Jurnal Teknik Elektro, Vol. 19, No. 1, Maret 2026: 45-50
P-ISSN 1411-870X, E-ISSN 2656-3304

MQTTLOGGER: SISTEM LOGGING UNTUK MENCEGAH KEHILANGAN DATA SAAT EMERGENCY

Erick Alim Santoso[1], Hendra Khoirul[2], Felix Pasila[1]
[1] Program Studi Teknik Elektro, Fakultas Teknologi Industri, Universitas Kristen Petra, Surabaya, Jawa Timur
[2] PT. Bosch Rexroth, Surabaya, Jawa Timur
E-Mail: erick.alim.santoso@gmail.com, felix@petra.ac.id

Abstrak – Penelitian ini mengembangkan MQTTLogger, sebuah custom palette node berbasis Node-Red pada perangkat industri CtrlX Core, untuk mengatasi kehilangan data sensor akibat gangguan koneksi MQTT dalam sistem IoT industri. MQTTLogger secara otomatis mencatat data ke file CSV dan database InfluxDB lokal saat koneksi terputus, lalu mengirim ulang data secara terstruktur saat koneksi pulih. Dilengkapi filter untuk menghemat penyimpanan, fitur penghapusan file otomatis, dan notifikasi email, sistem ini meningkatkan efisiensi dan kemudahan penggunaan. Pengujian dilakukan dalam tiga skenario: koneksi stabil, terputus, dan pulih, serta implementasi lapangan pada proyek rumah pompa Gunungsari II. Hasilnya menunjukkan MQTTLogger mampu mencegah kehilangan data, menjaga integritas waktu, dan mengurangi penggunaan penyimpanan hingga 95% dengan filter. Antarmuka HTML yang intuitif mempermudah konfigurasi, menjadikan sistem ini solusi efisien, adaptif, dan user-friendly untuk pemantauan data industri berbasis IoT, dengan potensi penggunaan ulang yang tinggi di berbagai proyek.

Kata Kunci – mqtt, data logging, node-red, ctrlx core, iot industri

I. PENDAHULUAN

Indonesia merupakan negara dengan pertumbuhan sektor industri yang pesat. Data dari Badan Pusat Statistik menunjukkan bahwa pada tahun 2023 terdapat lebih dari 33.000 perusahaan industri manufaktur skala menengah dan besar yang aktif di Indonesia [1]. Dalam menghadapi tantangan era Industri 4.0, perusahaan dituntut untuk meningkatkan efisiensi dan akurasi proses produksi, salah satunya melalui pemanfaatan teknologi otomasi dan sistem pemantauan berbasis Internet of Things (IoT). Sensor industri digunakan secara luas untuk mengumpulkan data secara real-time yang sangat krusial dalam proses pengambilan keputusan dan pelaporan performa operasional.

Salah satu teknologi utama dalam komunikasi data IoT adalah protokol Message Queuing Telemetry Transport (MQTT). MQTT merupakan protokol publish-subscribe yang ringan, efisien, dan dirancang untuk komunikasi dengan latensi rendah serta handal dalam kondisi jaringan terbatas [2]. Namun, implementasi MQTT sangat bergantung pada kestabilan koneksi jaringan. Ketika koneksi dengan broker MQTT terputus, data sensor yang seharusnya dikirim dapat hilang dan tidak tersimpan. Hilangnya data ini menjadi kendala serius karena dapat memengaruhi akurasi analisis performa sistem, mengurangi integritas data historis, dan menyebabkan bias dalam pengambilan keputusan berbasis data aktual [3].

Beberapa pendekatan telah digunakan untuk mengatasi masalah kehilangan data, salah satunya adalah dengan menerapkan metode moving average untuk merekonstruksi data yang hilang. Namun, metode ini menghasilkan estimasi berdasarkan rata-rata, yang bukan merupakan data aktual. Selain itu, metode ini tidak mampu menangkap fluktuasi data yang signifikan secara akurat [4]. Sehingga metode tersebut kurang tepat untuk keperluan analisis performa sistem yang memerlukan ketelitian tinggi.

Penelitian terdahulu oleh Manowska et al. (2023) menunjukkan pemanfaatan MQTT untuk sistem akuisisi data energi berbasis cloud, tetapi belum mencakup mekanisme penyimpanan data lokal saat koneksi terputus maupun pengiriman ulang data setelah koneksi kembali [5]. Studi lain oleh Moura. et al. (2018) juga hanya berfokus pada pengiriman data secara real-time tanpa mempertimbangkan kasus koneksi MQTT yang tidak stabil [6].

