertama yang dilakukan adalah menghapus seluruh data dari Logs, sehingga kondisi awal menjadi kosong. Ini penting untuk memastikan bahwa tidak ada data sebelumnya yang mempengaruhi hasil pengujian.



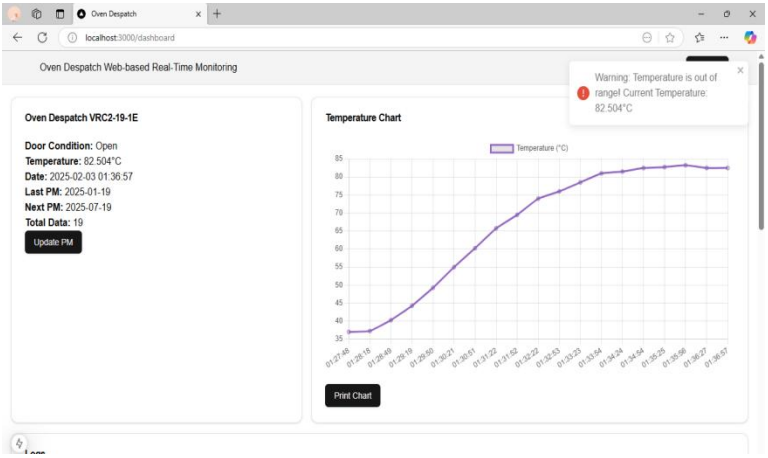
Gambar 45. Menghapus Logs data.

Setelah itu, Pada gambar 46 alat yang diuji dalam keadaan terhubung dengan Wi-Fi, namun laptop yang digunakan untuk memantau tidak terhubung ke jaringan yang sama.



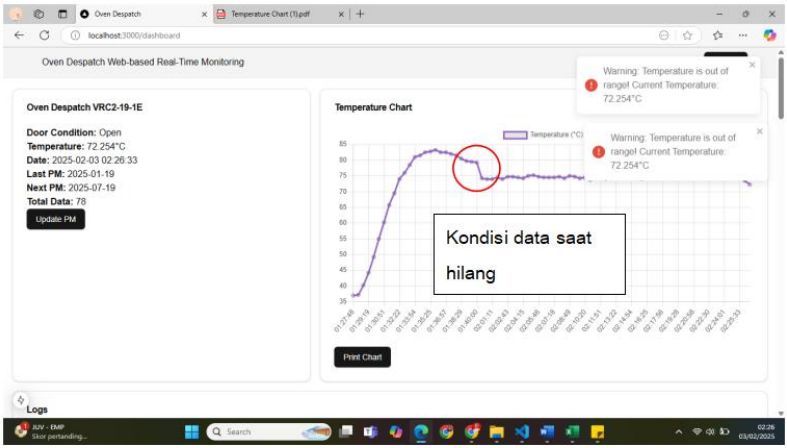
Gambar 46. Tampilan pada Laptop 1.

Pada gambar 47 Setelah laptop terhubung ke Wi-Fi, data dari alat masih ada dikarenakan *Fetch* data menggunakan Metode GET. Metode ini digunakan untuk mengambil data dari *server*. Ini adalah metode baca-saja, jadi tidak ada risiko data berubah atau rusak.



Gambar 47. Tampilan pada Laptop 2.

Pada gambar 48 diketahui ketika alat tidak terhubung ke Wi-Fi. Dalam kondisi ini, data yang dikumpulkan oleh alat tidak dapat dikirim, yang berpotensi menyebabkan kehilangan informasi penting. Meskipun demikian, setelah alat terhubung kembali ke Wi-Fi dalam waktu 10 detik, alat tersebut secara otomatis mengirimkan data yang telah terkumpul sebelumnya ke *Firebase*. Proses ini menunjukkan kemampuan alat untuk memulihkan dan mengirimkan data secara efisien setelah terputus dari jaringan, memberikan jaminan bahwa informasi yang dikumpulkan tetap bisa disimpan dan diakses meskipun terjadi gangguan konektivitas.



Gambar 48. Tampilan alat tidak terhubung Wi-Fi.

Dengan demikian, metode pengujian ini tidak hanya menilai kemampuan alat dalam mengumpulkan data, tetapi juga mengevaluasi ketahanan dan efektivitas sistem dalam mengelola data selama situasi konektivitas yang tidak stabil.

4.2. Pembahasan

4.2.1. *Cover*

Pada proyek sistem pemantauan oven, perlindungan komponen merupakan aspek penting untuk memastikan ketahanan alat terhadap kondisi lingkungan operasional yang menantang. Cover harus memiliki ketentuan tingkat perlindungan atau *Ingress Protection* (IP) untuk melindungi komponen elektronik dari debu dan air. Dalam konteks ini, bahan yang digunakan untuk 3D printing harus memiliki sifat mekanik dan termal yang memadai untuk menghadapi suhu operasional oven yang tinggi.

Selain *cover* 3D printed, metode epoxy potting digunakan untuk memberikan lapisan perlindungan tambahan pada komponen sensitif seperti modul elektronik dan konektor. Epoxy potting tidak hanya melindungi komponen dari debu dan air tetapi juga meningkatkan ketahanan terhadap getaran, benturan, dan fluktuasi suhu. Dalam proyek ini, penggunaan epoxy potting berfungsi memastikan keandalan perangkat ketika terpapar panas dari oven atau kelembapan lingkungan.

