Sistem Informasi Pembacaan Angka Meter PDAM dengan Memanfaatkan ESP32-CAM dan *Optical Character Recognition*

1st Rachmad Sukri
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
rachmadsukri@gmail.com

2nd Endro Ariyanto
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
endroa@telkomuniversity.ac.id

3rd Fazmah Arif Yuhanto
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
fazmaharif@telkomuniversity.ac.id

***Abstrak* — Pencatatan angka meteran air pada Perusahaan Daerah Air Minum (PDAM) selama ini masih dilakukan secara manual dengan petugas mendatangi rumah pelanggan, sehingga dinilai tidak efisien dan memakan banyak waktu. Penelitian ini bertujuan merancang serta membangun sistem otomatisasi pembacaan meter air untuk mengatasi permasalahan tersebut. Metode penelitian yang digunakan adalah *Research and Development*(R&D). Sistem dikembangkan dengan memanfaatkan modul ESP32-CAM untuk mengambil gambar meteran secara periodik dan mengirimkannya ke server backend berbasis Laravel. Selanjutnya, proses *Optical Character Recognition*(OCR) dilakukan menggunakan script Python yang terhubung dengan Application Programming Interface (API) Gemini untuk mengonversi gambar menjadi data numerik. Data hasil pembacaan disimpan dalam database MySQL dan ditampilkan pada platform website berbasis Next.js, sehingga pengguna maupun admin dapat melakukan pemantauan secara real-time. Hasil penelitian menunjukkan bahwa perangkat keras berbasis ESP32-CAM berhasil diintegrasikan dengan platform server dan menghasilkan akurasi pembacaan mencapai 100%. Namun, kelemahan sistem masih terdapat pada desain fisik yang rentan bergeser dan terpengaruh gangguan visual seperti kotoran atau pantulan cahaya. Untuk pengembangan selanjutnya, disarankan peningkatan stabilitas perangkat, integrasi mekanisme pembersihan otomatis, penggunaan protokol komunikasi real-time, serta pemanfaatan panel surya untuk meningkatkan portabilitas dan kemandirian energi.**

Kata kunci*— *OCR*, ESP 32-CAM, *Gemini*, *Meter Air*, *Internet of Things (IoT)

I. PENDAHULUAN

Air merupakan kebutuhan pokok yang sangat penting bagi manusia untuk menjaga kelangsungan hidup [1]. Setiap harinya, masyarakat menggunakan air untuk berbagai keperluan, seperti memasak, mandi, mencuci, dan untuk pertanian. Namun, penggunaan air yang berlebihan dapat memiliki dampak negatif terhadap lingkungan dan ketersediaan air di masa depan. Seiring dengan bertambahnya jumlah penduduk setiap tahunnya, sumber daya air yang dapat digunakan juga semakin terbatas [1]. Indonesia memiliki dua iklim yaitu musim penghujan dan musim kemarau dimana pada musim kemarau debit air sangat berpotensi mengalami penurunan debit air yang signifikan. Seperti pada kasus di daerah Lombok Barat, Nusa Tenggara Barat tahun 2023 pada musim kemarau debit air menurun kurang lebih 5-10 persen dari kondisi normal [2]. Untuk antisipasi tidak terjadi krisis air, perlu menjaga dan melestarikan sumber air yang ada, efisiensi dalam penggunaan air serta pencarian alternatif sumber baru.

Monitoring penggunaan air saat ini masih banyak dilakukan secara manual, misalnya dengan mendatangi rumah pengguna untuk mengambil foto meteran air. Cara ini tidak hanya memakan waktu dan tenaga, tetapi juga kurang efisien dalam pengelolaan data. [1]. Kelemahan dari metode pemantauan penggunaan air dengan flow meter adalah diperlukannya penggantian meteran air yang sudah ada, yang mana proses tersebut memerlukan biaya dan tenaga tambahan. Pada penelitian sebelumnya yang menggunakan metode *Run-Length Smearing Algorithm* (RLSA) yang dapat membaca angka pemakaian pelanggan pada meteran air untuk mengatasi keterbatasan ini, dengan memungkinkan pelanggan memantau penggunaan air yang mereka butuhkan [3].

Untuk mengatasi tantangan tersebut, penggunaan teknologi menjadi solusi yang tepat. Dengan memanfaatkan teknologi ESP32-CAM dan image processing, pembacaan meteran air dapat dilakukan dengan lebih mudah dan efisien. ESP32 CAM adalah modul mikrokontroler yang dilengkapi dengan kamera, sehingga memungkinkan untuk diakses pada jarak jauh dan pengambilan gambar meteran air bisa dilakukan secara otomatis [4]. Proses pengambilan gambar ini dapat diprogram untuk dilakukan secara periodik atau sesuai dengan kebutuhan monitoring. *Optical Character Recognition*(OCR) dapat digunakan untuk

mengkonversi dokumen cetak atau tulisan tangan hasil scan menjadi karakter ASCII (karakter yang dapat dibaca oleh mesin) [3]. Untuk memaksimalkan hasil OCR, maka digunakan *Run Length Smearing Algorithm* (RLSA) untuk menemukan lokasi teks. Data yang terkumpul dari gambar meteran air dapat diolah secara otomatis untuk mengidentifikasi dan mencatat penggunaan air dengan akurasi tinggi. Data yang dihasilkan merupakan hasil model *machine learning* dari data yang sudah ditraining berasal dari sekumpulan data yang sudah ada sebelumnya.

II. KAJIAN TEORI

A. Perusahaan Daerah Air Minum (PDAM)

PDAM adalah salah satu unit usaha milik daerah yang bergerak di bidang distribusi air bersih bagi masyarakat umum. PDAM merupakan Perusahaan daerah yang diawasi dan dimonitor oleh aparat eksekutif maupun legislatif. Lokasi PDAM terdapat di setiap provinsi, kabupaten, dan kotamadya di seluruh Indonesia. Awal mula PDAM dimulai sejak zaman penjajahan Belanda pada tahun 1920 an dengan nama Waterleiding, sedangkan pada zaman pendudukan Jepang perusahaan air minum dinamai Suido Syo. Namun semenjak Indonesia merdeka, sebagian besar pengelolaan air harus berada di bawah pemerintah. Hal ini diatur dalam Pasal 33 ayat (3) Undang Undang Dasar Negara Republik Indonesia Tahun 1945 [5].

B. *Optical Character Recognition*(OCR)

Optical Character Recognition(OCR) adalah sistem yang dapat mengenali dan mengonversi huruf-huruf dari dokumen cetak, baik dari pencetak (printer atau mesin ketik) maupun tulisan tangan, menjadi teks yang dapat dibaca oleh komputer. OCR bekerja dengan cara membandingkan pola karakter per-baris dengan pola yang telah disimpan dalam database aplikasi, kemudian menerjemahkan gambar karakter menjadi teks [6].