Untuk menjawab permasalahan tersebut, penelitian ini mengembangkan custom palette node berbasis Node-Red yang diberi nama MQTTLogger dan diimplementasikan pada perangkat industri CtrlX Core. MQTTLogger dirancang untuk mencatat data secara otomatis dalam format file CSV saat koneksi ke broker MQTT terputus, dan mengirimkannya kembali secara terstruktur saat koneksi pulih. Didalam MQTTLogger terdapat sistem filter yang mampu menghemat memori dan menghindari pencatatan data yang nilainya sama secara berulang. Selain itu, fitur tambahan seperti penghapusan otomatis file dan pengiriman notifikasi email ditambahkan untuk meningkatkan kemudahan penggunaan dan efisiensi pemantauan.

Inovasi utama dari penelitian ini terletak pada integrasi sistem logging otomatis berbasis koneksi MQTT yang belum tersedia secara default di Node-Red serta penyederhanaan konfigurasi pengguna melalui desain antarmuka berbasis HTML. Dengan demikian, penelitian ini berkontribusi dalam menyediakan solusi efisien, adaptif, dan user-friendly untuk kebutuhan pencatatan dan pemulihan data di sistem industri berbasis IoT.

II. METODOLOGI PENELITIAN

A. Desain Sistem

Penelitian ini menggunakan pendekatan rekayasa sistem (engineering approach) dengan tujuan merancang dan mengimplementasikan sebuah sistem data logging yang handal dan efisien. Sistem ini dikembangkan dalam bentuk custom palette node bernama MQTTLogger yang dijalankan pada perangkat CtrlX Core berbasis Node-Red. Fokus utama dari sistem adalah mencatat data sensor industri secara otomatis saat koneksi ke MQTT Broker terputus, serta mengirimkan ulang data yang tersimpan tersebut ketika koneksi kembali normal. Dengan pendekatan ini, data yang hilang akibat

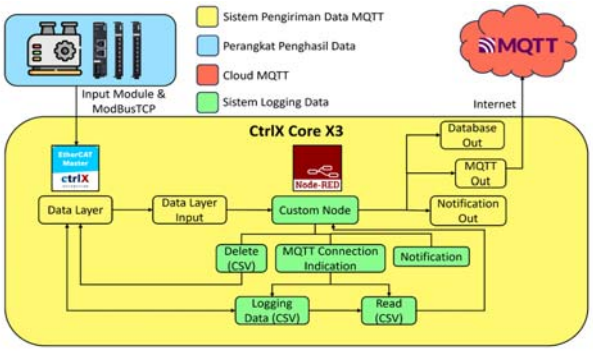
gangguan jaringan dapat diminimalkan dan efisiensi dalam pemrosesan serta penyimpanan data tetap terjaga.

Perancangan sistem pendukung MQTTLogger dibagi dalam dua komponen utama yaitu perangkat keras (hardware) dan perangkat lunak (software). Pada sisi hardware, sistem dibangun dengan menggunakan CtrlX Core X3, yang merupakan perangkat industri berbasis Linux dari Bosch Rexroth dengan kemampuan komunikasi multi-protokol dan pemrosesan data berkecepatan tinggi. CtrlX Core berfungsi sebagai pusat pengolahan dan pengiriman data ke broker MQTT. Untuk membaca data dari sensor industri, digunakan dua modul input, yaitu XI110208 DI8 sebagai digital input (dengan sinyal biner/dry contact) dan XI342204 AI4 sebagai analog input (dengan rentang arus 4–20 mA). Kedua modul tersebut dihubungkan ke XB-EC-12 Coupler, yang berfungsi sebagai penghubung antara modul I/O dengan jaringan EtherCAT serta sebagai penyedia tegangan logika dan daya sistem. Sebagai penyedia konektivitas, digunakan router internet yang menghubungkan CtrlX Core dengan broker MQTT berbasis cloud. Untuk kebutuhan pengujian, data input disimulasikan menggunakan Variabel Frequency Drive EFC3610 dan injektor arus.

Pada sisi software, digunakan beberapa aplikasi pendukung utama. CtrlX I/O Engineering digunakan untuk mengenali dan mengkonfigurasi modul-modul I/O yang terhubung ke CtrlX Core. Setelah konfigurasi berhasil, CtrlX EtherCAT Master bertugas untuk membaca data dari modul-modul tersebut secara periodik. Selanjutnya, Node-Red digunakan sebagai platform pengembangan utama. Node-Red dipilih karena sifatnya yang low-code, modular, dan sangat fleksibel untuk membangun alur data IoT. Untuk penyimpanan dan visualisasi data, digunakan InfluxDB sebagai basis data time-series dan Grafana sebagai platform monitoring real-time berbasis web.

B. Pengembangan Custom Palette Node MQTTLogger

Untuk mencapai fungsionalitas khusus yang tidak tersedia secara default di Node-Red, dikembangkan sebuah custom palette node bernama MQTTLogger. Node ini dikembangkan dengan menggunakan tiga file utama, yakni file JavaScript, file HTML, dan file package.json.