Dapat dilihat pada gambar 45 gabungan penggunaan *cover* 3D printed dan epoxy potting memungkinkan pengoptimalan perlindungan komponen dalam proyek ini. Dengan ini dapat dipastikan bahwa alat tidak hanya aman untuk digunakan dalam kondisi operasional biasa tetapi juga mampu bertahan dalam kondisi ekstrem seperti paparan suhu tinggi dan kelembapan. Pendekatan ini memberikan kombinasi perlindungan struktural dan fungsional, memastikan umur pakai perangkat yang lebih panjang, dan meningkatkan keselamatan serta keandalan alat secara keseluruhan. Pada gambar 45 merupakan tampilan dari pengaplikasian epoxy resin:

Gambar 49. Epoxy resin elektrikal.

4.2.2. Node.js

Node.js berperan sebagai middleware penting dalam sistem pemantauan oven yang menghubungkan *frontend* berbasis React.js dengan backend firebase *realtime database*. Dalam perannya ini, Node.js berfungsi untuk mengabstraksi kunci API *firebase*, sehingga mencegah akses langsung dari klien dan meningkatkan keamanan data. Node.js juga bertugas memvalidasi data yang masuk dari frontend, memastikan hanya data valid yang diteruskan ke *firebase* untuk disimpan.

Selain itu, *middleware* ini mengelola lalu lintas data antara *frontend* dan *backend*, menciptakan sinkronisasi *real-time* yang stabil sehingga pengguna dapat menikmati pengalaman antarmuka yang responsif. Sebagai server lokal, Node.js menjalankan perintah seperti node server/server.js untuk menerima permintaan dari *frontend*, memproses data, dan mengirimkannya ke *firebase* atau mengembalikan respons untuk ditampilkan di antarmuka pengguna. Dengan sifat asinkron dan non-blokir, Node.js sangat cocok untuk aplikasi *real-time* seperti sistem *monitoring* oven ini. Untuk meningkatkan fungsi *middleware*, *framework* seperti Express.js dapat digunakan untuk menyederhanakan pengelolaan rute dan *middleware* lainnya. Fitur tambahan seperti Socket.IO juga dapat diintegrasikan untuk mendukung komunikasi *real-time* yang lebih andal, misalnya untuk notifikasi perubahan data. Pada gambar 46 merupakan tampilan dari logo Node.js:

Gambar 50. Logo Node.js.

Node.js juga memungkinkan implementasi *middleware* kustom yang dapat digunakan untuk menambahkan fitur seperti log aktivitas pengguna, pembatasan kecepatan permintaan (*rate limiting*), atau *caching data*. Dengan menggunakan pendekatan ini, sistem *monitoring* oven dapat menjadi lebih aman, reliabel, dan efisien.

Bab 5. Kesimpulan dan Saran

5.1. Kesimpulan

Dari hasil perancangan dan pengujian sistem pemantauan *Oven Despatch* yang telah dilakukan, dapat disimpulkan bahwa Tugas Akhir ini berhasil mencapai tujuan yang telah ditetapkan. Kesimpulan dari Tugas Akhir ini adalah sebagai berikut:

1. Sistem pemantauan berbasis web untuk *Oven Despatch* berhasil dirancang dan diimplementasikan, yang memungkinkan pemantauan suhu dan status pintu oven secara *real-time* dari jarak jauh.
2. Sistem ini menyediakan fitur yang memungkinkan pengguna untuk menyimpan data grafik suhu secara langsung dalam format PDF, memudahkan dokumentasi dan tindak lanjut.
3. Desain Antarmuka Pengguna (GUI) dalam sistem pemantauan Oven Despatch telah dikembangkan dengan pendekatan yang intuitif, memastikan aksesibilitas bagi pengguna dengan berbagai latar belakang teknis. Dengan penyajian data suhu dan status pintu yang jelas, pengguna dapat dengan cepat mengakses informasi penting, yang mendukung pengambilan keputusan yang tepat dan respons yang lebih cepat terhadap kondisi operasional.

5.2. Saran

Saran dari penulis untuk pengembangan lebih lanjut sistem pemantauan ini adalah:

1. Melakukan penelitian lebih lanjut untuk mengeksplorasi teknologi baru dan tren dalam IoT yang dapat diterapkan untuk meningkatkan sistem pemantauan, termasuk penggunaan kecerdasan buatan (AI).
2. Melakukan analisis lebih lanjut untuk meningkatkan keandalan alat dengan menambahkan fungsi datalogger yang mencatat data uptime dan implementasi error handling. Hal ini diharapkan dapat membantu mengidentifikasi potensi masalah dan memberikan masukan untuk pengembangan lebih lanjut.

Daftar Pustaka

[1] A. Gotmare and S. Bokade, “Internet of Things in Manufacturing : A Review on Applications, Challenges and Future Directions Internet of Things in Manufacturing: A Review on Applications, Challenges and Future Directions,” 2019. [Online]. Available: https://www.researchgate.net/publication/336995388

[2] Y. P. Kondratenko, O. V. Kozlov, O. V. Korobko, and A. M. Topalov, “Internet of Things approach for automation of the complex industrial systems,” *CEUR Workshop Proc.*, vol. 1844, pp. 3–18, 2017.

