DAFTAR GAMBAR

DAFTAR TABEL

INTISARI

PENGEMBANGAN SISTEM KONTROL PENGISIAN DAYA MOBIL LISTRIK BERBASIS APLIKASI *WEB* DAN IOT DI HOTEL TENTREM YOGYAKARTA

Rafli Rajendra Permana
21/473999/SV/18868

Penggunaan kendaraan listrik di Indonesia mengalami peningkatan signifikan yang ditandai dengan kenaikan penjualan mobil listrik sebesar 39,91% pada Mei 2024 dibandingkan tahun sebelumnya. Peningkatan ini mendorong munculnya infrastruktur pendukung seperti *charging station* di kota-kota besar. Hotel Tentrem Yogyakarta memiliki *charging station* bermerek Porsche yang masih dioperasikan secara manual. Pengoperasian memerlukan bantuan mekanik atau *engineer* setelah tamu membeli waktu pengisian di register. Sistem manual ini dinilai kurang efisien dibandingkan *charging station* yang dapat melayani sendiri. Berdasarkan permasalahan tersebut, dikembangkan sistem *token* berbasis IoT dengan *timer* otomatis pada *charging station*.

Pengembangan sistem memanfaatkan beberapa komponen utama. ESP32 digunakan sebagai *microcontroller*. Sensor arus SCT-013 dipasang untuk mendeteksi penggunaan *charging station*. Antarmuka pengguna dikembangkan dalam bentuk aplikasi *web* berbasis *Laravel Framework*. Protokol MQTT diimplementasikan dalam komunikasi antara perangkat IoT dan aplikasi *web*. Hasil pengujian menunjukkan keberhasilan integrasi perangkat IoT dengan aplikasi *web*. Sistem *token* berhasil diterapkan dengan mekanisme penggunaan berulang dalam rentang waktu 24 jam hingga durasi habis. *Timer* berjalan secara otomatis saat arus terdeteksi melebihi 0,5 A dan dapat dihentikan sementara ketika arus turun di bawah 0,3 A. Antarmuka *monitoring* berhasil disinkronkan dengan antarmuka *charging station* secara *real-time*.

Kata kunci: *charging station*, IoT, Laravel, MQTT, LAMP *Stack*

ABSTRACT

DEVELOPMENT OF WEB APPLICATION AND IOT BASED CHARGING STATION CONTROL SYSTEM AT HOTEL TENTREM YOGYAKARTA

Rafli Rajendra Permana
21/473999/SV/18868

The use of electric vehicles in Indonesia has experienced a significant increase, with electric car sales rising by 39.91% in May 2024 compared to the previous year. This increase has led to the emergence of electric vehicle supporting infrastructure such as charging stations in major cities. Hotel Tentrem Yogyakarta has a Porsche-branded charging station that are still operated manually, requiring assistance from engineers to operate them after the guest purchase charging time at the register. This manual system is less efficient compared to self-serving charging stations. Based on this issue, this project was conducted to develop an IoT-based token system with automatic timer for these charging stations.

The system uses ESP32 as the microcontroller, SCT-013 current sensor to detect charging station usage, a Laravel Framework-based web application for the user interface, and MQTT for communication between IoT devices and the web application. Test results show that the system successfully integrated IoT devices with the web application. The token system was successfully implemented with a logic allowing repeated use within a 24-hour period until the remaining time or duration is exhausted. The timer can run automatically when current is detected above 0.5 A and can be temporarily paused when current drops below 0.3 A. The monitoring interface successfully synchronizes with the charging station interface to monitor usage in real-time.

Keyword: charging station, IoT, Laravel, MQTT, LAMP Stack

BAB I
PENDAHULUAN

1.1. Latar Belakang

Selama setahun terakhir, terjadi peningkatan pesat dalam penggunaan kendaraan listrik sebagai kendaraan pribadi. Berdasarkan data Gabungan Industri Kendaraan Bermotor Indonesia, penjualan mobil listrik mencapai 6.033 unit pada Mei 2024. Angka ini menunjukkan kenaikan sebesar 39,91% dibandingkan dengan Mei 2023. Peningkatan permintaan terhadap teknologi kendaraan dengan sumber energi bersih terjadi sebagai upaya pengurangan emisi karbon. Transisi menuju penggunaan energi yang lebih ramah lingkungan menjadi langkah penting dalam pembangunan berkelanjutan [1].