Gambar 1. Diagram Sistem Logging Data Pada CtrlX Core X3

Pendekatan ini membuat MQTTLogger mudah digunakan jika dibandingkan dengan merangkai banyak node secara manual.

C. Skema dan Tahapan Pengujian

Pengujian sistem dilakukan untuk menguji fungsionalitas dari MQTTLogger melalui tiga skenario utama yang mewakili kondisi nyata di lapangan, yaitu:

1. Koneksi normal: Dalam kondisi koneksi MQTT yang stabil, sistem diharapkan dapat meneruskan data real-time dari CtrlX Core ke broker MQTT secara langsung. Monitoring data dilakukan menggunakan Grafana untuk memastikan data diterima secara utuh dan tepat waktu.
2. Koneksi terputus: Dalam skenario ini, koneksi antara CtrlX Core dan broker MQTT sengaja diputuskan (misalnya dengan mematikan router atau mencabut kabel ethernet antara CtrlX dengan router). Sistem diharapkan mencatat semua data ke dalam file CSV lokal maupun database influxDB perangkat dan menunjukkan status terputus melalui editor Node-Red.
3. Koneksi pulih: Setelah koneksi dikembalikan, sistem akan secara otomatis membaca file CSV buffer, mengirimkan ulang data yang hilang ke broker MQTT, dan menghapus file jika telah melampaui batas penjadwalan penghapusan file. Pengujian ini memastikan bahwa tidak ada data yang hilang selama pemutusan koneksi.

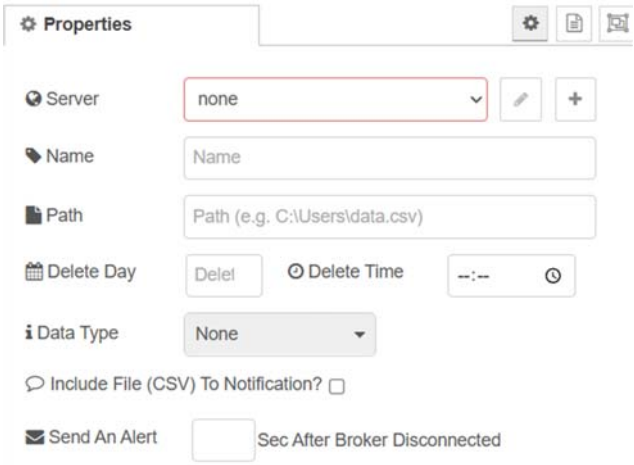
Selain simulasi pengujian, MQTTLogger juga diterapkan secara langsung pada proyek rumah pompa Gunungsari II sebagai uji coba di lingkungan industri yang nyata. Pengujian ini akan memperlihatkan bahwa sistem mampu bekerja secara stabil dan memberikan data logging yang valid pada saat kondisi internet yang fluktuatif.

III. HASIL DAN PEMBAHASAN

A. Hasil Implementasi Sistem MQTTLogger

Gambar 2. Desain Custom Palette Node MQTTLogger

Setelah tahap perancangan dan pengembangan selesai, sistem MQTTLogger diimplementasikan dan dijalankan pada perangkat CtrlX Core X3. Implementasi dilakukan dengan mengunggah direktori node hasil pengembangan ke dalam sistem operasi CtrlX melalui protokol WebDAV (Web Distributed Authoring and Versioning). Setelah direktori selesai diunggah, palette node tersebut dimuat ulang agar Node-Red dapat mengenali dan menampilkan node baru pada menu palette editor.



Gambar 3. Tampilan Setting MQTTLogger Pada Node-Red

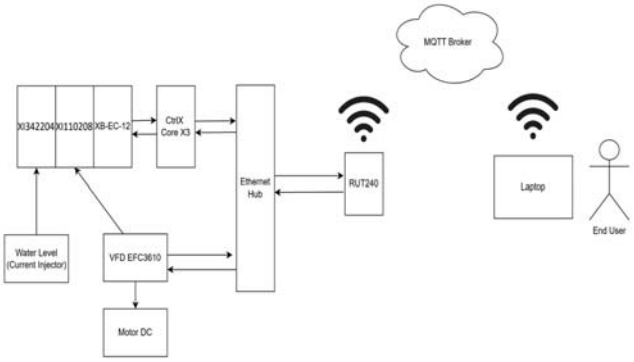
Tampilan antarmuka pengguna dari MQTTLogger memungkinkan pengguna mengatur berbagai parameter melalui editor berbasis HTML yang intuitif. Pada sisi pengguna, konfigurasi dapat dilakukan tanpa mengubah kode inti melainkan cukup dengan memilih jenis data, mengatur path file, mengaktifkan atau menonaktifkan filter, memasang notifikasi email serta mengatur waktu penghapusan file otomatis.