[3] C. Fadhilah, Ariyan Zubaidi, and Ahmad Zafrullah Mardiansyah, “A Design of Early Fire Detection System In Tobacco Oven Based on Internet of Things (Case Study: East Landah Praya Village),” *J. Comput. Sci. Informatics Eng.*, vol. 7, no. 1, 2023, doi: 10.29303/jcosine.v7i1.473.

[4] F. Hashim, R. Mohamad, M. Kassim, S. I. Suliman, N. M. Anas, and A. Z. A. Bakar, “Implementation of embedded real-time monitoring temperature and humidity system,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 16, no. 1, pp. 184–190, 2019, doi: 10.11591/ijeecs.v16.i1.pp184-190.

[5] S. Purwanti, A. Febriani, Mardeni, and Y. Irawan, “Temperature monitoring system for egg incubators using raspberry Pi3 based on internet of things (IoT),” *J. Robot. Control*, vol. 2, no. 5, pp. 349–352, 2021, doi: 10.18196/jrc.25105.

[6] D. Adirinarso, “No Titleیب ل ی,” *Nucl. Phys.*, vol. 13, no. 1, pp. 104–116, 2023.

[7] E. Hariadi, Y. Anistyasari, M. S. Zuhrie, and R. E. Putra, “Mesin Oven Pengering Cerdas Berbasis Internet of Things (IoT),” *Indones. J. Eng. Technol.*, vol. 2, no. 1, pp. 18–23, 2022, doi: 10.26740/inajet.v2n1.p18-23.

[8] A. A. Jaber, F. K. I. Al-Mousawi, and H. S. Jasem, “Internet of things based industrial environment monitoring and control: A design approach,” *Int. J. Electr. Comput. Eng.*, vol. 9, no. 6, pp. 4657–4667, 2019, doi: 10.11591/ijece.v9i6.pp4657-4667.

[9] H. A. Wahid, J. Maulindar, and A. I. Pradana, “Rancang Bangun Sistem Penyiraman Tanaman Otomatis Aglonema Berbasis IoT Menggunakan Blynk dan NodeMCU 32,” *Innov. J. Soc. Sci. Res.*, vol. 3, no. 2, pp. 6265–6276, 2023.

[10] K. Kusmiyati *et al.*, “Monitoring Sistem Kontrol Mesin Drying Kopi Secara Real Time Berbasis IoT,” *Elektrika*, vol. 15, no. 2, p. 90, 2023, doi: 10.26623/elektrika.v15i2.7857.

[11] “Last minute,” *Schweizerische Ärztezeitung*, vol. 81, no. 21. pp. 1125–1125, 2000. doi: 10.4414/saez.2000.07334.

[12] Y. Wishnu, P. Prayudha, S. Muhammad, and S. Novianto, “Rancang Bangun Sistem Pengukuran Alat Thermobath sebagai Alat Kalibrasi Temperatur dengan Sistem Arduino Uno Design and Manufacture a Thermobath Measurement System as a Temperature Calibration Tool with Arduino Uno Sistem System,” vol. 4, pp. 25–34, 2022.

[13] A. T. S, P. Hari babu, and R. Divya Sree, “IR Sensor Based Obstacle Detection and Avoiding Robot PJAEE, 17 (9) (2020) IR Sensor Based Obstacle Detection and Avoiding Robot,” vol. 17, no. 9, pp. 3328–3336, 2020.

[14] “circuit digest,” p. 20.

[15] M. B. Nendya, B. Susanto, G. I. W. Tamtama, and T. J. Wijaya, “Desain Level Berbasis Storyboard Pada Perancangan Game Edukasi Augmented Reality Tap The Trash,” *Fountain Informatics J.*, vol. 8, no. 1, pp. 1–6, 2023, doi: 10.21111/fij.v8i1.8836.

[16] Normah and F. Sihaloho, “Perancangan User Interface (UI) dan User Experince (UX) Aplikasi Pendistribution Alat-alat Kesehatan pada PT. Rekamileniumindo Selaras Jakarta Barat,” *Indones. J. Softw. Eng.*, vol. 9, no. 1, pp. 33–38, 2023.