Peningkatan jumlah mobil listrik seharusnya diiringi dengan penambahan stasiun pengisian daya (*charging station*). Dalam skenario *Business as Usual* (BAU), kebutuhan stasiun pengisian daya di Indonesia diprediksi akan meningkat dari 7.953 unit pada tahun 2025 menjadi 202.424 unit pada tahun 2040. Pembangunan infrastruktur pengisian daya untuk menunjang pertumbuhan jumlah mobil listrik perlu dilakukan dengan tepat, mengikuti standar perancangan dan pembangunan infrastruktur seperti yang telah diterapkan di Amerika Serikat. Dibandingkan dengan SPBU konvensional, stasiun pengisian daya mobil listrik memiliki keunggulan karena lebih mudah ditempatkan di berbagai lokasi umum, termasuk area parkir. Ukuran stasiun pengisian yang relatif kecil dan tidak memerlukan tangki penyimpanan bahan bakar menjadi faktor utama kemudahan penempatan tersebut [2].

Saat ini, Hotel Tentrem Yogyakarta memiliki dua unit stasiun pengisian daya bermerek Porsche yang masih dioperasikan secara manual. Pengoperasian stasiun pengisian memerlukan bantuan teknisi atau *engineer* yang bertugas untuk menyalakan pemutus arus (*circuit breaker*) tiga fase setelah tamu melakukan pembayaran di bagian register. Sistem manual tersebut memakan waktu karena pihak register harus menghubungi *engineer* terlebih dahulu. Pengembangan sistem kontrol stasiun pengisian daya yang dilengkapi mekanisme *token* dan *timer*

diusulkan sebagai solusi, dengan menggunakan mikrokontroler ESP32 dan Laravel *Framework* sebagai kerangka utama sistem.

1.2. Rumusan Masalah

Stasiun pengisian daya (*charging station*) di Hotel Tentrem Yogyakarta memerlukan sistem kontrol karena pengoperasiannya masih dilakukan secara manual dan membutuhkan bantuan *engineer*.

1.3. Tujuan Proyek Akhir

Proyek ini bertujuan untuk mengembangkan sistem kontrol pada *charging station* di Hotel Tentrem Yogyakarta yang terintegrasi dengan aplikasi *web* dan IoT, serta dilengkapi dengan mekanisme *token* dan *timer* otomatis untuk menggantikan sistem manual yang ada.

1.4. Batasan Masalah

Adapun batasan masalah dalam melaksanakan proyek ini adalah sebagai berikut:

1. Produk yang dijual oleh perusahaan melalui proyek ini hanya berbentuk waktu (dalam satuan jam), sementara pengukuran konsumsi daya tidak termasuk dalam lingkup proyek mengikuti kebijakan perusahaan.
2. Aplikasi *web* dibuat menggunakan *framework* Laravel versi 11.
3. Mikrokontroler yang digunakan adalah ESP32 dengan protokol komunikasi MQTT (*publish subscribe*).
4. Sensor arus yang digunakan adalah SCT-013 100, dan sensor ini hanya digunakan untuk mendeteksi adanya arus listrik.
5. Sistem yang dikembangkan tidak tergabung ke dalam sistem administrasi perusahaan, namun menggunakan server terpisah.

1.5. Manfaat Proyek Akhir

Dengan dilakukannya proyek ini, diharapkan dapat memberikan manfaat sebagai berikut:

1. Meminimalisir *human-error* dalam pengoperasian infrastruktur pengisian daya.
2. Mempersingkat alur transaksi alokasi waktu pengisian daya dari Register atau *Administrator* ke mobil listrik pelanggan.

3. Meningkatkan *revenue* yang dihasilkan dari fasilitas pengisian daya untuk mobil listrik.
4. Memberikan inspirasi bagi perusahaan tentang penerapan aplikasi *web* berbasis Laravel Framework.
5. Mempermudah pekerjaan tim *engineer* dalam pengoperasian fasilitas pengisian daya.

1.6. Sistematika Penulisan

Secara garis besar, laporan proyek akhir ini akan ditulis dengan sistematika penelitian yang dibagi dalam 5 bab. Berikut adalah bagian-bagian tersebut:

BAB I Pendahuluan

Bab ini berisi latar belakang, tujuan proyek akhir, batasan masalah, manfaat proyek akhir, dan sistematika penulisan.

BAB II Tinjauan Pustaka

Bab ini berisi studi pustaka yang menguraikan penelitian-penelitian yang sudah dilakukan terkait dengan topik-topik dan kata kunci yang digunakan dalam proyek akhir ini, dasar teori yang berisi penjelasan mendetail tentang kata kunci tersebut, dan hipotesis.

BAB III Metode Proyek Akhir

Pada bab ini berisi rincian alat dan bahan, penjelasan rancangan atau desain dan alur sistem, dan juga tahapan-tahapan yang dilakukan dalam melaksanakan proyek akhir ini sesuai dengan rancangan.