Node MQTTLogger kemudian dihubungkan dengan input dari Data Layer CtrlX Core, yang membaca data dari modul digital input dan analog input melalui protokol EtherCAT. Data ini dikirimkan ke MQTT Broker eksternal melalui koneksi jaringan yang disimulasikan dengan kondisi stabil dan tidak stabil untuk keperluan pengujian.

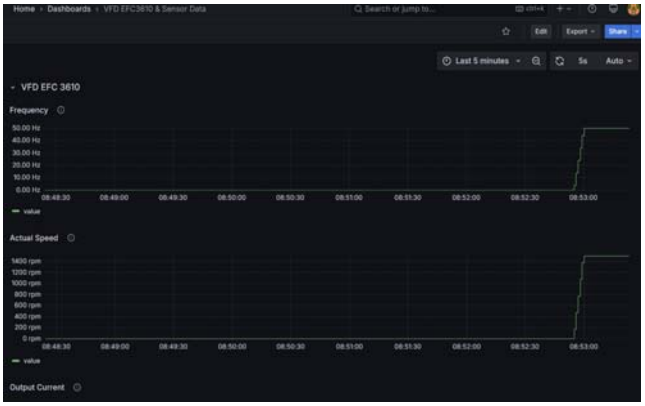
B. Hasil Pengujian Fungsionalitas

Pengujian dilakukan untuk mengevaluasi kinerja sistem dalam tiga skenario utama yakni koneksi normal, terputus, dan pulih. Untuk keperluan pengujian ini, digunakan broker MQTT HiveMQ Cloud sebagai target pengiriman data. InfluxDB digunakan untuk mencatat data yang masuk, dan Grafana digunakan untuk menampilkan data dalam bentuk grafik.

1. Kondisi Koneksi MQTT Stabil



Gambar 5. Simulasi Pengujian Sistem MQTTLogger Saat Terdapat Internet



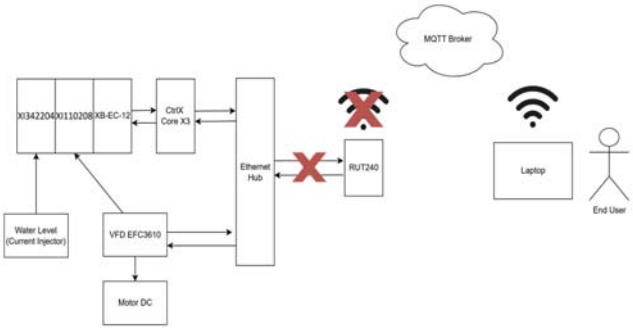
Gambar 6. Grafana Laptop Menerima Data Real Time Dari MQTT dan InfluxDB

Pengujian dilakukan untuk memverifikasi fungsi utama MQTTLogger saat koneksi internet tersedia dan sistem berhasil terhubung dengan broker MQTT. Pada percobaan ini, penulis menggunakan HiveMQ sebagai broker MQTT dan menghubungkan perangkat CtrlX Core ke jaringan Wi-Fi agar dapat melakukan komunikasi data secara real-time.

Langkah-langkah pengujian meliputi konfigurasi MQTTLogger sesuai jenis data yang diuji (digital dan analog), konfigurasi broker HiveMQ, deployment program Node-Red, dan pengamatan aliran data menuju platform visualisasi Grafana yang terhubung dengan InfluxDB pada laptop. Data yang dikirim berasal dari perangkat VFD EFC3610, modul digital input, dan modul analog input.

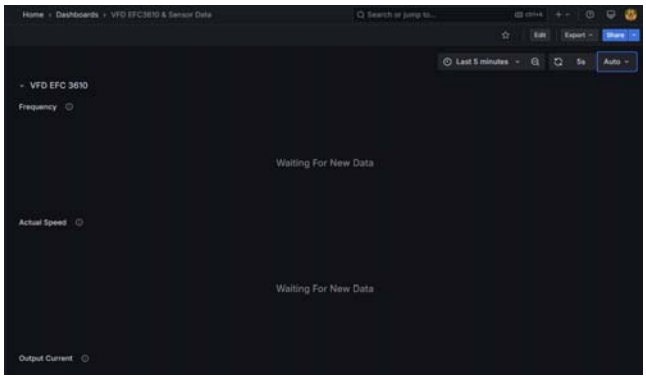
Hasil pengujian menunjukkan bahwa semua data berhasil dikirim dan diterima tanpa kendala melalui protokol MQTT. Seluruh dua belas saluran data yang diuji tercatat dengan baik dalam database InfluxDB lokal. Dengan demikian, pengujian ini menunjukkan bahwa sistem dapat mengirimkan data secara stabil saat koneksi broker tersedia, tanpa perlu menyimpan file lokal.