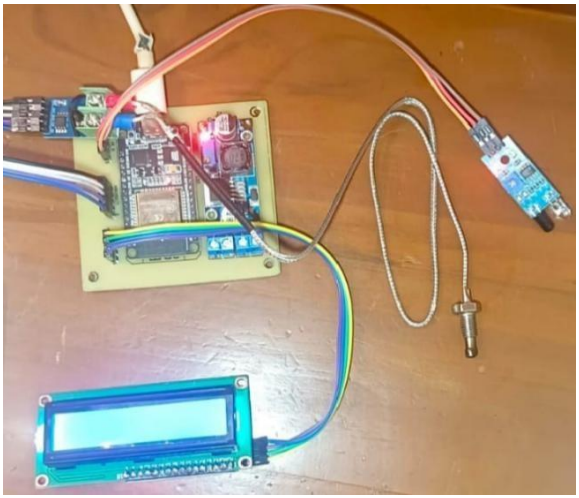
Biodata

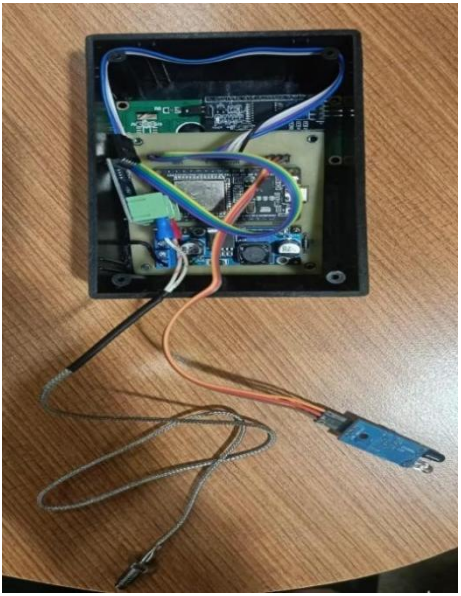
Nama	: Zidan Gunawan Kusumah
TTL	: Bandung, 6 November 2001
Agama	: Islam
Alamat	: Eden Park Blok F no 10,Batam Center,Batam
Email	: zidanwgk07@gmail.com
Riwayat Pendidikan	: SMPN 003 (2014 - 2017) SMKN 1 BATAM (2017 - 2020)

Nama	: Mochammad Nuruddin Mustaqim
TTL	: Madiun, 30 Oktober 2000
Agama	: Islam
Alamat	: Legenda Malaka Blok E1 no 12,Batam Center,Batam
Email	: Udinmustaqim11@gmail.com
Riwayat Pendidikan	: SMPN 12 BATAM (2013 - 2016) SMKN 1 BATAM (2016 - 2019)

Lampiran A

Instalasi elektrikal, antar komponen dihubungkan dengan *wire jumper* .





Lampiran B

Program pada Arduino IDE.

```
#include <Arduino.h>
#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include <LiquidCrystal_I2C.h>
#include <max6675.h>
#include <Wire.h>
#include "time.h"

#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"

#define WIFI_SSID "YUDAIN"
#define WIFI_PASSWORD "12345678"
#define API_KEY "AIzaSyCEsWUXlwJMjAoQOlTaGD7pB0Ob293I-kU"
#define DATABASE_URL "https://tatemp3006-default-rtdb.asia-
southeast1.firebasedatabase.app/"

const char* ntpServer = "pool.ntp.org";
const long  gmtOffset_sec = 25200;
const int   daylightOffset_sec = 0;

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

String databasePath;
String doorPath = "/Door Condition";
String tempPath = "/Temperature";
String timePath = "/Timestamp";
String parentPath;
```

```
int timestamp;
FirebaseJson json;

unsigned long sendDataPrevMillis = 0;
bool signupOK = false;

#define SENSOR_PIN 34
int lastState = HIGH;
int currentState;

int lcdColumns = 16;
int lcdRows = 2;

int thermoDO = 19;
int thermoCS = 23;
int thermoCLK = 5;

LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);
MAX6675 thermocouple(thermoCLK, thermoCS, thermoDO);

unsigned long getTime()
  { time_t now;
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo))
    { return(0);
  }
  time(&now);
  return now;
}

void setup()
  { Serial.begin(115200);
  lcd.init();
  lcd.backlight();
```

```
  lcd.setCursor(0, 0);
  lcd.print("Loading...");
  lcd.setCursor(0, 1);
  lcd.print("Please Wait");
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);

  config.api_key = API_KEY;

  config.database_url = DATABASE_URL;

  if (Firebase.signUp(&config, &auth, "", ""))
    { Serial.println("ok");
    signupOK = true;
  }
  else {
    Serial.printf("%s\n",
config.signer.signupError.message.c_str());
  }
  lcd.clear();
  lcd.print("Connected!");
  config.token_status_callback = tokenStatusCallback; //see
addons/TokenHelper.h
```

```
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

  pinMode(SENSOR_PIN, INPUT);
  databasePath = "/Logs";
  delay(500);
}

void sendData() {
  if (Firebase.ready() && signupOK && (millis() -
sendDataPrevMillis > 30000 || sendDataPrevMillis == 0)){
    sendDataPrevMillis = millis();

    struct tm timeinfo;
    if(!getLocalTime(&timeinfo)){
      Serial.println("Failed to obtain time");
      return;
    }

    String doorStatus = (currentState == HIGH) ? "Open" : "Close";

    if (Firebase.RTDB.setString(&fbdo, "Data/Door Condition",
doorStatus)){
      Serial.println("PASSED");
      Serial.println("PATH: " + fbdo.dataPath());
      Serial.println("TYPE: " + fbdo.dataType());
    }
    else {
      Serial.println("FAILED"); Serial.println("REASON:
      " + fbdo.errorReason());
    }

    float temperature = thermocouple.readCelsius();
    float calTemp = temperature * 1.004 - 1.488;
```

```
    if (Firebase.RTDB.setFloat(&fbdo, "Data/Temperature",
calTemp)){
      Serial.println("PASSED");
      Serial.println("PATH: " + fbdo.dataPath());
      Serial.println("TYPE: " + fbdo.dataType());
    }
    else {
      Serial.println("FAILED"); Serial.println("REASON:
      " + fbdo.errorReason());
    }

    char Time[20];
    strftime(Time,20, "%F %T", &timeinfo);
    if (Firebase.RTDB.setString(&fbdo, "Data/Timestamp",
      Time)){ Serial.println("PASSED");
      Serial.println("PATH: " + fbdo.dataPath());
      Serial.println("TYPE: " + fbdo.dataType());
    }
    else {
      Serial.println("FAILED"); Serial.println("REASON:
      " + fbdo.errorReason());
    }
    timestamp = getTime();

    parentPath= databasePath + "/" + String(timestamp);
    json.set(doorPath.c_str(), doorStatus);
    json.set(tempPath.c_str(),String (calTemp));
    json.set(timePath.c_str(), Time);
    Serial.printf("Set json... %s\n", Firebase.RTDB.setJSON(&fbdo,
parentPath.c_str(), &json) ? "ok" : fbdo.errorReason().c_str());
  }
}

void loop() {
```

```
  sendData();
  doorCond();
  checkTemp();
}

void doorCond(){
  currentState = digitalRead(SENSOR_PIN);

  if (lastState == HIGH && currentState ==
    LOW){ lcd.setCursor(0, 1);
    lcd.print("Door Closed");
    }
  else if (lastState == LOW && currentState ==
    HIGH){ lcd.setCursor(0, 1);
    lcd.print("Door Opened");
    }

  delay(50);
  lastState = currentState;
  }

void checkTemp(){
  float temperature = thermocouple.readCelsius();
  float calTemp = temperature * 1.004 - 1.488;
  char buffer[7];
  dtostrf(calTemp, 6, 2, buffer);

  lcd.setCursor(0, 0);
  lcd.print("Temp : ");
  lcd.print(buffer);
  lcd.print(" ");
  lcd.print((char)223);
  lcd.print("C");
```

```
delay(1000);
}
```