Penerapan OCR memungkinkan komputer untuk melakukan berbagai proses lebih lanjut, seperti translasi ke bahasa asing, pencarian teks, sistem baca otomatis untuk tunanetra, input data, pengenalan karakter seperti plat nomor kendaraan, penyelesaian CAPTCHA, dan berbagai masalah teks lainnya. Akurasi, fleksibilitas, dan kecepatan adalah fitur utama yang mencirikan sistem OCR yang baik. Beberapa algoritma untuk pengenalan karakter telah dikembangkan berdasarkan fitur seleksi. Beberapa di antaranya telah menjadi produk komersial yang layak seperti OmniPage, Wordscan, dan TypeReader. Kinerja sistem OCR sering dibatasi oleh ketergantungan pada jenis huruf (*font*), ukuran, dan orientasi teks. Tingkat pengenalan karakter dalam algoritma ini bergantung pada fitur-fitur yang dipilih. Sebagian besar algoritma memerlukan pemrosesan gambar yang ekstensif sebelum fitur-fitur ini diekstraksi, yang dapat mengakibatkan peningkatan waktu komputasi.

C. *Cloud computing*

Model layanan *cloud computing* didasarkan pada konsep berbagi sumber daya komputasi sesuai permintaan, perangkat lunak, dan informasi melalui internet [7]. Perusahaan atau individu membayar untuk mengakses kumpulan virtual sumber daya bersama, termasuk komputasi, penyimpanan, dan layanan jaringan, yang terletak di server jarak jauh yang dimiliki dan dikelola oleh penyedia layanan.

1. *Public Cloud*

Public cloud dijalankan oleh penyedia layanan *cloud* pihak ketiga. Mereka menawarkan komputasi, penyimpanan, dan sumber daya jaringan melalui internet, sehingga perusahaan dapat mengakses sumber daya sesuai permintaan bersama berdasarkan kebutuhan unik dan tujuan bisnis mereka.

2. *Private Cloud*

Private cloud dibangun, dikelola, dan dimiliki oleh satu organisasi dan di-host secara pribadi di pusat data mereka sendiri, umumnya dikenal sebagai "di tempat" atau "on-prem". *Cloud* ini memberikan kontrol, keamanan, dan manajemen data yang lebih besar, namun tetap memungkinkan pengguna internal untuk mendapatkan manfaat dari sumber daya komputasi, penyimpanan, dan jaringan bersama.

3. *Hybrid Cloud*

Hybrid cloud menggabungkan model *public cloud* dan *private cloud* , sehingga perusahaan dapat memanfaatkan layanan *cloud* publik dan mempertahankan kemampuan keamanan dan kepatuhan yang biasa ditemukan dalam arsitektur *private cloud* .

D. Gemini

Gemini adalah antarmuka untuk *LLM* multimodal yang mampu menangani teks, audio, gambar dan lainnya[8]. Gemini didasarkan pada penelitian mutakhir google dalam *LLM* yang dimulai dengan makalah Word2Vec pada tahun 2013. Gemini awalnya mengusulkan arsitektur model baru yang memetakan kata-kata sebagai konsep matematika, diikuti dengan pengenalan model percakapan saraf pada tahun 2015. Kerangka kerja tersebut menunjukkan bagaimana model dapat memprediksi kalimat berikutnya dalam percakapan berdasarkan kalimat sebelumnya.

1. Pra-pelatihan

Gemini didukung oleh model AI Google yang paling mumpuni, yang dirancang dengan berbagai kemampuan dan kasus penggunaan. Seperti kebanyakan LLM saat ini, model-model ini telah dilatih terlebih dahulu pada berbagai data dari sumber yang tersedia untuk umum. Gemini menerapkan filter kualitas pada semua dataset, menggunakan aturan heuristik dan pengklasifikasi berbasis model.

2. Pasca- pelatihan

Setelah pelatihan awal, para LLM menjalani langkah-langkah tambahan untuk menyempurnakan jawaban mereka. Salah satunya disebut *Supervised Fine-Tuning* (SFT), yang melatih model dengan contoh-contoh jawaban yang sangat baik yang telah dipilih dengan cermat.

3. Tanggapan terhadap permintaan pengguna

Pembuatan tanggapan mirip dengan cara manusia melakukan curah pendapat tentang pendekatan yang berbeda untuk menjawab pertanyaan. Setelah pengguna memberikan perintah, Gemini menggunakan LLM yang telah dilatih, konteks dalam perintah dan interaksi dengan pengguna untuk membuat beberapa versi tanggapan.

4. Umpan Balik dan evaluasi manusia

Implementasi model *multimodal* Gemini untuk otomatisasi pembacaan meteran air dimulai dengan tahap akuisisi citra, dimana gambar meteran diumpankan sebagai

input melalui *Application Programming Interface* (API). Model kemudian diinstruksikan melalui *prompt* yang dirancang spesifik untuk menjalankan fungsi *Optical Character Recognition*(OCR) dan mengekstraksi sekuens numerik yang tertera pada *display* meteran. Proses inferensi ini menghasilkan *output* berupa teks digital yang merepresentasikan nilai volume air secara akurat, menggantikan proses pencatatan manual.

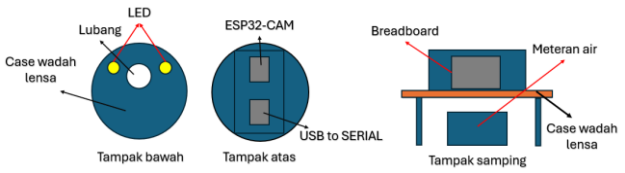
III. METODE

A. Analisis Kebutuhan

Seiring dengan perkembangan teknologi yang semakin pesat, kebutuhan akan sistem yang efektif dan efisien menjadi semakin penting. Analisis kebutuhan sistem merupakan tahap awal yang krusial dalam proses pengembangan sistem, dimana kebutuhan dan keinginan pengguna diidentifikasi dan didokumentasikan secara sistematis. Tujuan dari analisis ini adalah untuk memastikan bahwa sistem yang dikembangkan dapat memenuhi ekspektasi pengguna dan memberikan solusi yang tepat terhadap permasalahan yang ada.

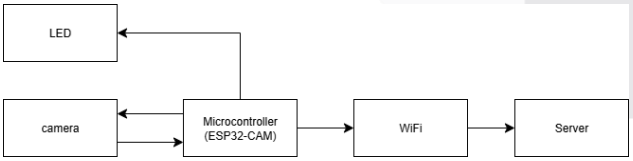
B. Rancangan Sistem Perangkat Keras

Tahapan perancangan sistem bertujuan untuk memberikan gambaran yang jelas mengenai arsitektur dan cara kerja alat yang dibuat. Dalam bab ini, perancangan sistem akan diuraikan melalui dua pendekatan utama: diagram blok untuk menjelaskan struktur komponen perangkat keras, dan *flowchart* untuk menggambarkan alur kerja program.



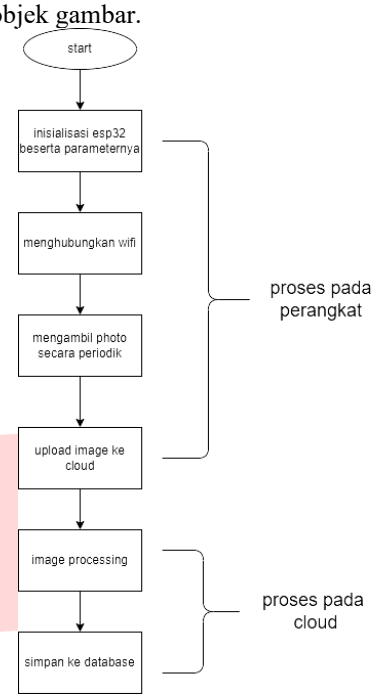
Gambar 1. Sketsa Perangkat

Pada gambar 1 di atas, digambarkan penempatan perangkat keras seperti LED, ESP32-CAM, *USB to SERIAL*, *breadboard*, dan lensa. Lensa berfungsi untuk menambah tingkat kefokusan kamera pada ESP32-CAM.