2. Koneksi Terputus



Gambar 4. Simulasi Pengujian Sistem MQTTLogger Saat Internet Terputus

Pada pengujian koneksi terputus, penulis secara sengaja memutus koneksi internet antara perangkat CtrlX Core dengan router internet untuk mensimulasikan gangguan jaringan menuju broker MQTT. Sistem MQTTLogger secara otomatis mendeteksi kondisi ini melalui parameter KeepAlive sebesar 15 detik. Ketika tidak ada respon dari broker selama periode tersebut, node MQTTLogger akan masuk ke mode disconnect dan menghentikan pengiriman data ke broker. Hal ini ditunjukkan melalui status node di Node-Red serta ketidakhadiran data pada panel Grafana di laptop yang terhubung ke broker.



Gambar 7. Grafana laptop yang Tidak Menerima Data Karena MQTT Terputus

Saat koneksi terputus, seluruh data yang masuk akan dialihkan ke buffer lokal dalam bentuk file CSV dan disimpan di

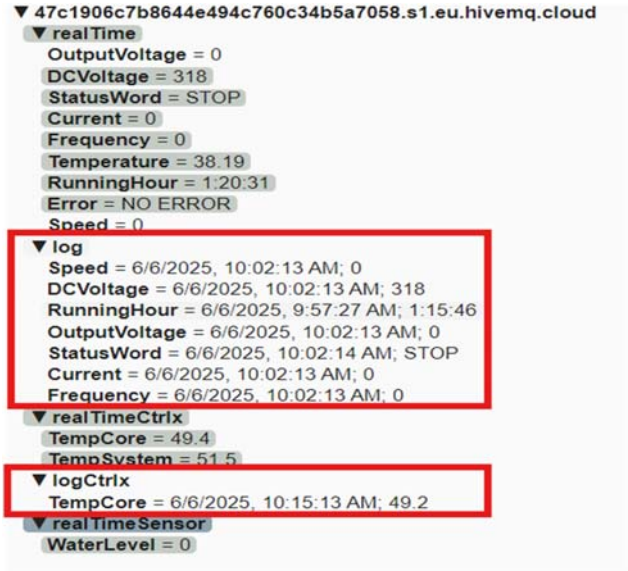
direktori penyimpanan CtrlX Core. Selain itu, data juga dicatat dalam InfluxDB lokal pada perangkat yang sama sebagai cadangan. Penulis juga mengaktifkan fitur filter pencatatan data guna mencegah pencatatan data berulang. Hasilnya menunjukkan bahwa hanya data yang mengalami perubahan yang dicatat sehingga ukuran file tetap efisien meskipun durasi logging cukup panjang.

Sistem juga mengaktifkan fitur notifikasi melalui output node ketiga untuk memberikan peringatan koneksi terputus. Namun, karena kondisi terputus disebabkan oleh jaringan internet, sistem tidak mengirimkan email notifikasi karena tidak dapat mengakses SMTP server. Fitur ini tetap berguna sebagai deteksi awal gangguan, karena jika email notifikasi diterima saat koneksi internet stabil, hal tersebut mengindikasikan bahwa penyebab gangguan bukan pada jaringan, melainkan pada konfigurasi broker atau parameter KeepAlive. Berdasarkan pengujian ini, dapat disimpulkan bahwa MQTTLogger mampu menjalankan fungsi pencatatan data darurat secara efektif dan adaptif terhadap kondisi jaringan yang tidak stabil.

3. Saat Koneksi Pulih

Setelah koneksi internet antara CtrlX Core dan broker MQTT dipulihkan, sistem MQTTLogger secara otomatis mendeteksi pemulihan tersebut. Begitu node berhasil terhubung, MQTTLogger akan menunggu hingga node output kedua (MQTT out) juga terhubung, sebelum mulai mengirimkan kembali seluruh data yang sebelumnya tersimpan dalam file buffer. Proses ini memastikan bahwa tidak ada data yang terlewat atau dikirim sebelum koneksi stabil sepenuhnya.

Data yang dikirim dari file CSV buffer kemudian diterima oleh broker MQTT dan diteruskan ke InfluxDB serta ditampilkan pada panel Grafana di sisi laptop penerima. Semua data historis berhasil di-plot ulang berdasarkan timestamp aslinya, yang menunjukkan bahwa sistem tidak hanya berhasil mengirimkan ulang data, tetapi juga mempertahankan urutan kronologis data. Pengujian ini membuktikan bahwa mekanisme resend buffer berjalan dengan baik tanpa adanya duplikasi maupun kehilangan data.