Lampiran C

1. Program pada *Interface Login*.

```
'use client'

import { useState } from 'react'
import { useRouter } from 'next/navigation'
import { Input } from '@/components/ui/input'
import { Button } from '@/components/ui/button'
import { Alert, AlertDescription, AlertTitle } from
'@/components/ui/alert'
import { Card, CardHeader, CardTitle, CardContent } from
'@/components/ui/card'

const SignInPage = () => {
  const [username, setUsername] = useState('')
  const [password, setPassword] = useState('')
  const [error, setError] = useState('')
  const router = useRouter()

  const handleSubmit = async (e: React.FormEvent) =>
    { e.preventDefault()

    setError('')

    if (!username || !password)
      { setError('Please fill in both fields.')
      return
    }

    try {
```

```
      const response = await
fetch('http://localhost:5000/login', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ username, password }),
      })

      const data = await response.json()

      if (response.ok)
        { router.push('/dashboard')
      } else {
        setError(data.error || 'Invalid username or password.')
      }
    } catch (err) {
      setError('An error occurred during login.')
    }
  }

  return (
    <div className="login-container">
      <Card>
        <CardHeader>
          <CardTitle>Sign In</CardTitle>
        </CardHeader>
        <CardContent>
          <form onSubmit={handleSubmit}>
            <div className="form-group">
              <label htmlFor="username">Username</label>
              <Input
                id="username"
                value={username}
```

```
                onChange={(e) => setUsername(e.target.value)}
                placeholder="Enter your username"
                required
              />
            </div>

            <div className="form-group">
              <label htmlFor="password">Password</label>
              <Input
                id="password"
                type="password"
                value={password}
                onChange={(e) => setPassword(e.target.value)}
                placeholder="Enter your password"
                required
              />
            </div>

            {error && (
              <Alert variant="destructive" className="error-
message">
                <AlertTitle>Error</AlertTitle>
                <AlertDescription>{error}</AlertDescription>
              </Alert>
            )}

            <Button type="submit" className="submit-btn">
              Sign In
            </Button>
          </form>
        </CardContent>
      </Card>

      <div className="sign-up-container">
```

```
        <p>
          Don't have an account?{' '}
          <Button
            variant="link"
            className="sign-up-btn"
            onClick={() => router.push('/sign-up')}
          >
            Sign Up
          </Button>
        </p>
      </div>

      <style jsx>{`
        .login-container
          { max-width:
          400px; margin: 0
          auto; padding:
          20px;
        }

        .form-group {
          margin-bottom: 15px;
        }

        label {
          display: block;
          font-weight: bold;
        }

        .submit-btn { width:
          100%; padding:
          10px; margin-top:
          10px;
          background-color: #0070f3;
```

```
        color: white;
```

```
          border: none;
          border-radius: 4px;
          cursor: pointer;
        }

        .submit-btn:hover
          { background-color:
          #005bb5;
        }

        .sign-up-container
          { margin-top: 20px;
          text-align: center;
        }

        .sign-up-btn
          { color: #0070f3;
          text-decoration: underline;
          cursor: pointer;
        }

        .sign-up-btn:hover
          { color: #005bb5;
        }

        .error-message
          { margin-top:
          10px;
        }
      `}</style>
    </div>
  )
}
```

```
export default SignInPage
```