Gambar 2. Blok Diagram

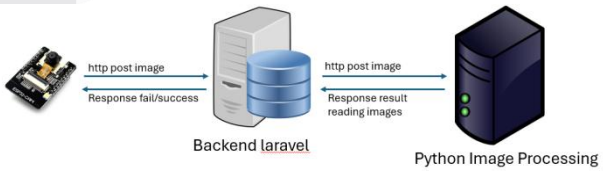
Pada gambar 2 di atas, sistem dibangun menggunakan ESP32-CAM dan juga *cloud* sebagai tempat program disimpan agar dapat diakses secara publik. Pemilihan ESP32-CAM dikarenakan ESP32-CAM bukan hanya memiliki fungsi dasar sebagai mikrokontroler yang dilengkapi fitur wifi agar dapat terhubung ke server yang ada, tetapi juga memiliki fungsi dasar tambahan yaitu fungsi kamera yang digunakan untuk mengambil gambar dari meteran air PDAM pengguna. ESP32-CAM ini akan memiliki id unik yang menandakan kepemilikan seorang pengguna. LED memiliki fungsi untuk menerangi meteran air sebagai objek gambar.



Gambar 3. *Flow Chart* ESP32-CAM

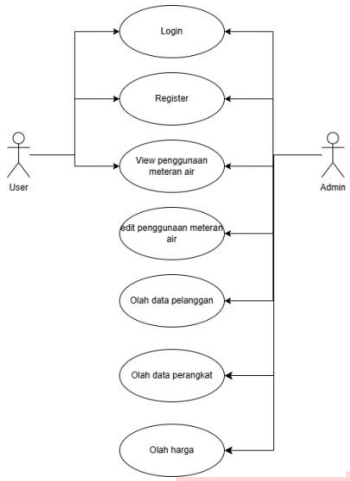
Pada gambar 3 di atas, digambarkan proses yang terjadi pada perangkat keras. Proses yang terjadi pada ESP32-CAM dimulai dari menginisialisasi beberapa paramater penting diantaranya adalah *setting* wifi dan id unik yang menandakan kepemilikan user. Setelah itu ESP32-CAM akan menghubungkan ke wifi agar nantinya dapat mengirimkan data ke server melalui koneksi internet yang ada. Setelah itu ESP32-CAM akan mengambil foto meteran PDAM pengguna secara periodik. Dalam hal ini pengambilan foto akan dilakukan setiap tiga hari sekali. Hal ini dipertimbangkan dari sisi pengguna dan dari sisi beban server yang akan ditanggung. Dikarenakan apabila pengambilan gambar dilakukan per jam atau per dua belas jam ataupun per dua puluh empat jam perubahan angka di sisi meteran air pengguna belum terlalu signifikan, serta akan sangat membebani server untuk menampung data tersebut. Setelah pengambilan gambar, foto tersebut akan dikirimkan ke *cloud* untuk diolah datanya sehingga menghasilkan angka yang dapat dibaca otomatis, lalu hasil pemrosesan tersebut disimpan di database.

C. Arsitektur Aplikasi Pembaca Meteran PDAM



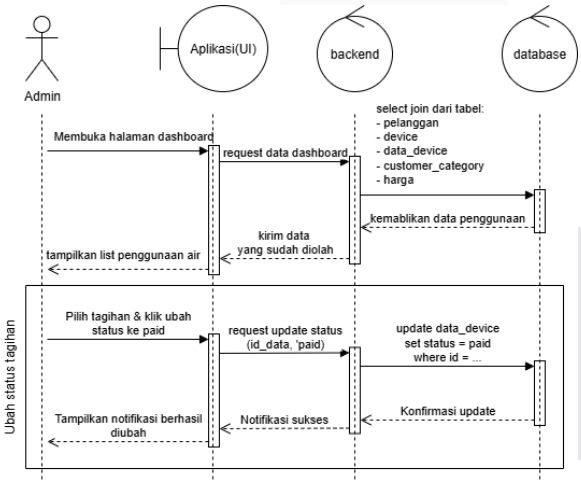
Gambar 4. Rancangan *Software*

Gambar 4 di atas menggambarkan arsitektur sistem yang dirancang. ESP32-CAM akan melakukan proses pengambilan gambar lalu mengirimkannya ke *backend* Laravel. Python akan mengolah gambar tersebut yang selanjutnya disimpan ke database MySQL.



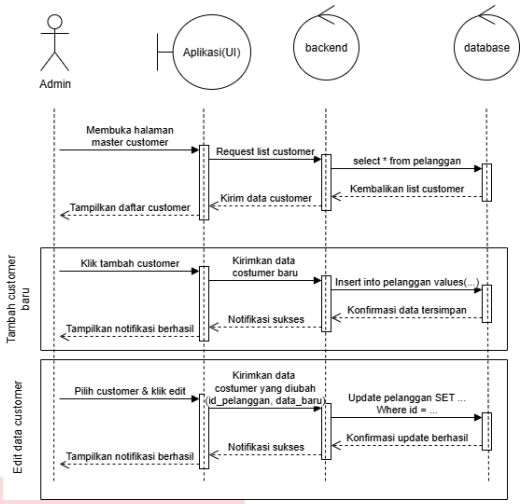
Gambar 5. *Use Case* Diagram

Pada gambar 5 di atas, digambarkan hak akses terhadap pengguna dan admin. Penentuan hak akses adalah aspek krusial dalam pengelolaan data dan informasi, yang berfungsi untuk memastikan bahwa hanya individu yang berwenang dapat mengakses. Hal ini bertujuan agar tidak terjadi tumpang tindih peran dan fungsionalitas masing-masing. Dalam hal ini user atau pengguna umum hanya dapat melakukan login dan melihat penggunaan air, sedangkan admin dapat login, melihat data penggunaan air serta melakukan perubahan data. Perubahan data dilakukan jika terjadi kesalahan dalam mengambil angka meteran karena berbagai faktor. Salah satu faktor yang dapat terjadi dari kondisi lapangan yaitu meteran yang tertutup kotoran (debu), sehingga menyebabkan kesalahan membaca angka meteran dikarenakan angka yang terdapat pada meteran menyerupai angka lainnya.