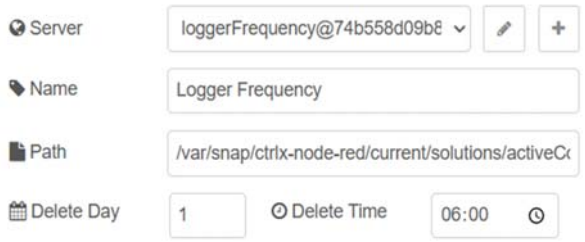


Gambar 8. MQTT Broker Menerima Data Dari File CSV

Selain pengiriman ulang data, MQTTLogger juga mengirimkan notifikasi melalui email kepada pengguna saat node berhasil kembali terhubung ke broker MQTT. Fitur ini memberikan informasi tambahan kepada pengguna bahwa sistem telah pulih sepenuhnya. Dengan demikian, pengujian ini membuktikan bahwa sistem tidak hanya mampu mencatat data saat koneksi terputus, tetapi juga dapat mengelola pemulihan dan integrasi data secara otomatis serta memberikan umpan balik yang jelas melalui notifikasi.

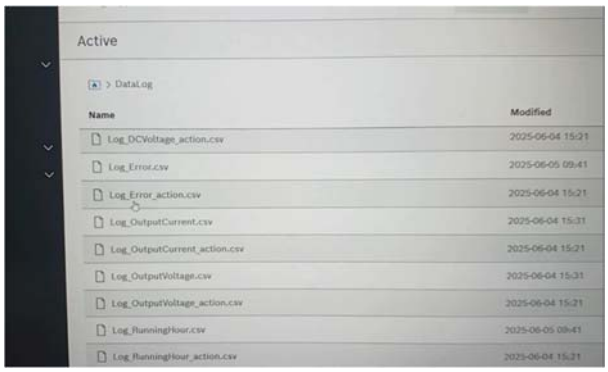
4. Sistem Penghapusan Data Otomatis

Pengujian ini bertujuan untuk memastikan bahwa fitur penghapusan file CSV secara otomatis pada MQTTLogger berjalan sesuai dengan jadwal yang telah dikonfigurasi pengguna. Penulis menetapkan parameter penghapusan dengan mengisi input delete day sebesar "1" dan delete time pada pukul "06:00", yang berarti file akan dihapus satu hari setelah konfigurasi dilakukan, tepat pada pukul 06:00 WIB. Konfigurasi ini tersimpan dalam file JSON sebagai acuan waktu sistem dalam menjalankan penghapusan otomatis.



Gambar 9. Penjadwalan Penghapusan File CSV Frekuensi Pada MQTTLogger

Sebagai bagian dari verifikasi, penulis memastikan bahwa file yang akan dihapus, yaitu Log_Frequency.csv dan Log_Frequency_Action.csv, telah ada di direktori penyimpanan CtrlX Core. Setelah penjadwalan disimpan, CtrlX Core dimatikan dan dibiarkan hingga hari berikutnya melewati waktu penghapusan yang telah dijadwalkan. Pada pagi hari tanggal 05-06-2025 pukul 09:00 WIB, perangkat kembali dinyalakan untuk mengamati hasil penghapusan.



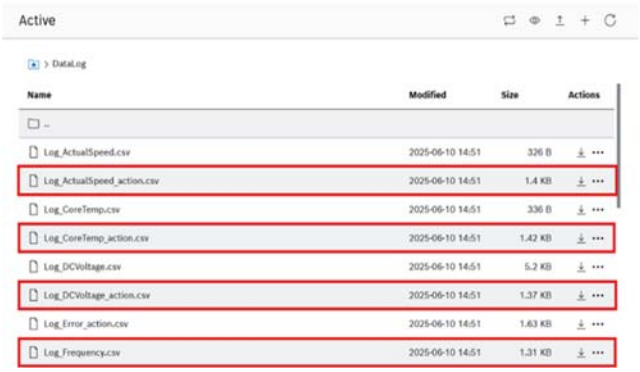
Gambar 10. File CSV Frekuensi Berhasil Dihapus

Hasil pengujian menunjukkan bahwa kedua file CSV telah berhasil dihapus secara otomatis oleh sistem sesuai waktu yang dijadwalkan sebelumnya. Setelah penghapusan selesai, MQTTLogger secara otomatis menetapkan ulang jadwal penghapusan berikutnya. Pengujian ini membuktikan bahwa fitur penghapusan otomatis berjalan dengan baik dan dapat berfungsi meskipun perangkat tidak aktif saat waktu penghapusan berlangsung. Fitur ini sangat berguna dalam menjaga efisiensi ruang penyimpanan pada perangkat edge seperti CtrlX Core.