2. Program *Interface Dashboard*.

```
"use client"

import * as React from "react"
import { format, addMonths} from "date-fns"
import { io } from "socket.io-client"
import { Check, ChevronsUpDown, CalendarIcon } from "lucide-react"
import { Button } from "@/components/ui/button"
import { Calendar } from "@/components/ui/calendar"
import {Command, CommandInput, CommandEmpty, CommandGroup,
CommandItem, CommandList,} from "@/components/ui/command"
import {Popover, PopoverContent, PopoverTrigger,} from
"@/components/ui/popover"
import { Table, TableBody, TableCell, TableHead, TableHeader,
TableRow } from "@/components/ui/table"
import { Line } from 'react-chartjs-2'
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  Title,
  Tooltip,
  Legend,
} from "chart.js"
import { Card, CardHeader, CardTitle, CardContent, CardFooter } from
"@/components/ui/card"
import axios from 'axios';
import { config } from "process"

ChartJS.register(CategoryScale, LinearScale, PointElement,
LineElement, Title, Tooltip, Legend)
```

```
const socket = io("http://localhost:5000")

export default function Dashboard() {
  const [logsData, setLogsData] = React.useState<any | null>(null)
  const [realtimeData, setRealtimeData] = React.useState<any |
null>(null)
  const [selectedFilter, setSelectedFilter] = React.useState("No
Filter")
  const [startDate, setStartDate] = React.useState<Date | null>(null)
  const [endDate, setEndDate] = React.useState<Date | null>(null)
  const [PM, setPM] = React.useState<string | null>(null);

  React.useEffect(() =>
    { socket.on('data', (data) => {
      console.log('Received data:', data);
      setLogsData(data.logsData)
      setRealtimeData(data.realtimeData)
    })

    socket.emit('get_data')

    const interval = setInterval(() =>
      {          console.log('Requesting
      data...'); socket.emit('get_data')
    }, 5000)

    return () =>
      { socket.off('data')
      clearInterval(interval)
    }
  }, [])

  const handleUpdatePM = async() => {
    const today = format(new Date(), "yyyy-MM-dd");
```

```
    setPM(today);

    try {
      console.log("Sending PM value:", { PM: today });
      const response = await
axios.patch('http://localhost:5000/write-data', { PM: today });
      console.log("PM update saved to Firebase:", response.data);
  } catch (error) {
      console.error("Error saving PM update:", error);
  }
  };

  const filterLogsByDate = (logsData: any, filter: string,
startDate?: Date, endDate?: Date) => {
    const currentDate = new Date();
    const filteredLogs: any[] = [];

    switch (filter)
      { case "Today":
        logsData.forEach((log: any) => {
          const logDate = new Date(log.Timestamp);
          if (
            logDate.getDate() === currentDate.getDate() &&
            logDate.getMonth() === currentDate.getMonth() &&
            logDate.getFullYear() === currentDate.getFullYear()
          ) {
            filteredLogs.push(log);
          }
        });
        break;
      case "Custom Filter":
        logsData.forEach((log: any) => {
          const logDate = new Date(log.Timestamp);
          if (startDate && endDate) {
```

```
            if (logDate >= startDate && logDate <= endDate)
              { filteredLogs.push(log);
            }
          }
        });
        break;
      default:
        return logsData;
    }

    return filteredLogs.sort((a, b) => new
Date(b.Timestamp).getTime() - new Date(a.Timestamp).getTime());
};

  const filteredLogs = logsData ?
filterLogsByDate(Object.values(logsData), selectedFilter, startDate,
endDate) : []

  const filterOptions = [
    { value: "Today", label: "Today" },
    { value: "Custom Filter", label: "Custom Filter" },
    { value: "No Filter", label: "No Filter" },
  ]

  const chartData = {
    labels: filteredLogs.map((log: any) => format(new
Date(log.Timestamp), "HH:mm:ss")),
    datasets: [
      {
        label: 'Temperature (°C)',
        data: filteredLogs.map((log: any) => log["Temperature"]),
        fill: false,
        borderColor: 'rgb(147, 105, 194)',
        tension: 0.1,
```

```
      }
    ]
  }

  return (
    <div className="space-y-4 p-4">
      <div className="flex gap-4">
        <Card className="w-1/2">
          <CardHeader>
            <CardTitle>Oven Despatch VRC2-19-1E</CardTitle>
          </CardHeader>
          <CardContent>
            {realtimeData ? (
              <div>
                <p><strong>Door Condition:</strong>
{realtimeData["Door Condition"]}</p>
                <p><strong>Temperature:</strong>
{realtimeData["Temperature"]}°C</p>
                <p><strong>Date:</strong>
{realtimeData.Timestamp}</p>
                <p><strong>Last PM:</strong> {realtimeData["PM"] ?
format(new Date(realtimeData["PM"]), "yyyy-MM-dd") : "No date"}</p>
                <p><strong>Next PM:</strong> {realtimeData["PM"] ?
format(addMonths(new Date(realtimeData["PM"]),6), "yyyy-MM-dd") :