BAB IV Hasil dan Pembahasan

Bab ini berisi hasil yang didapat dari proyek akhir, data yang diambil pada saat pengujian, serta analisis tentang proyek akhir yang telah selesai.

BAB V Penutup

Bab ini berisi kesimpulan dan saran untuk penelitian atau proyek selanjutnya.

BAB II
KAJIAN PUSTAKA

2.1. Studi Pustaka

Penggunaan mobil listrik dan kendaraan bertenaga listrik lainnya semakin meningkat menjelang pertengahan dekade ini. Peningkatan ini terlihat dari pertumbuhan penjualan mobil listrik, baik di Indonesia maupun secara global. Fenomena tersebut menandakan bahwa masyarakat mulai beralih dari kendaraan berbahan bakar gas ke kendaraan listrik. Perubahan ini perlu didukung dengan infrastruktur *charging station* yang memadai guna meningkatkan insentif penggunaan mobil listrik. Sistem *charging station* dapat dilengkapi dengan teknologi *Internet of Things* (IoT) demi peningkatan *quality of life* bagi penggunanya [3].

Mikrokontroler ESP32 merupakan mikrokontroler berbiaya rendah yang dapat mengintegrasikan sensor dan aktuator dalam *Building Automation System* (BAS). Namun, ESP32 memiliki keterbatasan, khususnya dalam hal konversi sinyal analog ke digital (ADC) dan digital ke analog (DAC). Keterbatasan ini seringkali memerlukan kalibrasi tambahan agar dapat memenuhi standar industri. Peningkatan akurasi karena kesalahan *offset* pada ADC dan DAC dapat dilakukan dengan dua cara. Cara pertama adalah kompensasi statis yang menggunakan nilai rata-rata *offset* dari hasil analisis statistik. Solusi ini sederhana tetapi kurang fleksibel karena belum dapat memperbaiki variasi *offset* yang besar, khususnya pada DAC yang memiliki tingkat variabilitas tinggi. Cara kedua adalah kompensasi dinamis yang menggunakan *closed-loop control* dalam kalibrasi otomatis. Solusi ini lebih akurat karena dapat menyesuaikan *offset* berdasarkan kondisi aktual setiap saat. Kompensasi dinamis juga mengandalkan modul PWM (*Pulse Width Modulation*) dalam pengaturan kompensasi *offset* secara lebih presisi [4].

Pembacaan sensor yang akurat pada ESP32 dapat diperoleh dengan membandingkan hasil sensor terhadap nilai standar yang sudah diketahui. Pengujian ini dilakukan melalui perbandingan hasil pengukuran sensor dengan alat ukur standar, seperti sensor suhu yang dibandingkan dengan termometer standar.

Pengujian presisi dilakukan dengan pengukuran berulang dalam kondisi yang sama guna melihat konsistensi sensor dalam memberikan hasil. Hasil pengujian ini menunjukkan seberapa konsisten sebuah alat ukur dalam memberikan nilai yang sama pada pengukuran berulang [5].

Sensor SCT-013 digunakan dalam pengukuran arus listrik yang mengalir melalui sebuah konduktor tanpa memutus rangkaian listrik. Sensor ini termasuk sensor non-invasif yang melakukan pengukuran arus tanpa kontak langsung dengan arus listrik melalui induksi magnetik di sekitar kabel konduktor. Kalibrasi diperlukan guna memastikan hasil pengukuran yang akurat. Proses kalibrasi melibatkan perhitungan nilai resistansi beban (*burden resistor*) yang tepat, sehingga tegangan keluaran dari sensor SCT-013 dapat diterjemahkan menjadi nilai arus listrik yang akurat. Sistem ini juga menggunakan AC-DC *Converter* HLK-PM01 dalam pengubahan arus AC (*Alternating Current*) menjadi arus DC (*Direct Current*) sebagai *power supply* dari sistem tersebut [6].

Kalibrasi sensor SCT-013 dalam pengukuran arus dilakukan di dalam *software* ESP32. Kalibrasi ini memanfaatkan *library Emon* dalam pengukuran arus melalui "emon.Irms", dan menggunakan nilai "1.52" pada "currCalibration" dalam kalibrasi sensor SCT-013 [7].