5. Fitur Logging Data Sesuai Input Pengguna

Pengujian ini bertujuan untuk mengevaluasi kemampuan MQTTLogger dalam menjalankan fungsi pencatatan data (logging) secara terjadwal atas permintaan pengguna. Fitur ini memungkinkan pengguna untuk memperoleh data sebanyak mungkin dalam rentang waktu tertentu, misalnya untuk keperluan analisis mendalam atau pengujian khusus. Dalam pengujian ini, penulis mengatur jadwal pencatatan data selama satu menit menggunakan antarmuka dashboard di Node-Red yaitu dimulai pada pukul 14:51 dan berakhir pada 14:52 WIB, tanggal 10 Juni 2025.

Sebelum jadwal dimulai, file buffer *_action.csv belum terbentuk karena belum ada perintah logging yang aktif. Setelah waktu pencatatan dimulai, MQTTLogger mulai menyimpan data ke dalam file CSV sesuai waktu yang ditentukan. Sistem mencatat semua data yang masuk selama periode aktif, dan secara otomatis menghentikan proses pencatatan ketika waktu selesai tercapai. Verifikasi dilakukan dengan memeriksa keberadaan dan isi file *_action.csv, yang menunjukkan bahwa data telah tercatat sesuai rentang waktu yang ditetapkan.



Gambar 11. File Logging (_action.csv) Muncul dan Menyimpan Data Logging

Selain menyimpan data ke dalam file lokal, sistem juga dapat secara paralel mengirimkan data tersebut ke database InfluxDB jika node tujuan dipasang. Pengujian ini menunjukkan bahwa fitur logging berdasarkan permintaan pengguna berjalan dengan baik dan fleksibel. Dengan demikian, MQTTLogger tidak hanya berfungsi secara otomatis saat terjadi gangguan, tetapi juga mampu digunakan secara manual untuk mencatat data sesuai kebutuhan teknisi atau operator lapangan.

6. Fitur Pengiriman Ulang Data Buffer

Pengujian ini dilakukan untuk memastikan bahwa MQTTLogger mampu mengirimkan ulang data buffer secara manual ketika sebagian data tidak terkirim ke broker akibat gangguan jaringan atau nilai KeepAlive yang terlalu besar. Dalam kondisi tertentu, sistem masih mempertahankan status koneksi meskipun jaringan sudah tidak stabil, sehingga proses pengiriman buffer berjalan sebelum sistem menyadari bahwa koneksi sebenarnya telah terputus. Hal ini dapat menyebabkan sebagian data tidak sampai ke broker MQTT dan mengakibatkan celah data dalam tampilan Grafana.

Untuk mengatasi hal ini, pengguna dapat melakukan injeksi perintah ulang dengan memberikan nilai true pada variabel msg.resendMQTT. Perintah ini memicu MQTTLogger untuk membaca kembali seluruh isi file CSV buffer dan mengirimkan ulang data dari baris pertama hingga baris terakhir. Hasil pengujian menunjukkan bahwa setelah perintah dikirimkan, data yang sebelumnya tidak muncul dalam grafik Grafana berhasil ditampilkan kembali. Dengan demikian, fitur pengiriman ulang manual ini sangat berguna dalam situasi di mana integritas data historis menjadi prioritas, dan memastikan tidak ada informasi penting yang hilang akibat gangguan jaringan.

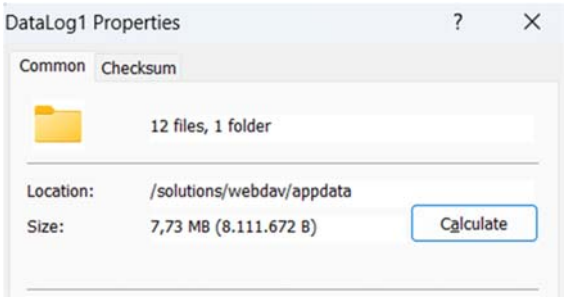
Gambar 12. Tampilan Grafana Saat Ada Data Yang Belum Diterima

Gambar 13. Data hilang sudah berhasil diterima pada Grafana laptop

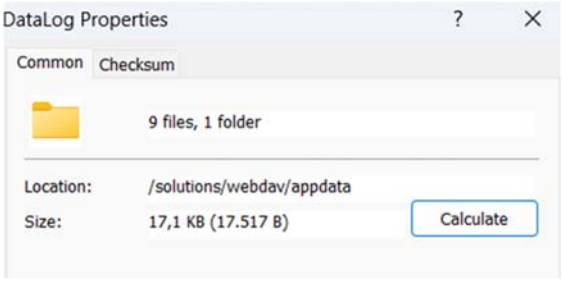
Pengujian ini membuktikan bahwa sistem mampu menyimpan, menjaga urutan waktu, dan mengelola ulang data secara mandiri tanpa intervensi pengguna. Dengan demikian, data historis tetap dapat dimanfaatkan walaupun terjadi gangguan jaringan.