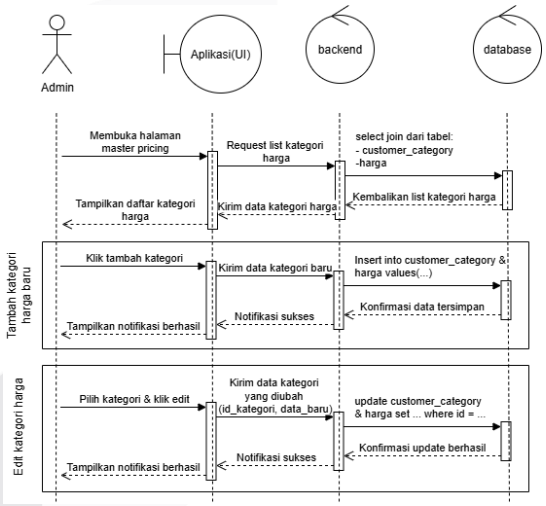
Gambar 6 Alur Kerja Halaman *dashboard Admin*

Pada gambar 6 di atas mengilustrasikan alur kerja admin saat mengakses *dashboard* dan memperbarui status pembayaran. Awalnya, sistem menampilkan data penggunaan dan tagihan yang diambil dari *database* saat admin membuka halaman *dashboard*. Selanjutnya, terdapat alur opsional di mana admin dapat memilih tagihan untuk diubah statusnya menjadi lunas ('paid'). Permintaan ini akan dieksekusi oleh *backend* dengan melakukan *update* ke *database*, lalu sistem memberikan notifikasi keberhasilan kepada admin melalui antarmuka aplikasi.



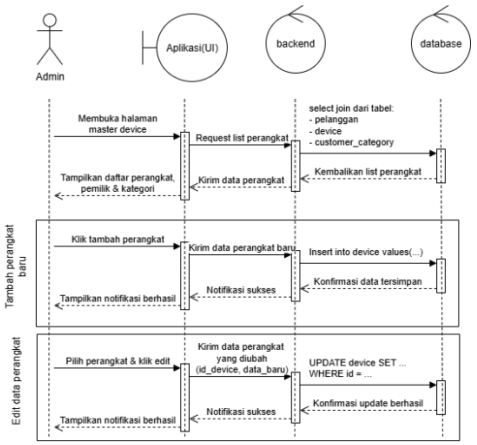
Gambar 7. Alur Kerja Halaman *master customer*

Pada gambar 7 di atas mengilustrasikan alur kerja admin dalam mengelola data master *customer* yang meliputi proses melihat, menambah, dan mengubah data. Setiap aksi yang diinisiasi oleh admin melalui antarmuka aplikasi (UI) akan diteruskan ke *backend* untuk melakukan operasi pada *database*, seperti SELECT untuk menampilkan daftar, INSERT untuk menambah data baru, atau UPDATE untuk mengubah data yang ada. Setelah operasi *database* berhasil, sistem akan mengirimkan notifikasi sukses kembali kepada admin.



Gambar 8. Alur Kerja Halaman *master pricing*

Pada gambar 8 di atas mengilustrasikan alur interaksi seorang admin dengan sistem untuk mengelola data master kategori harga. Proses ini melibatkan tiga aktivitas utama: menampilkan daftar, menambah, dan mengubah data kategori harga. Setiap aksi yang dilakukan admin pada antarmuka aplikasi (UI) akan memicu permintaan ke backend, yang kemudian akan mengeksekusi perintah SQL (select, insert, atau update) ke database dan memberikan notifikasi status keberhasilan kembali ke admin.

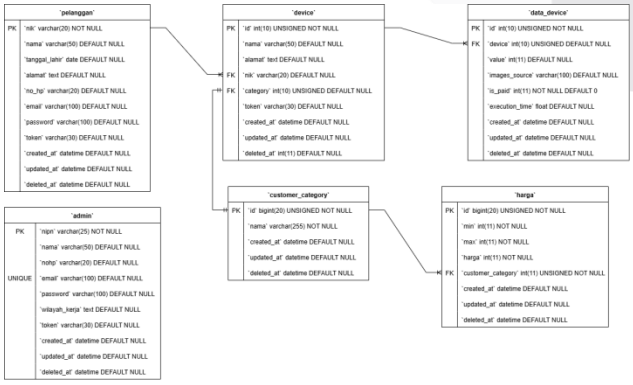


Gambar 9. Alur Kerja Halaman *master device*

Pada gambar 9 mengilustrasikan alur kerja seorang admin dalam mengelola data master perangkat pada sebuah sistem aplikasi. Proses tersebut mencakup tiga skenario utama, yaitu menampilkan daftar perangkat, menambahkan perangkat baru, dan mengubah data perangkat yang telah ada. Setiap aksi dari admin pada antarmuka (UI) akan diproses oleh backend yang kemudian mengeksekusi perintah SQL (SELECT, INSERT, UPDATE) ke database dan diakhiri dengan notifikasi keberhasilan yang ditampilkan kepada admin.

Pada tahap rancangan *backend*, Laravel dikembangkan dengan menggunakan prinsip MVC (*Model-View-Controller*) untuk memisahkan logika aplikasi, tampilan, dan pengelolaan data. Berikut adalah langkah-langkah utama dalam implementasi Laravel:

1. Pembuatan proyek Laravel
2. Konfigurasi database
3. Pembuatan *Model, Controller,* dan *Route*.
   a. *Model*: Model dibuat untuk merepresentasikan entitas yang akan disimpan di database, seperti data pengguna dan hasil analisis gambar
   b. *Controller:* Controller merupakan logika untuk menangani permintaan HTTP, seperti upload dan pengambilan data hasil pemrosesan.
   c. *Route: Routing* mengatur *endpoint* yang bisa diakses oleh *fronted* atau klien lain
5. *Middleware*, digunakan untuk melindungi *endpoint* tertentu dengan menerapkan sistem authentikasi



Gambar 10. ERD database

Struktur database dirancang untuk memenuhi kebutuhan sistem. Tabel utama yang dibuat mencakup:

a. Tabel pelanggan: menyimpan informasi pelanggan seperti, nama, nik, tanggal lahir, Alamat, no handphone, email, password dalam bentuk hash, dan token
b. Tabel device: menyimpan informasi perangkat, seperti nama, alamat, nik sebagai identitas kepemilikan, dan token.
c. Tabel data device: menyimpan informasi data perangkat, seperti id perangkat yang terkait, nilai hasil bacaan pemrosesan gambar, dan *path* gambar yang tersimpan dan token.

Pemrosesan gambar dilakukan dengan Python menggunakan *library* bawaan Gemini untuk menghubungkan ke API Gemini. Gambar yang diunggah oleh pengguna di-*backend* akan dikirim ke Python untuk diproses. Hasil olah gambar akan diteruskan ke *backend* lagi agar diproses lebih lanjut.

D. Skenario Pengujian

Pengujian pertama yang dilakukan adalah pengujian pengiriman gambar. Tujuan dari pengujian ini adalah untuk memastikan bahwa perangkat keras ESP 32-CAM dapat terhubung dengan jaringan, mengambil gambar, serta mengirimkannya ke *server* dengan baik. Langkah awal pengujian dilakukan dengan menghubungkan ESP 32-CAM ke sumber daya dan memastikan konektivitas ke jaringan *WiFi*. Setelah koneksi berhasil, perangkat kemudian diaktifkan dalam mode pengiriman data. Parameter yang diamati dalam pengujian ini meliputi keberhasilan perangkat dalam mengambil gambar dan mengirimkan data gambar ke *server*, munculnya pesan konfirmasi pada *Serial Monitor Arduino IDE* yang menunjukkan bahwa data telah diterima oleh *server* beserta hasil bacaannya, serta tersimpannya gambar secara otomatis di *server*.