MQTT berperan dalam komunikasi data antara ESP32, Raspberry Pi, dan perangkat pengguna pada sistem SCADA (*Supervisory Control and Data Acquisition*). MQTT merupakan protokol komunikasi berbasis *publish/subscribe* yang memungkinkan komunikasi dua arah. Dalam sistem SCADA, Raspberry Pi berperan sebagai server lokal yang menjalankan Mosquitto, sebuah perangkat lunak *broker* MQTT. *Broker* ini bertugas dalam penerimaan pesan dari perangkat *client* seperti ESP32 dan pendistribusiannya ke *client* lain yang telah *subscribe* pada topik yang sesuai. ESP32 berperan sebagai *client* MQTT yang melakukan publikasi data sensor seperti suhu, kelembapan, tekanan, dan intensitas cahaya ke *broker* MQTT yang dijalankan di Raspberry Pi. *Broker* MQTT kemudian melakukan penerusan pesan ke semua perangkat *client* lain yang telah *subscribe* pada topik yang relevan. ESP32 tidak hanya mengirimkan data sensor, tetapi juga dapat menerima perintah

dari *broker* MQTT yang dikirim oleh perangkat pengguna, seperti komputer atau *smartphone*, dalam pengendalian perangkat rumah [8].

MQTT juga dimanfaatkan dalam sistem Lo-BEMS (*LoRa-Building Energy Management System*). Sistem ini menggunakan Laravel sebagai server. Data yang diterima oleh *Broker* MQTT diteruskan ke server Laravel dalam pemrosesan dan penyimpanan di *database*. Laravel berperan dalam penyiapan API yang digunakan dalam pengambilan data dari perangkat IoT melalui komunikasi MQTT. API Laravel memungkinkan pengelolaan data secara efisien dengan penyediaan *endpoint* yang menerima data dari *broker* MQTT dan menyimpannya ke dalam basis data. Laravel memanfaatkan data yang dikirim melalui MQTT dalam penampilan informasi kepada pengguna melalui antarmuka berbasis *web*. Data mengenai konsumsi energi dan status perangkat ditampilkan dalam bentuk grafik dan tabel pada *dashboard* yang dibuat dengan Laravel [9].

Laravel *Framework* juga digunakan dalam pembuatan aplikasi *web* dan juga bertindak sebagai *web server* utama yang menangani permintaan dari *client* atau pengguna. *Framework* ini berfungsi sebagai pengelola seluruh data yang diterima dari sensor-sensor, seperti suhu, kelembapan, dan intensitas cahaya, yang dikirim oleh ESP32 melalui protokol MQTT. Laravel digunakan dalam fungsi-fungsi *development* umum seperti *routing*, autentikasi, sesi, dan *caching*, yang mempermudah pengelolaan otorisasi dan pengontrolan akses ke semua *resource* dalam sistem. Laravel juga digunakan dalam penampilan data dalam bentuk tabel dan grafik melalui komponen *controller* seperti *User Controller*, *Table Controller*, dan *Chart Controller*. Data yang dikirim melalui MQTT disimpan dalam *database* SQLite yang terhubung dengan Laravel melalui komponen *model*. Laravel merupakan *framework* fleksibel yang dapat digunakan dalam berbagai kebutuhan, tak terkecuali dalam pengelolaan dan pemrosesan data IoT, serta penampilan informasi kepada pengguna melalui GUI (*Graphical User Interface*) *web*. *Framework* ini juga mempermudah manajemen otorisasi dan akses data dalam sistem pemantauan *real-time* [10].

Pengembangan sistem pemantauan *charging station* berbasis IoT secara *real-time* dilakukan dalam pemantauan status berbagai *charging station* dan membantu

pengguna kendaraan listrik menemukan *charging station* terdekat dengan cepat dan efisien. Mikrokontroler ESP32 yang dilengkapi dengan modul GPS digunakan dalam sistem ini guna mendapatkan data lokasi kendaraan listrik. Data ini diproses dan dikirimkan ke server Blynk dalam penyajian kepada pengguna melalui *mobile app*. Sistem ini juga menggunakan protokol MQTT sebagai media komunikasi data antara alat dan server. Penghitungan jarak terpendek antara lokasi kendaraan dan *charging station* terdekat dilakukan menggunakan rumus Haversine berdasarkan koordinat GPS. Data jarak ini digunakan dalam perkiraan waktu tempuh ke stasiun terdekat dengan mempertimbangkan kecepatan kendaraan saat ini. Sistem ini berhasil mengurangi waktu yang dibutuhkan pengguna kendaraan listrik dalam pencarian stasiun pengisian terdekat dan pengoptimalan konsumsi daya kendaraan [11].