"No date"}</p>
                <Button onClick={handleUpdatePM}>Update PM</Button>
              </div>
            ) : (
              <p>Loading real-time data...</p>
            )}
          </CardContent>
        </Card>

        <Card className="w-3/4">
```

```
            <CardHeader>
              <CardTitle>Temperature Chart</CardTitle>
            </CardHeader>
            <CardContent>
              <div className="h-96">
                <Line data={chartData} />
              </div>
              <Button>Print Chart</Button>
            </CardContent>
          </Card>
        </div>

        <Card>
          <CardHeader>
            <CardTitle>Logs</CardTitle>
          </CardHeader>

          <CardContent>
            <Popover>
              <PopoverTrigger asChild>
                <Button
                  variant="outline"
                  className="w-[200px] justify-between mb-4"
                >
                  {filterOptions.find(option => option.value ===
selectedFilter)?.label || "Select Filter"}
                  <ChevronsUpDown className="opacity-50" />
                </Button>
              </PopoverTrigger>
              <PopoverContent className="w-[200px] p-0">
                <Command>
                  <CommandInput placeholder="Search filter..." />
                  <CommandList>
                    <CommandEmpty>No filter found.</CommandEmpty>
```

```
                    <CommandGroup>
                      {filterOptions.map((filter) => (
                        <CommandItem
                          key={filter.value}
                          onSelect={() => {
                            setSelectedFilter(filter.value)
                          }}
                        >
                          {filter.label}
                          <Check
                            className={`ml-auto ${selectedFilter ===
filter.value ? "opacity-100" : "opacity-0"}`}
                          />
                        </CommandItem>
                      ))}
                    </CommandGroup>
                  </CommandList>
                </Command>
              </PopoverContent>
            </Popover>

            {selectedFilter === "Custom Filter" && (
              <div className="mt-4 space-x-4">
                <Popover>
                  <PopoverTrigger asChild>
                    <Button
                      variant={"outline"}
                      className="w-[280px] justify-start text-left
font-normal"
                    >
                      <CalendarIcon />
                      {startDate ? format(startDate, "PPP") :
<span>Pick a start date</span>}
                    </Button>
```

```
                </PopoverTrigger>
                <PopoverContent className="w-auto p-0">
                  <Calendar
                    mode="single"
                    selected={startDate}
                    onSelect={setStartDate}
                    initialFocus
                  />
                </PopoverContent>
              </Popover>
              <Popover>
                <PopoverTrigger asChild>
                  <Button
                    variant={"outline"}
                    className="w-[280px] justify-start text-left
font-normal"
                  >
                    <CalendarIcon />
                    {endDate ? format(endDate, "PPP") : <span>Pick
an end date</span>}
                  </Button>
                </PopoverTrigger>
                <PopoverContent className="w-auto p-0">
                  <Calendar
                    mode="single"
                    selected={endDate}
                    onSelect={setEndDate}
                    initialFocus
                  />
                </PopoverContent>
              </Popover>
            </div>
          )}
```

```
            <Table>
              <TableHeader>
                <TableRow>
                  <TableHead>Date</TableHead>
                  <TableHead>Door Condition</TableHead>
                  <TableHead>Temperature (°C)</TableHead>
                </TableRow>
              </TableHeader>
              <TableBody>
                {filteredLogs.length ?
                  ( filteredLogs.map((log, index) => (
                    <TableRow key={index}>
                      <TableCell>{log.Timestamp ? format(new
Date(log.Timestamp), "dd/MM/yyyy HH:mm:ss") : "No
timestamp"}</TableCell>
                      <TableCell>{log["Door Condition"]}</TableCell>
                      <TableCell>{log["Temperature"]}°C</TableCell>
                    </TableRow>
                  ))
                ) : (
                  <TableRow>
                    <TableCell colSpan={3} className="text-center">No
logs available.</TableCell>
                  </TableRow>
                )}
              </TableBody>
            </Table>
          </CardContent>
        </Card>
      </div>
    )

}
```