Pengujian fitur *Login Admin* yang bertujuan untuk memastikan bahwa sistem keamanan pada halaman *login* berfungsi dengan baik, yaitu hanya memberikan akses kepada pengguna yang memiliki kredensial yang valid. Pengujian dilakukan melalui dua skenario: pertama, pengguna membuka halaman *login*, kemudian memasukkan *username* dan *password* yang benar; kedua, pengguna memasukkan *username* atau *password* yang salah. Pada skenario pertama, sistem diharapkan dapat mengautentikasi pengguna dan mengarahkan ke halaman *dashboard admin*. Sebaliknya, pada skenario kedua, sistem harus menolak akses dan menampilkan pesan bahwa *login* gagal.

Selanjutnya pengujian fitur *dashboard admin* bertujuan untuk memastikan bahwa proses konversi gambar menjadi angka, menampilkan jumlah pemakaian dan besaran tagihannya dapat berjalan dengan baik, serta administrator dapat melakukan tindakan penting dari halaman *dashboard*, seperti mengubah angka meteran jika terjadi kesalahan pembacaan karena faktor tertentu, seperti terdapat kotoran pada meteran air yang menyebabkan angka pada meteran menyerupai angka lainnya, serta memperbarui status pembayaran pelanggan. Pengujian dilakukan dengan mengklik tombol *"Edit"* pada data pelanggan, kemudian mengubah nilai pembacaan meteran atau status pembayaran dari *"unpaid"* menjadi *"paid"*, atau sebaliknya. Setelah itu, pengguna menekan tombol *"Save Changes"*. Sistem

diharapkan dapat memperbarui data sesuai dengan perubahan yang dimasukkan oleh admin.

Pengujian fitur olah data pelanggan dilakukan untuk memverifikasi bahwa admin dapat mengelola data pelanggan, baik menambah data baru maupun mengubah data yang sudah ada. Untuk menambah data, admin mengakses menu *Customer*, melengkapi formulir pendaftaran pelanggan baru, kemudian menekan tombol *"Save Changes"*. Untuk mengubah data, admin memilih data pelanggan yang ingin diperbarui, lalu menekan tombol *"Edit"*, melakukan perubahan pada formulir, dan menyimpannya. Hasil yang diharapkan adalah sistem dapat menyimpan dan menampilkan data pelanggan baru dalam tabel, serta memperbarui informasi pelanggan yang telah ada dengan benar.

Pengujian fitur olah data perangkat untuk memastikan bahwa admin dapat mengelola data perangkat meteran air yang terhubung ke sistem. Proses pengujian melibatkan dua tindakan, yaitu penambahan dan pengubahan data perangkat. Untuk menambah perangkat, admin membuka menu *Devices*, menekan tombol tambah, melengkapi formulir data perangkat baru, lalu menyimpannya. Untuk mengubah data, admin memilih perangkat yang akan diperbarui, mengklik tombol *"Edit"*, melakukan perubahan, dan menekan *"Save Changes"*.

Pengujian fitur *Master* harga bertujuan untuk memvalidasi fungsi pengelolaan harga air, termasuk penambahan dan perubahan skema harga berdasarkan kategori pelanggan. Pengujian dilakukan dengan mengakses menu *Pricing*, menekan tombol tambah untuk menambahkan skema harga baru, kemudian mengisi formulir kategori dan rentang harga, lalu menyimpan data. Untuk mengubah harga, admin memilih salah satu entri harga yang tersedia, menekan tombol *"Edit"*, memperbarui informasi harga, dan menyimpannya.

Pengujian fitur *Dashboard* pengguna agar pengguna dapat melihat penggunaan air terkini dan besaran tagihannya. Pengujian dilakukan dengan login terlebih dahulu. Riwayat penggunaan air dapat diakses melalui tombol "History". Sistem yang diharapkan dapat menampilkan penggunaan air terkini beserta besaran tagihannya dan juga melihat riwayat penggunaan airnya.

Pengujian Akurasi merupakan ukuran seberapa dekat hasil pengukuran dengan nilai sebenarnya. Dalam konteks pembacaan otomatis meteran air, akurasi menunjukkan sejauh mana hasil pembacaan ESP32-CAM mendekati angka yang tercatat pada meteran air fisik secara manual. Metodologi pengujian akurasi sistem dilaksanakan secara komprehensif melalui tiga skenario utama untuk mengevaluasi performa dan robustisitas model. Pengujian diawali dengan penetapan tolok ukur (*baseline*) akurasi pada kondisi ideal, yang dicirikan oleh intensitas cahaya optimal dan ketiadaan objek penghalang pada digit meteran. Selanjutnya, ketahanan sistem dievaluasi terhadap variasi kondisi pencahayaan, yang dimanipulasi secara terkontrol menggunakan potensiometer untuk mensimulasikan spektrum cahaya dari terang hingga redup. Sebagai tahap akhir, kemampuan inferensi model diuji dalam skenario sebagian angkanya tertutup, di mana sebuah objek kecil diletakkan di atas digit angka untuk menciptakan ambiguitas visual dan mengukur ketahanan sistem terhadap data citra yang tidak sempurna. Untuk menghitung nilai akurasi tersebut dapat digunakan persamaan matematika berikut:

$$akurasi = \frac{Jumlah\ Data\ Benar}{Total\ Data} \times 100\% \qquad (1)$$

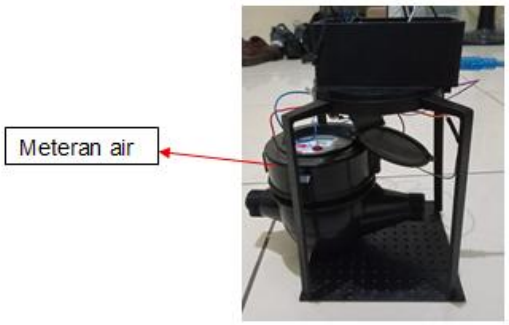
IV. HASIL DAN PEMBAHASAN

A. Pembuatan Perangkat Keras

Proses pembuatan perangkat keras pembacaan angka meter PDAM berbasis IoT berhasil diselesaikan dengan menggunakan beberapa komponen. Perangkat ini memadukan komponen kabel jumper, *breadboard*, kaca pembesar, LED, FTDI, dan ESP32-CAM sebagai komponen utamanya. Alat ini dirancang untuk untuk mengambil gambar pada meteran air lalu mengirimkannya pada server.

Gambar 11. Perangkat dari sisi atas

Gambar 11 menggambarkan peletakan komponen ESP32-CAM, *USB to SERIAL breadboard,* dan lensa. Ketiga komponen tersebut saling terhubung satu sama lain.



Gambar 12. Gambar Perangkat dari sisi samping

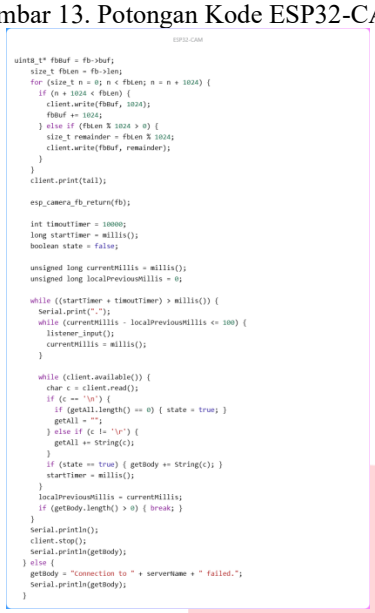
Gambar 12 menggambarkan peletakan meteran air yang diposisikan di bagian bawah dari komponen utama ESP32-CAM.

B. Implementasi Perangkat Lunak Pada Perangkat

Perangkat keras yang telah dirancang akan diterapkan program perangkat lunak agar mampu berjalan sesuai yang diinginkan

```
ESP32-CAM

unsigned long startTime_us = micros(); // High-resolution timer for function duration
  size_t initial_free_heap = esp_get_free_heap_size(); // Total free heap before
function

  String getAll;
  String getBody;

  camera_fb_t* fb = NULL;
  fb = esp_camera_fb_get();

  unsigned long currentMillisCam = millis();
  unsigned long localPreviousMillisCam = currentMillisCam;

  if (!fb) {
    Serial.println("Camera capture failed");

    //alternate delay 1000
    while (currentMillisCam - localPreviousMillisCam <= 1000) {
      listener_input();
      currentMillisCam = millis();
    }
    localPreviousMillisCam = currentMillisCam;

    ESP.restart();
  }

  Serial.println("Connecting to server: " + serverName);

  if (client.connect(serverName.c_str(), serverPort)) {
    Serial.println("Connection successful!");
    String head = "--RandomNerdTutorials\r\nContent-Disposition: form-data;
name=\"imageFile\"; filename=\"esp32-cam.jpg\"\r\nContent-Type: image/jpeg\r\n\r\n";
    String tail = "\r\n--RandomNerdTutorials--\r\n";

    uint32_t imageLen = fb->len;
    uint32_t extraLen = head.length() + tail.length();
    uint32_t totalLen = imageLen + extraLen;

    client.println("POST " + serverPath + " HTTP/1.1");
    client.println("Host: " + serverName);
    client.println("Content-Length: " + String(totalLen));
    client.println("Content-Type: multipart/form-data; boundary=RandomNerdTutorials");
    client.println();
    client.print(head);
```

Gambar 13. Potongan Kode ESP32-CAM



```
uint8_t* fbBuf = fb->buf;
    size_t fbLen = fb->len;
    for (size_t n = 0; n < fbLen; n = n + 1024) {
      if (n + 1024 < fbLen) {
        client.write(fbBuf, 1024);
        fbBuf += 1024;
      } else if (fbLen % 1024 > 0) {
        size_t remainder = fbLen % 1024;
        client.write(fbBuf, remainder);
      }
    }
    client.print(tail);

    esp_camera_fb_return(fb);

    int timoutTimer = 10000;
    long startTimer = millis();
    boolean state = false;

    unsigned long currentMillis = millis();
    unsigned long localPreviousMillis = 0;

    while ((startTimer + timoutTimer) > millis()) {
      Serial.print(".");
      while (currentMillis - localPreviousMillis <= 100) {
        listener_input();
        currentMillis = millis();
      }

      while (client.available()) {
        char c = client.read();
        if (c == '\n') {
          if (getAll.length() == 0) { state = true; }
          getAll = "";
        } else if (c != '\r') {
          getAll += String(c);
        }
        if (state == true) { getBody += String(c); }
        startTimer = millis();
      }
      localPreviousMillis = currentMillis;
      if (getBody.length() > 0) { break; }
    }
    Serial.println();
    client.stop();
    Serial.println(getBody);
  } else {
    getBody = "Connection to " + serverName + " failed.";
    Serial.println(getBody);
  }
```

Gambar 13. Potongan kode ESP 32-CAM (lanjutan)

Pada gambar 13 di atas merupakan potongan kode program pada ESP32-CAM. Program di atas melakukan proses pengambilan gambar pada ESP32-CAM lalu mengirimkannya ke server.

```
function addDataDevice(string $tokenUser, string $tokenDevice, Request $request)
    {
        $user = Pelanggan::where('token', $tokenUser);
        if ($user->count() == 0) {
            return response($this->responses(false, 'Pelanggan is not found'), 404);
        }

        $device = device::where('token', $tokenDevice)->where('nik', $user->first()->nik);
        if ($device->count() == 0) {
            return response($this->responses(false, 'Device is not found or device is not match with the customer'), 404);
        }

        if ($request->hasFile('imageFile')) {
            $files = $request->file('imageFile');
            $files->move(public_path($tokenDevice), 'live.jpg');

            $original_date = time();
            $tanggal = date("Y-m-d_H-i-s", $original_date);
            $filesNameOCR = $tanggal . "-ocr." . $files->getClientOriginalExtension();
            copy(public_path($tokenDevice . '/live.jpg'), public_path($tokenDevice . '/' . $filesNameOCR));

            $response = shell_exec('py ' . public_path('gemini.py') . ' ' . $tokenDevice . '/' . $filesNameOCR);
            $response = json_decode($response);

            unlink(public_path($tokenDevice . '/' . $filesNameOCR));

            if ($response->text != "") {
                $tanggal = date("Y-m-d_H-i-s", $original_date);
                $available_data = DataDevice::where('device', $device->first()->id)->whereDate('created_at', '=', date("Y-m-d", $original_date))->where('value', '=', $response->text);
                if ($available_data->count() == 0) {
                    $filesName = $tanggal . '.' . $files->getClientOriginalExtension();
                    copy(public_path($tokenDevice . '/live.jpg'), public_path($tokenDevice . '/' . $filesName));
                    $inserts = DataDevice::create([
                        'device' => $device->first()->id,
                        'value' => $response->text,
                        'images_source' => $tokenDevice . '/' . $filesName,
                        'execution_time' => $response->execution_time
                    ]);
                }
                return $this->responses(true, 'Data received successfully, the value is ' . $response->text);
            } else {
                return $this->responses(false, 'Data read the number fail');
            }
        } else {
            return response($this->responses(false, 'imageFile not found'), 400);
        }
    }
```

Gambar 14. Gambar potongan kode backend

Pada gambar 14 di atas merupakan kode program pada sisi *backend* yang melakukan proses menerima gambar dari perangkat yang selanjutnya dilakukan pencocokan data berdasarkan token lalu menghubungkannya ke Gemini.

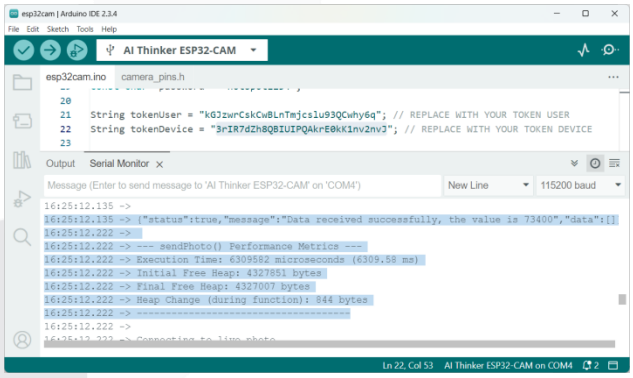
```
from google import genai
from google.genai import types
import sys
import re
import json
import time

file_path = sys.argv[1]

# with open('C:/Users/rachm/OneDrive/Dokumen/KULIAH/proposal/project/water-metering-app/public/Rm13i2xt2dQE63xCBjmkDb25QK6bLP/2025-05-31_08-11-19.jpg', 'rb') as f:
#     image_bytes = f.read()

client = genai.Client(api_key="AIzaSyBTqN5Th1a0a79NlMzcU9i6gkUPOSsHZZ0")

# Start the timer
start_time = time.time()

my_file = client.files.upload(file=file_path)
response = client.models.generate_content(
    model="gemini-2.0-flash",
    contents=[my_file,"get the water meter value, please only response the number. if fail just empty response"]
)

get_values = re.sub(r'[^0-9]', '', response.text)
if len(get_values) != 5:
    get_values = ""
# Calculate the time taken
execution_time = time.time() - start_time

print(json.dumps({"text":get_values, "execution_time": execution_time}))
```

Gambar 15. Gambar potongan kode Gemini

Pada gambar 15 di atas merupakan implementasi kode program yang dibangun menggunakan bahasa Python dengan memanfaatkan pustaka (*library*) resmi Gemini. Dalam pemanfaatan *Application Programming Interface* (API) ini, aspek privasi dan kerahasiaan data dapat terjaga melalui mekanisme kontrol akses, dimana pemilik kunci API (*API key owner*) tidak dapat melihat log histori interaksi apabila fitur pencatatan data untuk operasi baca (*read*) dan tulis (*write*) dinonaktifkan.

C. Implementasi dan Hasil Uji Fungsionalitas

Setelah menyelesaikan pembuatan perangkat keras dan perangkat lunak, sistem yang dibangun perlu dilakukan pengujian. Langkah ini diperlukan untuk memastikan bahwa alat tersebut mampu beroperasi dengan baik dan mencapai tujuan yang diinginkan. Langkah pengujian ini akan memberikan informasi seberapa efektif sistem yang ada serta mendeteksi permasalahan yang muncul atau potensi permasalahan yang akan muncul.



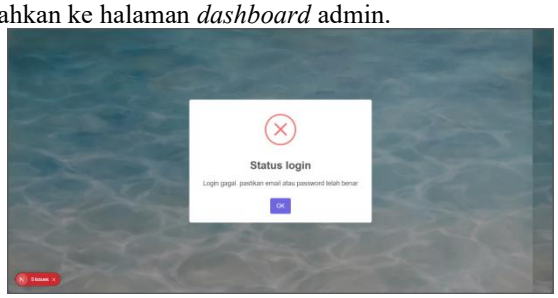
Gambar 16. Bukti Pengiriman gmabar berhasil pada serial monitor

Dari gambar 16 di atas, terlihat bahwa ESP32-CAM dapat mengirimkan gambar ke server dan menerima respon balik dari server.

Gambar 17. Contoh gambar yang berhasil disimpan

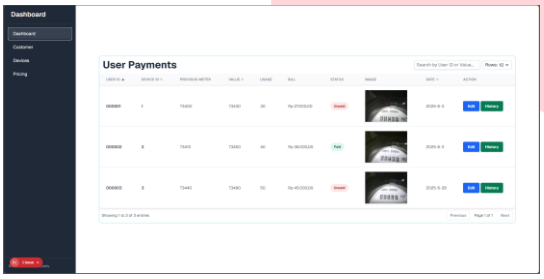
Gambar 17 di atas menunjukkan bahwa perangkat berhasil mengirimkan gambar ke server dan disimpan di server.

Proses setelah akun *admin* dianggap valid, admin akan diarahkan ke halaman *dashboard* admin.



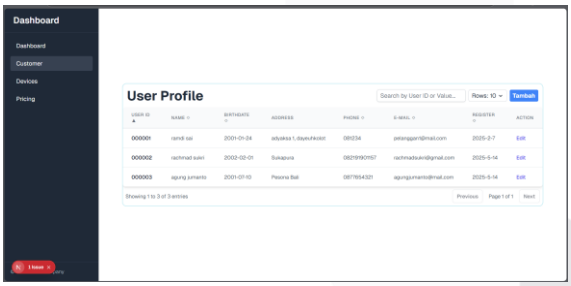
Gambar 18. Pesan gagal ketika akun tidak valid

Dari gambar 18 di atas menunjukkan pesan gagal ketika admin salah memasukkan email atau password, dalam hal ini menunjukkan akun tidak valid.



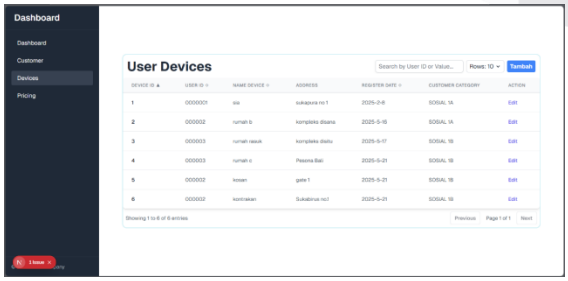
Gambar 19. Halaman *dashboard* admin

Pada gambar 19 di atas menampilkan data list penggunaan pelanggan, admin dapat melihat detail riwayat penggunaan tiap perperangkat. Admin dapat melakukan perubahan status pembayaran.



Gambar 20. Halaman Master Pelanggan

Dari gambar 10 di atas menampilkan halaman list pengguna, admin dapat menambah dan melakukan perubahan data diri pada pelanggan.

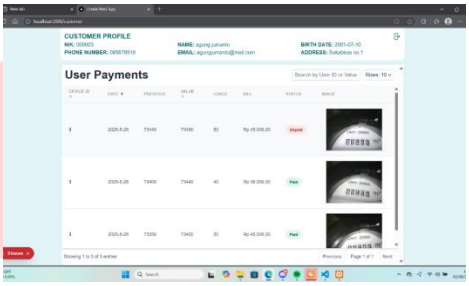


Gambar 21. Halaman Master Perangkat

Pada gambar 21 di atas menampilkan halaman perangkat, admin dapat melukakn penambahan perangkat maupun melakukan perubahan pada perangkat.

Gambar 22. Halaman master harga

Pada gambar 22 di atas menampilkan halaman *master* harga, admin dapat melakukan penambahan kategori harga dan melakukan perubahan kategori harga.



Gambar 23. Halaman *dashboard* Pengguna

Pada gambar 23 di atas, pengguna dapat melihat penggunaan air terkini dan juga riwayat penggunaan airnya.

D. Pengujian Akurasi dan Performa

Pengujian akurasi meteran air telah dilakukan untuk menilai keefektifan alat. Berikut tabel yang menunjukkan hasil dari 35 data.

Tabel 1. Sampel Data

Id	device	value	execution_time(s)	status
1	1	20986	5,3009	Sukses
2	1	11789	5,3483	Sukses
3	1	13562	5,8996	Sukses
4	1	14310	4,9550	Sukses
5	1	15876	4,9572	Sukses
6	1	24532	4,8862	Sukses
7	1	25301	4,0911	Sukses
8	1	26890	5,0411	Sukses
9	1	27567	6,8279	Sukses
10	1	30781	5,2243	Sukses
11	1	32456	5,7404	Sukses
12	1	33214	4,7816	Sukses
13	1	34923	5,8709	Sukses
14	1	36201	5,398	Sukses
15	1	37985	5,0015	Sukses
16	1	38410	4,7298	Sukses
17	1	39876	3,6827	Sukses
18	1	41234	4,1333	Sukses

Id	device	value	execution_time(s)	status
19	1	42765	5,4959	Sukses
20	1	43987	4,1579	Sukses
21	1	45231	4,1198	Sukses
22	1	46890	5,1023	Sukses
23	1	47901	4,8492	Sukses
24	1	48655	5,2689	Sukses
25	1	50210	5,0773	Sukses
26	1	51789	5,1439	Sukses
27	1	52987	4,9148	Sukses
28	1	54890	4,9825	Sukses
29	1	56321	5,0492	Sukses
30	1	57456	5,4742	Sukses
31	1	58901	4,8559	Sukses
33	1	60123	4,6363	Sukses
34	1	61890	5,7808	Sukses
35	1	63210	4,9221	Sukses
54	3	73532	6,3817	Sukses
RATA-RATA			4,4071	

Id pada tabel di atas merupakan *id* pada data tabel sistem *database*. Kolom device merupakan *id* pada perangkat dalam tabel sistem *database*. Analisis performa komputasi sistem, yang dievaluasi melalui 35 data pengujian, menunjukkan nilai rerata waktu eksekusi (*average execution time*) sebesar 4,407 detik. Nilai ini merepresentasikan waktu yang dibutuhkan oleh sistem untuk memproses satu citra masukan hingga menghasilkan data keluaran.

Berdasarkan data hasil uji akurasi meteran air dari 35 percobaan yang dilakukan, terdapat 35 data yang sesuai. Akurasi dapat ditentukan dengan menggunakan rumus
Maka akurasi dapat dihitung sebagai berikut:

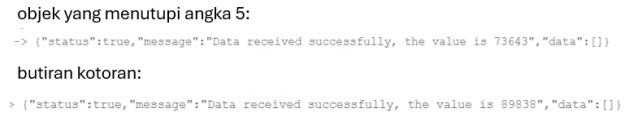
$$akurasi = \frac{35}{35} \times 100\% = 100\%$$

Berdasarkan serangkaian pengujian yang telah dilakukan, sistem menunjukkan performa yang optimal dengan mencapai akurasi 100%.

Tahapan pengujian ini berfokus pada evaluasi ketahanan model dalam mendeteksi atau mengenali angka ketika terdapat gangguan visual dari objek lain.

Gambar 24. Sampel Gangguan dengan kotoran

Pada gambar 24 di atas menunjukkan beberapa gambar dengan beberapa gangguan diantaranya objek yang menutupi angka 5 dan butiran kotoran. Berikut hasil bacaan pada sistem.



Gambar 25. Hasil Bacaan dari gangguan

Pada gambar 25 di atas menunjukkan hasil pembacaan kedua jenis gangguan yang dilakukan.

Gambar 26. Variasi Intensitas Cahaya

Pada gambar 26 di atas menunjukkan 3 intensitas cahaya yang berbeda dan memiliki hasil pembacaan angka 83838 secara konsisten.

Gambar 27. Pantulan cahaya pada angka meteran air

Pada gambar 27 di atas menunjukkan terdapat pantulan cahaya pada digit terakhir yang cukup mengganggu dalam proses pembacaan angka meteran air. Pantulan cahaya dapat terjadi karena desain perangkat yang dibangun masih rawan terhadap pergeseran sehingga dapat memicu pantulan cahaya jika meteran air berada pada posisi yang kurang tepat.

Gambar 28. Pembacaan angka salah faktor pantulan cahaya

Pada gambar 4.18 menunjukkan hasil pembacaan angka yang salah karena efek pantulan cahaya pada digit terakhir. Angka digit terakhir memiliki angka 8 namun efek pantulan cahaya mengakibatkan sistem membacanya sebagai angka 6.

V. KESIMPULAN

Berdasarkan penelitian dan pengujian yang telah dilakukan, dapat disimpulkan bahwa perangkat keras berhasil dirancang dan dibangun dengan baik, di mana ESP32-CAM mampu mengambil gambar serta mengirimkannya secara optimal. Integrasi antara perangkat dan platform juga berjalan lancar, sehingga data yang dikirim mikrokontroler dapat diterima oleh server dan ditampilkan pada *dashboard* dengan berbagai fitur pendukung untuk memaksimalkan pemanfaatan informasi. Selain itu, penggunaan API Gemini terbukti mampu memberikan tingkat akurasi data hingga 100%. Meskipun demikian, sistem ini masih memiliki beberapa keterbatasan, seperti potensi kesalahan pembacaan angka akibat gangguan visual oleh kotoran, serta desain perangkat yang masih rawan bergeser sehingga menimbulkan pantulan cahaya pada angka.

REFERENSI

[1] B. Roshan, S. Abhijeet, B. Rahul, dan P. Rahul, “Smart Water Meter and Supply Controlling System Using IoT,” *IRJETS*, vol. 2, no. 12, Des. 2020, ISSN: 2582-5208.

[2] V. Ahmad, “Debit air PDAM Giri Menang Turun 10 Persen Wilayah Perbukitan Terganggu,” Detik.com, 2023. [Online]. Tersedia: https://www.detik.com/bali/berita/d-6967492/debit-air-pdam-giri-menang-turun-10-persen-wilayah-perbukitan-terganggu [Diakses: 16 Mei 2024].

[3] D. P. Erwin, U. Marissa, dan H. R. Muhammad, “Identifikasi Text Meteran Air Menggunakan Metode *Run-Length Smearing Algorithm* (RLSA),” JUMIKA, vol. 4, no. 2, Des. 2021, ISSN: 2723-8091.

[4] Hendrick, F. Hanifa, dan F. Ivana, “Pemanfaatan ESP32-CAM untuk Mengukur Ketinggian Air Menggunakan Metode Image Processing,” Seminar Nasional Terapan Riset Inovatif (SENTRINOV), vol. 6, no. 1, 2020, ISSN: 2477-2097.

[5] F. Mohammad, J. Anarase, M. Shingote, dan P. Ghanwat, “*Optical Character Recognition*Implementation Using Pattern Matching,” International Journal of Computer Science and Information Technologies, pp. 2088–2090, 2014.

[6] Google *Cloud* , “*Cloud computing*,” 2024. [Online]. Tersedia: https://*cloud* .google.com/discover/types-of-*cloud* -computing [Diakses: 11 Mei 2024].

[7] M. Idris, “Kepanjangan PDAM, Pemilik, dan Layanannya,” Kompas.com, 2023. [Online]. Tersedia: https://money.kompas.com/read/2023/05/28/210433126/kepanjangan-pdam-pemilik-dan-layanannya [Diakses: 10 Juli 2025].

[8] Google, “An Overview of the Gemini App,” 2025. [Online]. Tersedia: https://gemini.google/overview/?hl [Diakses: 10 Juli 2025].