Pengembangan *charging station* universal yang dilengkapi dengan BMS (*Battery Management System*) menggunakan BMS dalam pemantauan parameter penting seperti arus, tegangan, dan suhu baterai. Sistem ini menggunakan STM32 sebagai mikrokontroler yang mengontrol seluruh proses pengisian daya dan distribusi arus ke kendaraan. STM32 dipilih karena memiliki banyak pin PWM (*Pulse Width Modulation*). Sistem ini juga menggunakan ESP32, namun tidak sebagai mikrokontroler sistem *charging station* melainkan sebagai mikrokontroler sistem BMS, atau *client* IoT dalam penghubungan BMS dengan aplikasi Blynk. *Charging station* dalam sistem ini menyediakan 4 *slot* pengisian daya dengan voltase yang berbeda (12V, 24V, 48V, 62V), sehingga dapat mengisi daya berbagai jenis kendaraan sekaligus. BMS digunakan dalam pemantauan berbagai parameter kritis dari baterai EV seperti arus, tegangan, dan suhu. Sistem ini berfungsi dalam memastikan bahwa baterai diisi dalam kondisi yang aman, sehingga mencegah kerusakan akibat pengisian berlebihan (*overcharging*), pelepasan berlebihan (*over-discharging*), atau suhu yang terlalu tinggi. Pengguna dapat melihat kondisi baterai mereka melalui aplikasi Blynk, yang menampilkan data yang dikirim oleh ESP32. Hal ini memungkinkan pengguna dalam pemantauan kesehatan baterai, prediksi kapan pengisian penuh akan tercapai, dan penerimaan notifikasi jika terjadi masalah, seperti lonjakan arus atau tegangan [12].

Smart EV-Hub atau *charging station* pintar yang dikembangkan memungkinkan pengguna dalam pemilihan *port* pengisian, pemantauan status baterai, dan pelaksanaan pembayaran melalui aplikasi *mobile* berbasis Android. Sistem menggunakan Blynk Server dalam pemantauan kondisi pengisian daya dan pemberian informasi mengenai ketersediaan daya dan waktu pengisian. Penggunaan AI dalam sistem ini tidak disebutkan secara detail, namun AI diusulkan dalam pengoptimalan proses pengisian daya dengan analisis data seperti status baterai, arus pengisian, dan kebutuhan energi. AI akan memberikan rekomendasi kapan waktu terbaik dalam pengisian daya, minimalisasi risiko *overcharging* atau *undercharging*, serta pengurangan waktu pengisian. AI juga berfungsi dalam pengelolaan prioritas pengisian jika beberapa kendaraan terhubung ke stasiun pengisian secara bersamaan, memastikan setiap kendaraan mendapatkan daya yang sesuai dengan kebutuhannya. Sistem ini juga mengusulkan penggunaan QR *code* sebagai metode dalam kemudahan pembayaran dan pengelolaan pengisian. Pengguna dapat memindai QR *code* di setiap *port* pengisian dalam pemilihan *port*, pemantauan saldo akun, serta pengaturan durasi pengisian (*timer*) berdasarkan saldo yang ada. Sistem ini juga membahas penggunaan GPS yang memungkinkan sistem dalam pemberian perkiraan jarak dan waktu tempuh ke stasiun pengisian daya terdekat. Namun, integrasi AI maupun GPS dalam sistem IoT ini tidak dibahas dengan detail dan terstruktur, sehingga seolah-olah terdapat dua sistem berbeda yang diusulkan [13].

Berdasarkan uraian referensi penelitian di atas, ringkasan penelitian-penelitian tersebut disajikan sebagai berikut pada Tabel 2.1.

Tabel 2.1 Ringkasan Referensi Penelitian

No	Penulis, Tahun	Judul Penelitian	Tujuan	Metode Penelitian	Perbedaan
1	Channareth Srun, 2024	*Efficient Energy Usage Monitoring System with ESP32 Technology*	Pengembangan sistem pemantauan energi menggunakan ESP32 dengan visualisasi secara *real-time*	Penerapan ESP32 sebagai mikrokontroler dengan sensor arus dan tegangan. Data dikirimkan ke server *Firebase* untuk *real-time monitoring*	Penelitian oleh Channareth Srun menggunakan Firebase sebagai server, sedangkan proyek ini menggunakan *web server* lokal
2	Atefeh Zare, 2020	*Low-Cost ESP32, Raspberry Pi, Node-Red, and MQTT protocol based SCADA system*	Perancangan dan implementasi sistem SCADA berbiaya rendah dan fleksibel untuk *home automation*	Penelitian ini memanfaatkan ESP32 untuk *gateway* sensor, Raspberry Pi sebagai server lokal, dan MQTT untuk komunikasi. Node-RED digunakan untuk visualisasi dengan SQLite	*Database* yang digunakan dalam penelitian oleh Atefeh Zare adalah SQLite, sedangkan *database* yang digunakan dalam proyek ini adalah MySQL