C. Evaluasi Efisiensi Penyimpanan

Efisiensi penyimpanan menjadi aspek penting dalam sistem logging terutama jika digunakan dalam perangkat seperti CtrlX Core yang memiliki keterbatasan ruang penyimpanan. Dalam pengujian selama 8 jam tanpa aktivasi filter (raw logging), file CSV menghasilkan total 7.73 MB data untuk 12 data dari VFD EFC3610. Namun, disaat yang bersamaan diaktifkan juga fitur filter otomatis untuk melakukan logging data serta ukuran file yang dihasilkan hanya sebesar 17.1 KB untuk periode waktu yang sama.



Gambar 14. Besar Folder Logging Tidak Dengan Filter



Gambar 15. Besar Folder Logging dengan Menggunakan Filter

Dengan ini, dapat disimpulkan bahwa filter yang diimplementasikan mampu mengurangi beban penyimpanan lebih dari 95% tanpa mengorbankan integritas informasi yang diperlukan. Hal ini sangat penting untuk penggunaan jangka panjang di perangkat yang terpasang di lapangan seperti sistem monitoring rumah pompa atau industri pengolahan.

D. Penerapan Lapangan dan Relevansi Industri

MQTTLogger telah diuji secara nyata dalam implementasi lapangan pada proyek rumah pompa Gunungsari II Surabaya. Sistem digunakan untuk mencatat status operasional dari soft starter pompa melalui power meter. Koneksi ke broker MQTT menggunakan jaringan internet RUT240 yang tidak selalu stabil. Dalam praktiknya, sistem dapat mencatat gangguan jaringan, menyimpan data secara lokal, dan mengirimkannya ulang ketika koneksi membaik. Karena MQTTLogger dikemas dalam bentuk custom node, sistem ini dapat digunakan kembali di proyek lain hanya dengan drag-and-drop di dalam Node-Red. Hal ini mendukung pengembangan sistem IoT yang modular dan mudah dirawat.

IV. KESIMPULAN

Berdasarkan hasil implementasi dan pengujian yang dilakukan pada bagian pengujian serta proyek rumah pompa, dapat disimpulkan bahwa sistem MQTTLogger berhasil memenuhi tujuan utama penelitian yaitu mencegah kehilangan data sensor saat koneksi MQTT terputus dengan mencatat data secara otomatis ke file CSV dan database influxDB lokal. Sistem juga mampu mengirimkan ulang data buffer secara bertahap saat koneksi pulih, lengkap dengan timestamp untuk menjaga konsistensi data. Efisiensi sistem terjaga melalui penerapan filter otomatis, serta fitur penghapusan otomatis file yang secara keseluruhan mendukung kestabilan dan kinerja jangka panjang. Selain itu, antarmuka konfigurasi yang sederhana dan intuitif membuat MQTTLogger mudah digunakan oleh teknisi serta mempercepat pengembangan sistem pemantauan yang kompleks melalui penggunaan custom palette node yang telah tersedia.

DAFTAR PUSTAKA

[1] Badan Pusat Statistik Indonesia. ”Direktori industri manufaktur Indonesia 2023”. 2023. Available at: https://www.bps.go.id/id/publication/2023/09/29/8c2d8435fe0c552c6ffdc528/direktori-industri-manufaktur-indonesia-2023.html

[2] Suthar, A., Kolhe, K., Gutte, V., & Patil, D. ”Predictive maintenance and real time monitoring using iot and cloud computing”. *2024 5th International Conference on Image Processing and Capsule Networks (ICIPCN)*, 814–820. 2024. Available at: https://doi.org/10.1109/ICIPCN63822.2024.00141

[3] Kang, H. “The prevention and handling of the missing data”. *Korean Journal of Anesthesiology*, *64*(5), 402. 2013. Available at: https://doi.org/10.4097/kjae.2013.64.5.402

[4] XTB. *Moving averages: What are moving averages?* XTB. Available at: https://www.xtb.com/en/education/moving-averages-what-are-moving-averages

[5] Manowska, A., Wycisk, A., Nowrot, A., & Pielot, J. “The use of the mqtt protocol in measurement, monitoring and control systems as part of the implementation of energy management systems”. *Electronics*, 12(1), 17. 2022. Available at: https://doi.org/10.3390/electronics12010017

[6] Moura, V. V. de, Pereira, R. I. S., & Jucá, S. C. S. “Iot embedded system for data acquisition using mqtt protocol”. *International Journal of Computer Applications*, 182(11), 1–4. 2018. Available at: https://doi.org/10.5120/ijca2018917736