Lampiran D

Program *Server* Node.Js.

```
const express = require('express');
const bcrypt = require('bcryptjs');
const cors = require('cors');
const axios = require('axios');
const { Server } = require('socket.io');
const http = require('http');
require('dotenv').config();
const jwt = require('jsonwebtoken');

const app = express();
const server = http.createServer(app);
const io = new Server(server, {
    cors: {
        origin: "*",
    },
});

const API_KEY = 'AIzaSyCEsWUXlwJMjAoQOlTaGD7pB0Ob293I-kU';
const DB_URL = 'https://tatemp3006-default-rtdb.asia-
southeast1.firebasedatabase.app/';

app.use(cors());
app.use(express.json());

const fetchData = async () =>
    { try {
        const logsUrl = `${DB_URL}/Logs.json?auth=${API_KEY}`;
        const dataUrl = `${DB_URL}/Data.json?auth=${API_KEY}`;

        const logsResponse = await axios.get(logsUrl);
        const dataResponse = await axios.get(dataUrl);
```

```
        return {
            logsData: logsResponse.data || {},
            realtimeData: dataResponse.data || {}
        };
    } catch (error) {
        console.error(`Error fetching data: ${error}`);
        return { logsData: {}, realtimeData: {} };
    }
};

app.get('/fetch-data', async (req, res) =>
    { const data = await fetchData();
    res.json(data);
});

io.on('connection', (socket) =>
    { console.log("Client connected");

    const sendData = async () =>
        { const data = await
        fetchData(); socket.emit('data',
        data);
    };

    sendData();

    socket.on('get_data', () =>
        { sendData();
    });

    socket.on('disconnect', () =>
        { console.log("Client disconnected");
});
```

```
    });

});
```

```
app.post('/register', async (req, res) => {
    const { username, password, confirmPassword } = req.body;

    if (!username || !password || !confirmPassword) {
        return res.status(400).json({ error: 'Please provide
username, password, and confirm password.' });
    }

    if (password !== confirmPassword) {
        return res.status(400).json({ error: 'Passwords do not
match.' });
    }

    try {
        const hashedPassword = await bcrypt.hash(password, 10);

        const user = { username:
            username,
            password: hashedPassword,
        };

        const userRef =
`${DB_URL}/res_users.json?auth=${API_KEY}`;
        const response = await axios.post(userRef, user);

        res.status(200).json({ message: 'User registered
successfully!' });
    } catch (error) {
        console.error('Error registering user:', error);
        res.status(500).json({ error: 'Error registering
user.' });
    }
});
```

```
app.post('/login', async (req, res) =>
    { const { username, password } = req.body;

    if (!username || !password) {
        return res.status(400).json({ error: 'Please provide
username and password.' });
    }

    try {
        const userRef =
`${DB_URL}/res_users.json?auth=${API_KEY}`;
        const usersResponse = await axios.get(userRef);
        const users = usersResponse.data;

        const user = Object.values(users).find(user =>
user.username === username);

        if (!user) {
            return res.status(400).json({ error: 'Invalid
username or password.' });
        }

        const match = await bcrypt.compare(password,
user.password);

        if (match) {
            // const token = jwt.sign({ username: user.username,
id: user.id }, JWT_SECRET, { expiresIn: '1h' });

            res.status(200).json({
                message: 'Login successful',
                // token: token
            });
```

```
        } else {
            res.status(400).json({ error: 'Invalid username or
password.' });
        }
    } catch (error) {
        console.error('Error logging in:', error);
        res.status(500).json({ error: 'An error occurred during
login.' });
    }
});

const verifyToken = (req, res, next) => {
    const token = req.header('Authorization')?.split(' ')[1];

    if (!token) {
        return res.status(403).json({ error: 'No token
provided.' });
    }

    try {
        const decoded = jwt.verify(token, JWT_SECRET);
        req.user = decoded;
        next();
    } catch (error) {
        res.status(401).json({ error: 'Invalid or expired
token.' });
    }
};

app.get('/protected', verifyToken, (req, res) =>
    { res.json({ message: 'This is a protected route.',
    user:
req.user });
});
```

```
app.patch('/write-data', async (req, res) => {
    const {PM} = req.body; // Get the lastPM value from the
request body

    if (!PM) {
        return res.status(400).json({ error: 'No lastPM data
provided.' });
    }

    try {
        const writeUrl = `${DB_URL}/Data.json?auth=${API_KEY}`;
// Adjust the path as needed to store the lastPM value
        const response = await axios.patch(writeUrl, {PM}); //
Write data to Firebase

        res.status(200).json({ message: 'Last PM written
successfully!', data: response.data });
    } catch (error) {
        console.error('Error writing lastPM to Firebase:', error);
        res.status(500).json({ error: 'Error writing data to
Firebase.' });
    }
});

const PORT = 5000;
server.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
});
```

Lampiran E

Package/Dependencies yang digunakan.

```
{
  "name": "oven-app",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev --turbopack",
    "build": "next build", "start":
    "next start",
    "lint": "next lint"
  },
  "dependencies":
    { "@hookform/resolvers": "^3.9.1",
    "@radix-ui/react-dialog": "^1.1.4",
    "@radix-ui/react-label": "^2.1.1",
    "@radix-ui/react-popover": "^1.1.4",
    "@radix-ui/react-slot": "^1.1.1",
    "@radix-ui/react-toast": "^1.2.4",
    "@tanstack/react-table": "^8.20.6",
    "axios": "^1.7.9",
    "bcryptjs": "^2.4.3",
    "chart.js": "^4.4.7",
    "class-variance-authority": "^0.7.1",
    "clsx": "^2.1.1",
    "cmdk": "^1.0.0",
    "date-fns": "^4.1.0",
    "dotenv": "^16.4.7",
    "express": "^4.21.2",
    "firebase": "^11.1.0",
    "jsonwebtoken": "^9.0.2",
    "jwt-decode": "^4.0.0",
    "lucide-react": "^0.469.0",
```

```
    "next": "15.1.2",
    "next-auth": "^4.24.11",
    "react": "^19.0.0",
    "react-chartjs-2": "^5.2.0",
    "react-csv": "^2.2.2",
    "react-day-picker": "^8.10.1",
    "react-dom": "^19.0.0",
    "react-hook-form": "^7.54.2",
    "socket.io": "^4.8.1",
    "socket.io-client": "^4.8.1",
    "tailwind-merge": "^2.5.5",
    "tailwindcss-animate": "^1.0.7",
    "uuid": "^11.0.3",
    "zod": "^3.24.1"
  },
  "devDependencies":
    { "@eslint/eslintrc": "^3",
    "@types/node": "^20",
    "@types/react": "^19",
    "@types/react-dom": "^19",
    "eslint": "^9",
    "eslint-config-next": "15.1.2",
    "postcss": "^8",
    "tailwindcss": "^3.4.1",
    "typescript": "^5"
  }

}
```

Lampiran F

Tampilan alat yang sudah jadi: